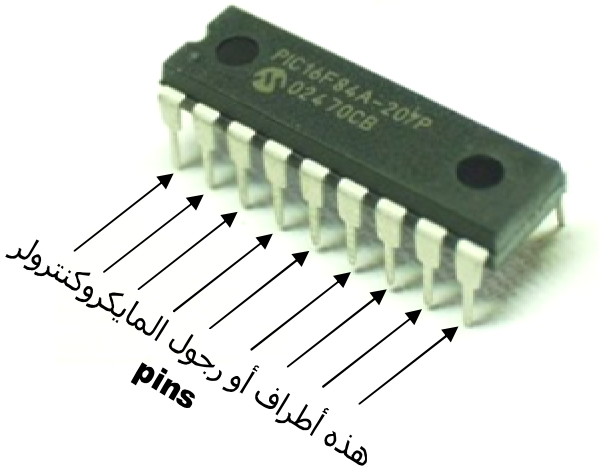
ما هو المايكروكنترولر :

يشبه الدائرة المتكاملة IC كما هو واضح بالشكل ولكنه يمتاز بعدة مميزات عن بقية الدوائر المتكاملة الأخرى من هذه المميزات :-

- أن الدوائر المتكاملة الأخرى لها وظيفة محدده تقوم بها فقط (وقليل منها لها عدة وظائف) أما المايكروكنترولر فليس له وظيفة واحدة فقط بل يمكن أن يكون له عشرات الوظائف والميزة الأكبر أن هذه الوظائف تستطيع تحديدها أنت عن طريق عملية البرمجة .
- بالإضافة إلى تميزه الكبير من خلال مكوناته الداخلية .

المايكروكنترولر من الداخل :

المايكروكنترولر من الداخل ما هو إلا كمبيوتر صغير Mini-Computer حيث يتكون من وحدة معالجة Processor وكذلك ذاكرة عشوائية RAM و ذاكرة من النوع ROM بالإضافة إلى وحدة تخزين يوضع عليها البرامج والبيانات (كما في الكمبيوتر الشخصي العادي) بالإضافة إلى وحدة الإدخال والإخراج حيث يكون الإدخال والإخراج عن طريق بعض أطراف المايكروكنترولر (Pins) - وتسمى الأطراف أحيانا بالرجول legs - كما هو موضح بالصورة التالية :



حيث يخرج المايكروكنترولر الإشارات الكهربية وكذلك يستقبلها عن طريق هذه الأطراف ، كل ذلك تتحكم فيه أنت كما تريد من خلال عملية البرمجة .

لماذا اسمه مايكروكنترولر MicroController ؟

لأن وظيفته هي التحكم سمي بـ controller حيث يتحكم بالدائرة الالكترونية وما تحتويه من عناصر الكترونية ويجري عليها العمليات المختلفة كما يشاء فهو بمثابة المدير الذي يأمر الموظفين بالعمل ويدير عملهم جميعا .

وسمي مايكرو(Micro) لأن حجمه صغير جدا بالنسبة لإمكانياته الكبيرة - ومن المعلوم أن كلمة مايكرو تعني أن القيمة مضروبة في ١٠ أس سالب ٦ - لذلك يطلق عليه البعض أحيانا (المتحكم الدقيق) أو المتحكم الصغير أو نكتبها قراءة لإسمه الانجليزي(المايكروكنترولر).

ما هي إمكانيات المايكروكنترولر :

إمكانيات المايكروكنترولر كثيرة جداً ، حيث أنه كما ذكرت يستطيع التحكم في العناصر الالكترونية أو الدوائرة الالكترنية كما يشاء بل ويتعامل أيضا مع الأجهزة الكهربية المختلفة وللتعرف على إمكانيات وقدرات المايكروكنترولر سنطرح بعض الأمثلة والمشاريع التي يمكن للمايكروكنترولر تنفيذها :-

مشروع التحكم في أجهزة المنزل عن طريق المايكروكنترولر . حيث يمكن هذا المشروع المستخدم من التحكم في أجهزة المنزل مثل المصابيح الكهربية وأجهزة التبريد وفتح الباب وغلقه عن طريق الريموت كنترول فعند الضغط على زر معين تقوم الدائرة الالكترونية التي تحتوي على المايكروكنترولر بتشغيل المصابيح الموجودة في الغرفة مثلا وعند الضغط على زر آخر يقوم المايكروكنترولر بغلق المصابيح وكذلك زر للتحكم في تشغيل وإطفاء الثلاجة أو الغسالة مثلا ... وهكذا .

يستطيع المايكروكنترولر ايضا أن ننفذ به دائرة تجعلنا نتحكم في تشغيل وإطفاء الأجهزة بعد مدة معينة فمثلا نحدد وقت وليكن عشر دقائق مثلاً يقوم فيها المايكروكنترولر بتشغيل المكيف أو المروحة الكهربية وبعد عشر دقائق يفصل التيار الكهربي عنها كما يمكننا جعل المستخدم هو من يحدد المدة المطلوبة ويزيدها أو ينقصها بواسطة مفاتيح switches موصلة أيضا بالمايكروكنترولر ، ونجعل الوقت المتبقي لفصل التيار الكهربي يظهر على شاشة LCD (أيضا الشاشة موصلة بالمايكروكنترولر) حيث يتم التحكم فيما يظهر على الشاشة بواسطة الميكروكنترولر .

ونستطيع أيضا أن نصمم دائرة يتم فيها قياس درجة حرارة المكان وعرضها على شاشة LCD أو seven segment وعند وصول درجة الحرارة لدرجة معينة يقوم المايكروكنترولر بتشغيل جهاز التبريد إلى أن تصل درجة حرارة المكان إلى درجة معينة فيفصل التيار الكهربي عن جهاز التبريد كنوع من توفير الطاقة أو تستخدم مثل هذه المشاريع في الحضانات للمحافظة على حياة الطفل كما يمكن استخدام حساس الأكسجين وحساس الرطوبة لتغذية الحضانة بالأكسجين المناسب والرطوبة المناسبة والتحكم في نسبة كل من الحرارة والأكسجين والرطوبة بدقة كبيرة .

نستطيع أيضا تصميم خط إنتاج مصنع باستخدام المايكروكنترولر حيث يتحكم المايكروكنترولر في المواتير الخاصة بالسير وكذلك في الأجهزة المختلفة والعمليات الدقيقة بكل سرعة وبدقة متناهية .

نستطيع أيضا تصميم دائرة تقوم بفتح الباب وغلقه أوتوماتيكيا بمجرد أن تقترب من الباب يفتح وبعد ان تبتعد عنه ينغلق . وكذلك يمكن عمل دائرة تكون بمثابة عداد للزوار تقوم بعد الزائرين الداخلين والخارجين من المنشأة أو المعرض ونحوه كما يمكن استخدامها أيضا في خط انتاج المصنع حيث تقوم بعد أعداد المنتجات التي تم إنتاجها .

نستطيع كذلك تصميم دوائر الأمن والحماية والتي تقوم بتشغيل إنذار معين عند دخول السارق بل وربما منعه من عملية السرقة وحبسه إن لزم الأمر .

أيضا يمكننا عمل مشروع هدفه أن لا يدخل الموظفين إلى الشركة أو المؤسسة التابعين لها إلا بواسطة بطاقة دخول فعند وضع الموظف للبطاقة يفتح له الباب ويتمكن من الدخول ويتم تسجيل الوقت والتاريخ الذي دخل فيه وكذلك في الانصراف ، وبذلك تتم عملية تسجيل الحضور والانصراف أوتوماتيكاً بواسطة دائرة الكترونية تحتوي على المايكروكنترولر وهذه الدائرة موصلة بالحاسب الآلي لترسل له المعلومات لكي يخزنها في قاعدة بيانات ضخمة ، كما يمكننا أن نجعل بعض المكاتب أو الخزائن لا يمكن دخولها إلا لموظفين معينين .

نستطيع كذلك التحكم في المواتير من ناحية السرعة وكذلك عدد اللفات التي تلفها فمثلا في مشاريع خط الانتاج (يوجد سير يحرك المنتج من مكان لأخر ليجرى عليه العمليات المختلفة) هذا السير يتحرك بمواتير نستيطع التحكم في سرعتها وفي وقت توقفها وفي عدد لفاتها للحصول على أجود وأدق النتائج ، وكذلك عن صناعة الرجل الآلي Robots لابد من التحكم الدقيق في المواتير المختلفة بكل دقة وكذلك اتخاذ قرارات معينة في أوقات معينة ، فمثلا إذا أردنا أن يسير الروبوت robot بحيث لا يصطدم بالحواجز ويغير من اتجاه سيره فور وجودها فلابد من استخدام حساسات معينة sensors موصلة بالمايكروكنترولر الذي يأخذ منها معلومات معينة يحولها لقرار معين للمواتير التي يتحكم فيها أيضاً .

نستطيع أيضا عمل آلة حاسبة(عادية أو علمية) ونضيف إليها الإمكانيات المختلفة على حسب ما نريد فمثلا نضيف فيها مثلاً خاصية التحويل من متر إلى سنتيمتر أو أي عملية تحويل أخرى. كما يمكنك جعلها تعمل بحيث عند الضغط على أي زر فيها تصدر صوتا . وهكذا .... كما تريد.

هل تريد عمل ألعاب للأطفال تعمل من خلال الريموت كنترول مثلا سيارة تسير بالريموت كنترول تتحكم في اتجاهها وكذلك في سرعتها .... هل تريد أن تجعل الرموت يتحكم فيها عن بعد كبير يصل إلى عشرات الأمتار .. تستطيع فعل ذلك باستخدام المايكروكنترولر .

هل تريد التحكم في بيتك أو في مصنعك عن طريق استخدام خط الهاتف فمثلا تتصل بالهاتف وتضغط الرقم السري وعند ضغطك على رقم معين يقوم المايكروكنترولر الموصل بالهاتف بتشغيل أجهزة التكييف فتدخل بيتك وتجده ذو درجة حرارة مناسبة . أو مثلا تخرج من بيتك ولا تعلم هل تركت أجهزة المنزل تعمل أم لا فتتصل بالهاتف وتضغط رقم معين فيقوم بإطفاء جميع أجهزة المنزل .. بل وهل تريد أن تتحكم بمصنعك حتى لو كان في دولة أخرى وتتحكم في الأجهزة التي به وتتابع أخبار المصنع والإنتاج لحظة بلحظة عن طريق اتصالك بالهاتف حيث يرد عليك المايكروكنترولر ... نعم .. صدق تستطيع فعل هذا بالمايكروكنترولر بل ويمكن أن تكون وسيلة الاتصال بأجهزتك ومصنعك عبر الانترنت فتكون الدائرة

الالكترونية الرئيسية لمشروعك لها القدرة على الدخول للانترنت لموقع معين وترسل البيانات إليه أولاً بأول فتتابع جميع العمليات المختلفة لحظة بلحظة وكأنك في نفس المكان بل ويمكنك أن ترسل أوامر وتعليمات من خلال الانترنت يقوم الميكروكنترولر بتنفيذها بعد استقباله إيها من الانترنت .

كل هذه المشاريع يستطيع عملها المايكروكنترولر والكثير الكثير ... فما سبق مجرد أمثلة فقط .

بشرى سارة للقارئ .. كل المشاريع السابقة التي ذكرتها سوف تتعلمها أنت وتتعلم كيفية إنشائها من خلال هذا الكتاب .. إن شاء الله تعالى .. وهناك الكثير والكثير من المشاريع الأخرى ، فقط تابع الكتاب خطوة خطوة .. وتمنياتي للجميع بالتوفيق وأسأل الله العلي القدير أن يرزقنا وإياكم العلم النافع وأن يوفقنا لنصرة الإسلام والمسلمين وذلك بأن ننهض علمياً بأنفسنا وأمتنا .

بعد أن تعرفنا على بعض إمكانيات المايكروكنترولر وما يستطيع أن يفعله هيا بنا لنبدأ الرحلة الممتعة والتي سنتعلم فيها برمجة المايكروكنترولر من النوع PIC فهيا بنا إلى التعرف على هذا العالم الجميل والممتع .

الأشياء التي ستحتاجها لإحتراف برمجة المايكروكنترولر:-

١- حاسب آلي (كمبيوتر) عادي

٢- جهاز برمجة (إذا لم تمتلك واحداً ... ارجع للملحق الخاص بجهاز البرمجة في آخر الكتاب).

٣- البرامج المستخدمة في عملية البرمجة وسنستخدم في هذا الكتاب برنامج MikroC كما ستحتاج البرنامج الذي يكتب على المايكروكنترولر وسنستخدم برنامج Picpgm يمكنك الحصول على روابط التحميل الخاصة بتلك البرامج بالرجوع للملحق الخاص بها في آخر الكتاب.

٤- مايكروكنترولر من النوع PIC وسنستخدم في البداية النوع PIC16f84a وبعد ذلك PIC16f628a ثم PIC16f877a وهناك أنواع أخرى كثيرة سنستخدمها وسنشير إليها في الوقت المناسب.

٥- بعض المكونات الكهربية والالكترونية مثل : مقاومات ، مكثفات ، سفن سيجمنت ، ريلاي ، Testd_Board و أسلاك توصيل (في كل مشروع سنذكر المتطلبات) ملحوظة إذا لم تكن

عندك خلفية عن الأشياء السابقة أو عن الالكترونيات يمكنك الرجوع إلى الملحق الخاص بالمبتدئين والذي ستجده في آخر الكتاب.

٦- اتباع تعليمات (كتاب أسرع طريق لاحتراف برمجة المايكروكنترولر) بدقة والاجتهاد لفهم محتوياته.

٧- الصبر .... نعم الصبر ففي أي مجال إذا أردت التفوق والتميز والاحتراف لابد عليك من التحلي بالصبر . وتذكر قول الله تعالى (إن الله مع الصابرين) وقوله تعالى (والله يحب الصابرين).

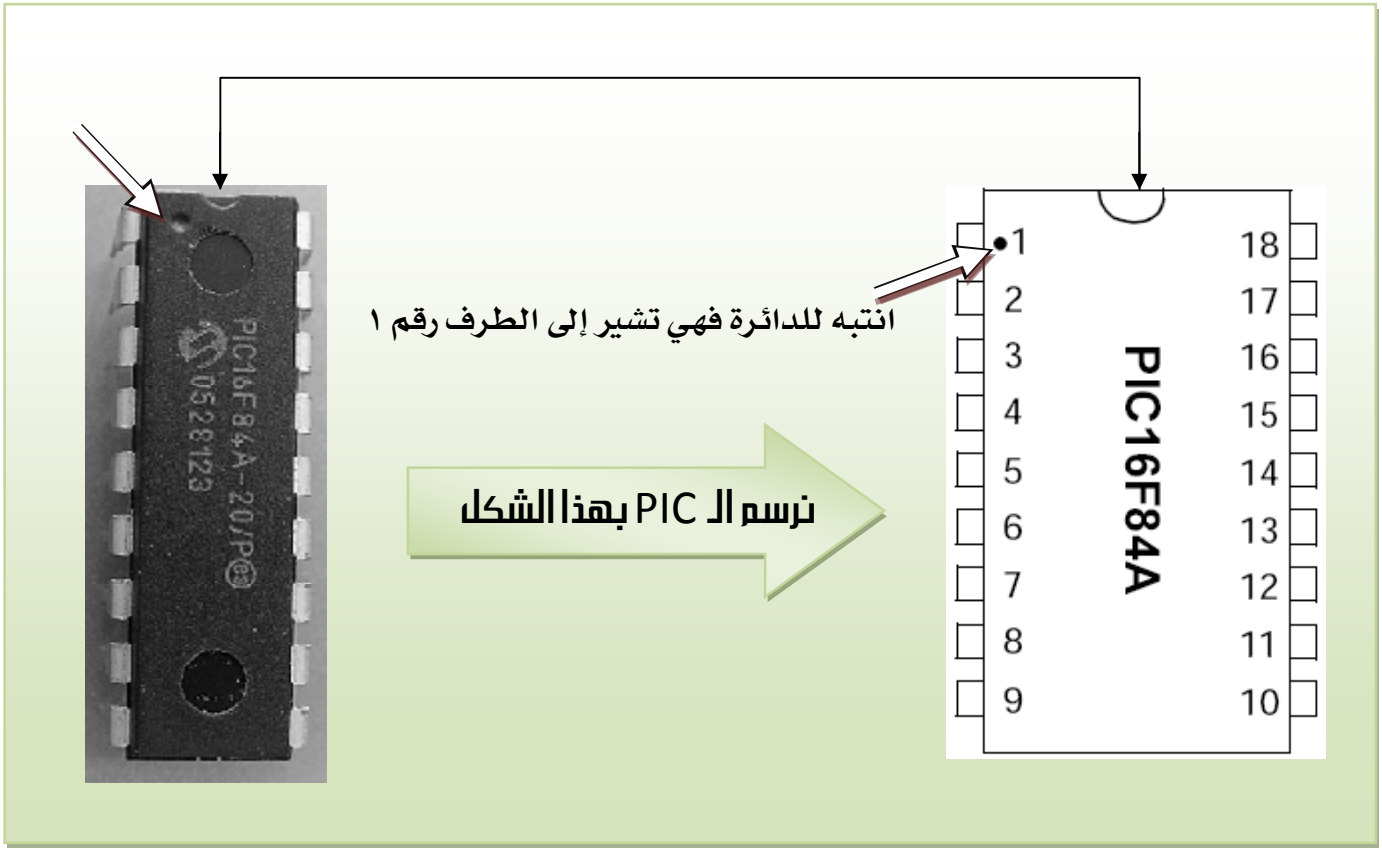
٨- حب القراءة وممارستها... يجب عليك باستمرار أن تقرأ في مجال المايكروكنترولر خصوصًا وفي مجال الالكترونيات عموما ومتابعة كل جديد والاطلاع على المواقع العربية والأجنبية لاكتساب الخبرات والعلم من الآخرين ... ولن يحدث ذلك إلا بحبك للقراءة والاستمتاع بها وتذكر أن أول ما نزل على النبي محمد صلى الله عليه وسلم هو قول الله تعالى (اقرأ) ...

٩- الإستعانة بأهل الخبرة في هذا المجال ... فمن المؤكد ستواجه بعض المشاكل وسينقصك بعض المعلومات .. فحاول الاستعانة بمن له خبرة في ذلك فيمكنك الرجوع لبعض المنتديات العلمية سواءاً العربية أو الأجنبية .. وإذا كنت تعرف شخصاً له خبرة في الموضوع فلا تتردد في سؤاله ... ويجب عليك أن تتذكر دائماً أن العلم لا يأتي لمستحي ولا لمستكبر . فاحذر من هاتين الخصلتين اللتان يحجبان العلم .

١٠- الدعاء وطلب العون من الله سبحانه وتعالى ... فالعلم كله بيد الله يرزقه من يشاء فادعو الله باستمرار أن يرزقك العلم النافع وأن يعينك في رحلة التعلم .

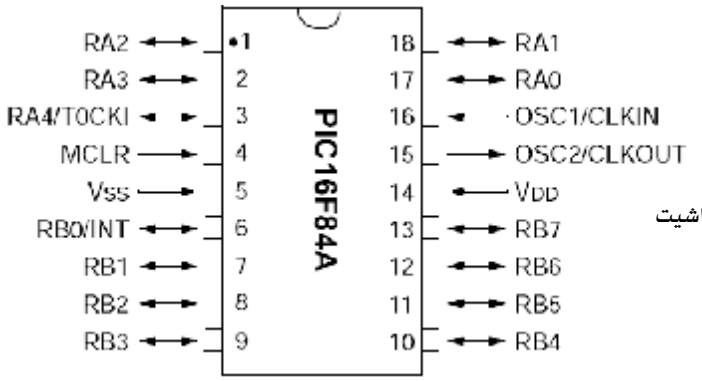
مفاهيم أساسية

توجد أنواع كثيرة من المايكروكنترولر مثل PIC و AVR و8051 ... إلخ ، و سنستخدم في هذا الكتاب مايكروكنترولر من النوع PIC وهو من إنتاج شركة MICROCHIP والطريقة التي سنرسمه بها في الدوائر ستكون كما بالشكل



الطرف رقم ١٧ ، ١٨ ، ١، ٢، ٣ نسميهم PORTA ونطلق عليهم بالترتيب A0 , A1 , A2 , A3 , A4

الأطراف من ٦ إلى ١٣ نسميهم PORTB ونطلق عليهم بالترتيب من B0 إلى B7 (انظر للرسم)



- قد تلاحظ حرف R قبل اسم الطرف ... لافرق

أي أن A0 هي RA0 وهكذا ...

- الرسمة هذه سنرجع إليها كثيرا عند التوصيل وهي مقتبسة من الداتاشيت

الخاصة بـ pic16f84a .

لعلك لاحظت أن الطرف رقم ١٤ مكتوب بجواره VDD وهذا الطرف من المتحكم سنوصل به +٥ فولت ولهذا في الدوائر التي ستكون في الكتاب عندما تجد VDD إعلم أنها تشير إلى جهد موجب خمسة فولت . وسنعلم كيف سنحصل عليه فيما بعد .

وأيضا الطرف رقم خمسة مكتوب بجواره VSS فهذا يشير إلى أنه يوصل بسالب البطارية أي جهد صفر فولت .

هذان الطرفان (VDD,VSS) هما المسئولان عن تغذية المايكروكنترولر و بدونهما لا يمكن أن يعمل .

بقية الأطراف سنتعرف عليها في وقتها إن شاء الله .

ماذا تعني عملية برمجة المايكروكنترولر ؟

هل تعلم أنه عند شراءك للمايكروكنترولر فإنه لن يكون له أي فائدة إذا وضعته في أي دائرة الكترونية ولن يقوم بأي وظيفة إلا إذا قمت بعملية البرمجة قبل وضعه في الدائرة .

وعملية البرمجة هذه هي التي تحدد وظيفته ، ولكي تفهم جيداً ماذا تعني عملية البرمجة إليك هذا المثال:

نفترض الآن أننا لدينا دائرة الكترونية معينة تحتوي على مصباح كهربي أو LED موصل بالمايكروكنترولر فإنه يمكننا أن نضع برنامج معين على الميكركنترولر (عن طريق عملية البرمجة) هذا البرنامج أو الكود سيجعل هذا المصباح أو LED يضيء ثم ينطفئ باستمرار كل نصف ثانية ، وبالتالي ستكون وظيفة الميكروكنترولر هي إضاءة وإطفاء المصباح باستمرار كل نصف ثانية . والآن نفس الدائرة تماماً ودون عمل أي تغيير عليها سوى تغيير البرنامج الذي على المايكروكنترولر (عن طريق عملية البرمجة) حيث سنضع برنامج آخر سيجعل المايكروكنترولر يضيء هذا المصباح أو LED مرتين فقط ثم يظل منطفئاً . إذن عملية البرمجة تؤثر كليا على وظيفة المايكروكنترولر ، وسنشرح عملية البرمجة بتفصيل أكثر في التجربة الأولى بإذن الله تعالى .

- - - - - - - - - - - -

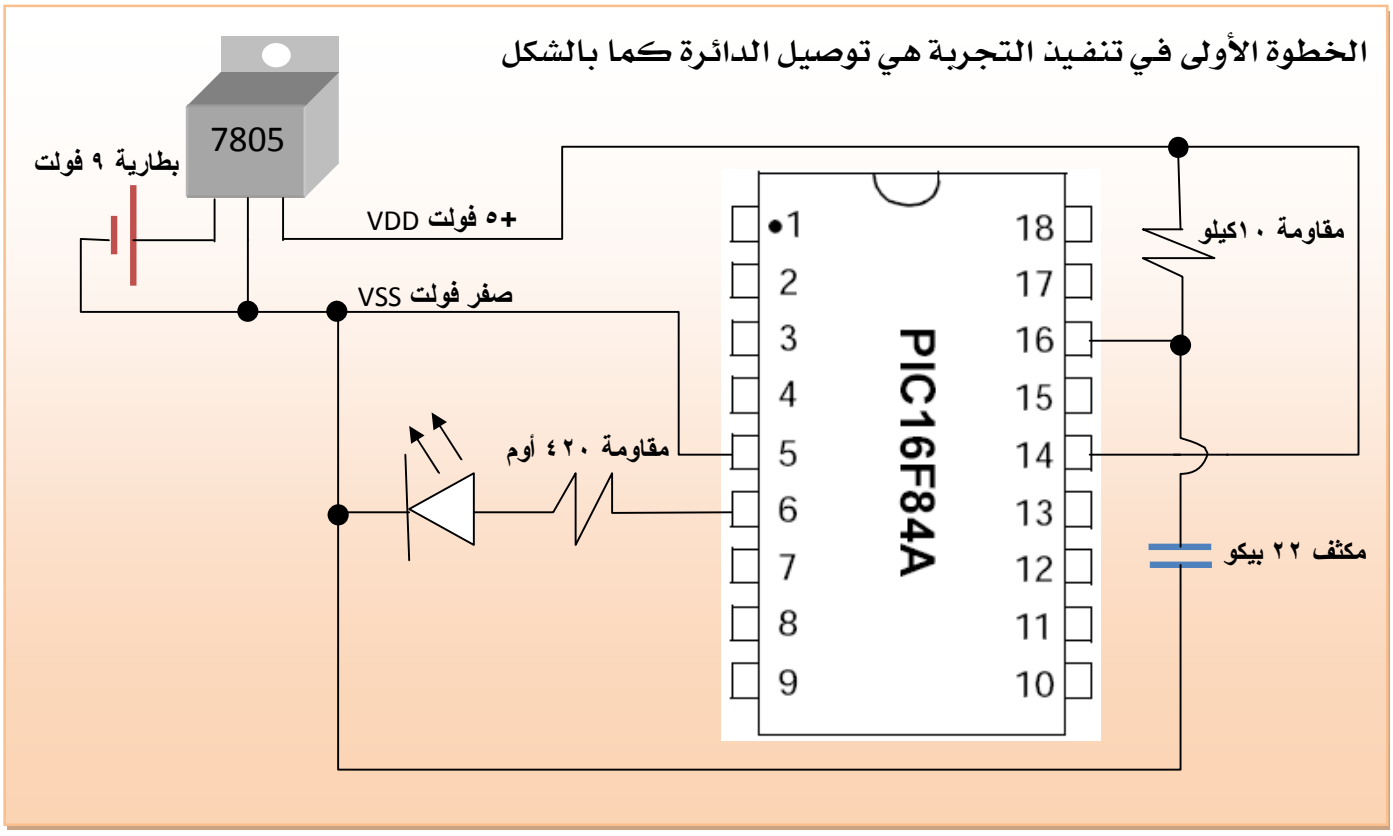
ملحوظة : ستجدني في هذا الكتاب اكتب بعض الكلمات الانجليزية باللغة العربية فمثلا Microcontroller تجدني كتبتها مايكروكنترولر نطقاً لها . وكذلك Led أكتبها ليد أو ليدات (للجمع) وكذلك كلمة bit تجدني أكتبها بت أو بتات (للجمع). وأحياناً ليس نطقا لها مثل PIC تجدني كتبتها بك أو البك أكتبها كذلك لأنها أشتهرت بهذه التسمية بين العرب.

التجربة (١)

هذه أول تجربة لبرمجة المايكروكنترولر وسنستخدم النوع PIC16f84a والهدف من التجربة هو تشغيل LED سنوصله على الطرف رقم B0 . القارئ العزيز : يجب عليك أن تعلم أن التجارب الأولى لها أهمية كبيرة جداً لأنك إذا نجحت في تنفيذها سيصبح الأمر لديك أسهل وأكثر متعة وستخترق هذا العالم وتفتح لك أبوابه وتذلل لك الصعاب .... أتمنى لك التوفيق .

توصيل الدائرة :-

الخطوة الأولى في تنفيذ التجربة هي توصيل الدائرة كما بالشكل



ملاحظات مهمة :-

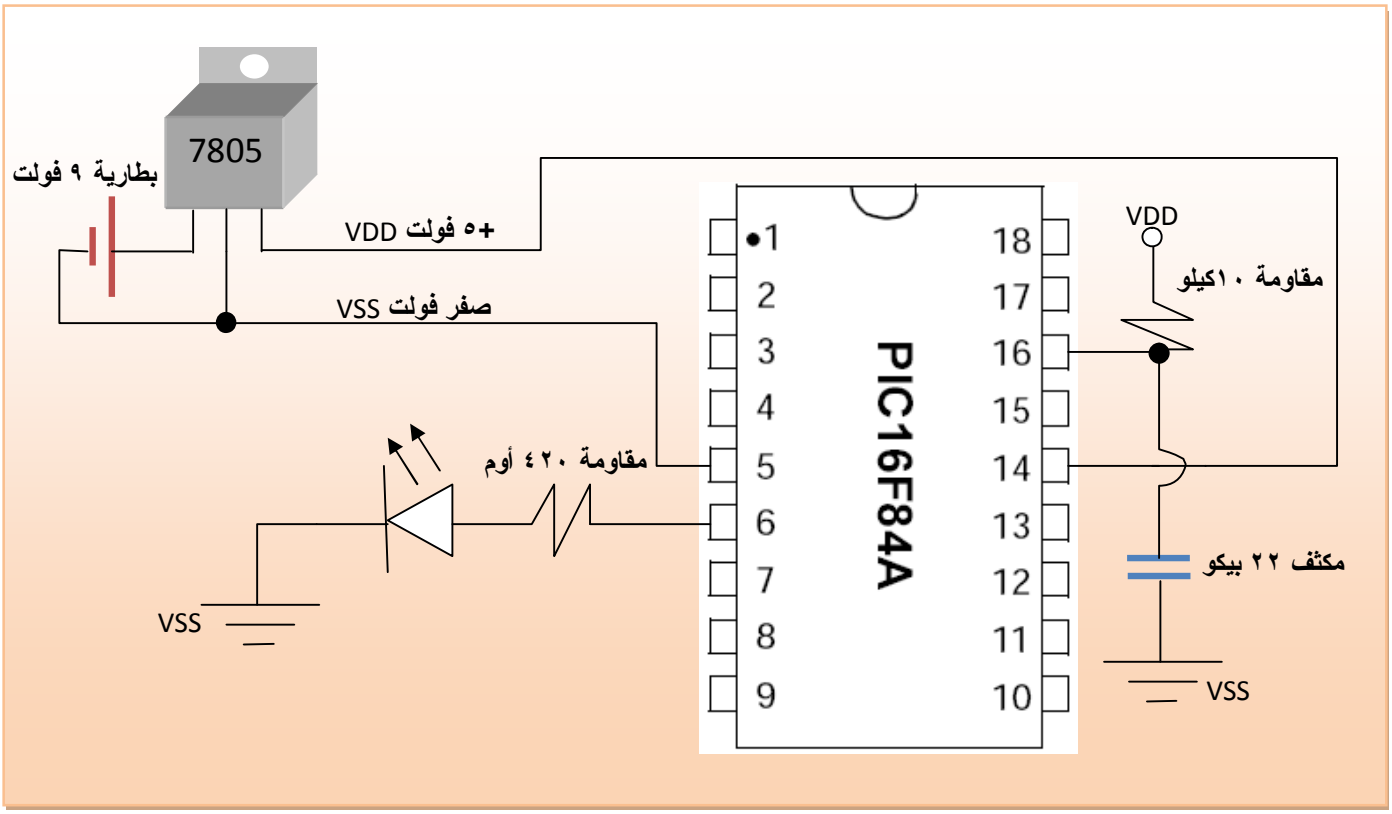
١- البك يحتاج إلى خمسة فولت لكي يعمل وإذا وصلنا به ٩ فولت سوف يتلف لذلك نتبع البطارية بمنظم جهد 7805 والذي يخرج خمسة فولت منتظمة .

٢- يمكن استبدال البطارية واستخدام مصدر جهد آخر power supply يخرج تيار مستمر ٩ فولت أو ٦ فولت بحيث تكون قيمة الخرج لمصدر الجهد هذا أكبر من خمسة فولت.

٣- بدون التوصيلة التي على الطرف رقم(١٦) لن يعمل البك . هذه التوصيلة تعتبر نوع من أنواع المذبذبات ويوجد أنواع أخرى غير المستخدمة في هذه التجربة وسنشرح ذلك لاحقاً بإذن الله .

٤- من المفترض أن البك سوف يخرج على الطرف رقم ٦ جهد مقداره ٥ فولت وهذا الجهد هو الذي يستطيع إخراجه البك من أطرافه ... ولكن إذا تم توصيل الطرف رقم ٦ مباشرة بالـ LED سوف يتلف الـ LED لذلك نضع مقاومة ولتكن قيمتها ٣٣٠ أوم أو ٤٢٠ أوم أو مقاومة قريبة من هذه القيمة ويمكن حساب المقاومة المطلوبة باستخدام قانون أوم وسأوضح ذلك فيما بعد .

٥- الدائرة السابقة يمكن تبسيطها حيث ترسم بالطريقة التالية .

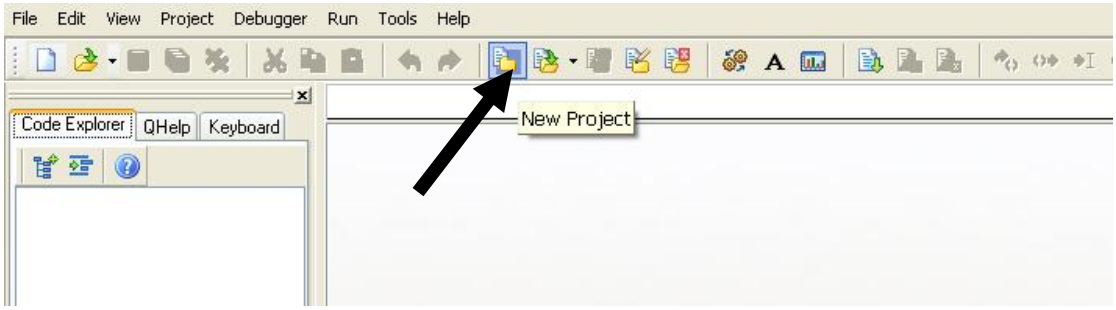


وإذا استخدمت الـ Tested Board ستجدها بهذا الشكل .

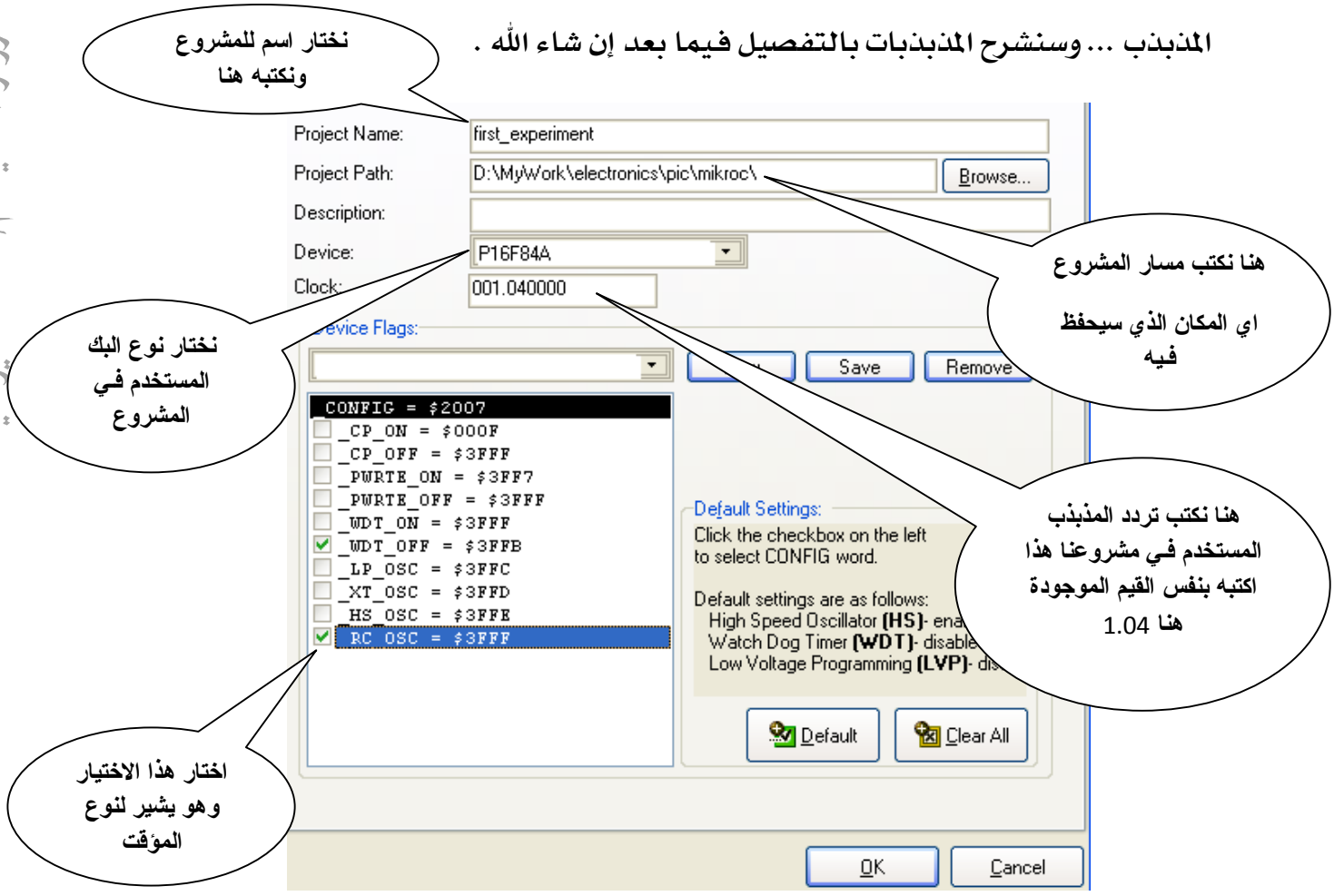
كتابة البرنامج المطلوب :-

الخطوة الثانية هي كتابة البرنامج المطلوب الذي يؤدي الوظيفة التي نريدها ... في هذه التجربة نريد أن نجعل البك يطبق جهد موجب على الطرف B0 ليشغل الليد led الموصل به ... ولنأمر البك بذلك نحتاج أن نخاطبه باللغة التي يفهمها .... إذاً سنستخدم لغة برمجة وسنستخدم في هذا الكتاب لغة MikroC بعد تحميل وتنزيل اللغة أي بعد عمل setup و install اتبع الخطوات التالية :

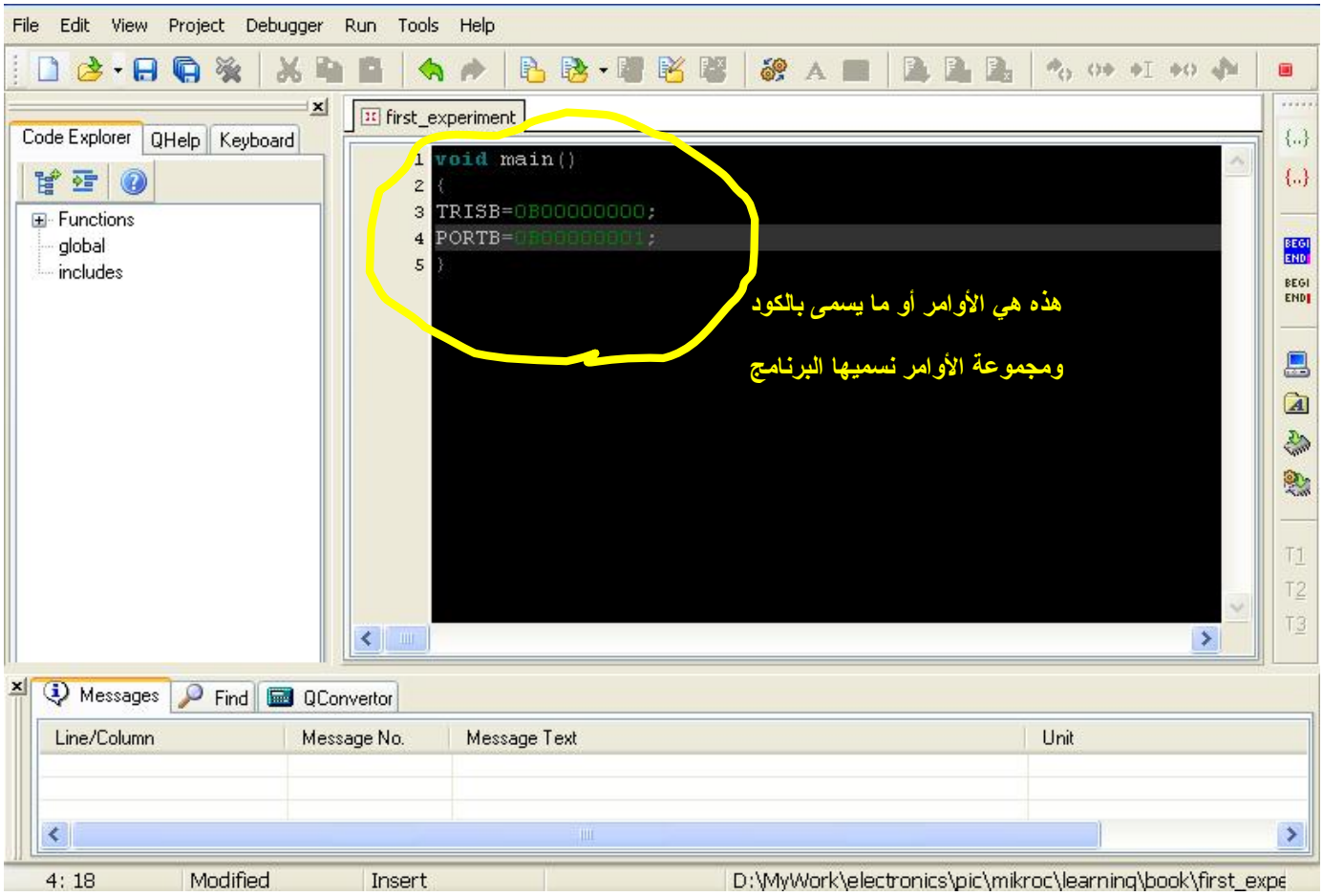
١- نقوم بفتح البرنامج(لغة البرمجة) ثم ننشئ مشروع جديد بالضغط بالماوس على  كما بالشكل .



٢- نقوم بكتابة اسم المشروع ونحدد المكان الذي سيحفظ فيه ونحدد نوع البك المستخدم وكذلك تردد المذبذب ... وسنشرح المذبذبات بالتفصيل فيما بعد إن شاء الله .



٣- الآن ستظهر لنا شاشة سنكتب فيها البرنامج كما بالشكل



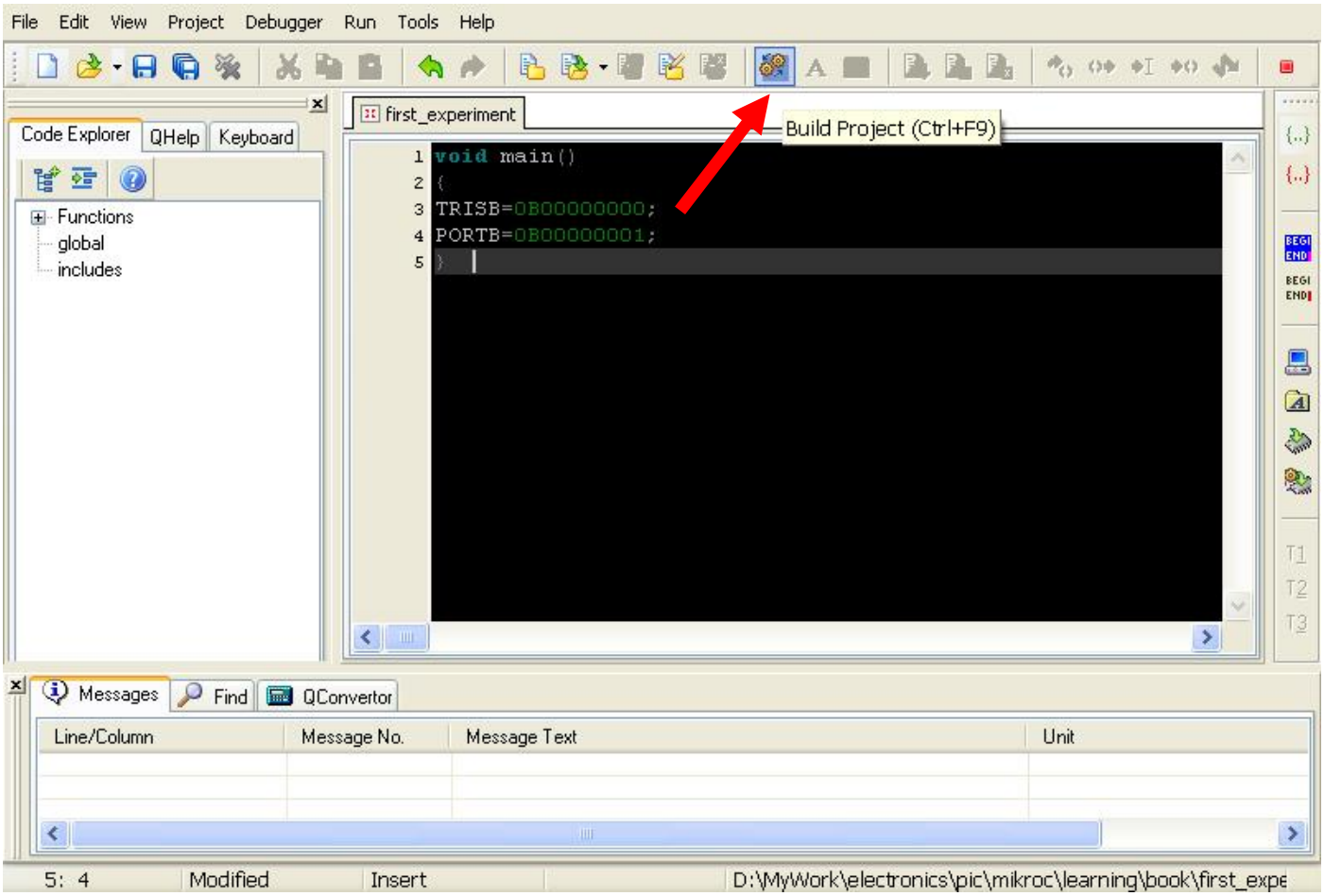
تلاحظ في المكان المخصص لكتابة البرنامج (ذو اللون الأسود) كتبنا فيه الأوامر أو الكود الذي سيقوم بتشغيل الليد الموصل على B0 إذا لم تكن الأوامر واضحة الرؤية انظر للشكل التالي .

```
void main()
{
TRISB=0B00000000;
PORTB=0B00000001;
 }
```

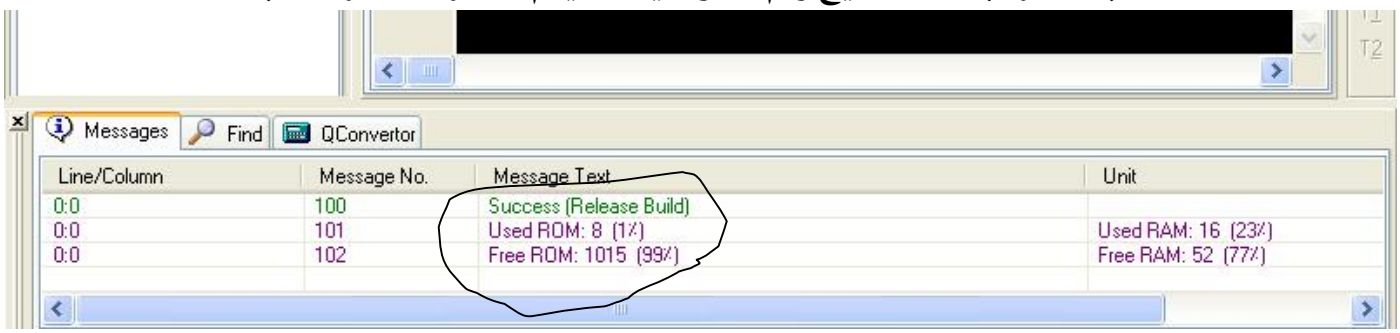
سنشرح هذه الأوامر بعد تنفيذ بقية خطوات التجربة .

ترجمة البرنامج ومراجعته من الأخطاء :-

الخطوة التالية هي ترجمة البرنامج حيث أن الأوامر التي كتبناها لا يستطيع أن يفهمها البك مباشرة ولكن يجب أن نترجمها للغة التي يفهمها وتسمى هذه العملية ببناء المشروع build project وذلك بالضغط بالماوس على  أو من لوحة المفاتيح على Ctrl+F9 كما بالشكل



الآن إذا كنت قد كتبت الكود بشكل صحيح ولم تنس شيئا فسيتم إظهار هذه الرسالة بالأسفل



أما إذا كتبت الكود بشكل خاطئ لن تظهر لك الرسالة السابقة (جرب حذف أي قوس ثم قم بعملية الترجمة ولاحظ ماذا سيظهر لك) ... عندها راجع ما كتبته بدقه واضغط على زر الترجمة مرة أخرى build project . وسنشرح الأخطاء في كتابة الكود بشكل مفصل فيما بعد بإذن الله .

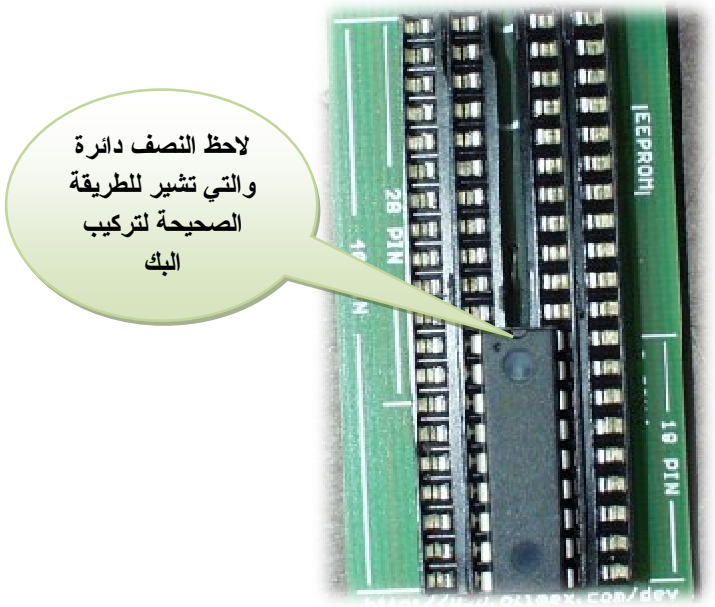
بعد عملية الترجمة السابقة ستكون لغة البرمجة MikroC قد وضعت الترجمة هذه في ملف امتداده hex أي أننا سنجده في نفس المكان الذي سيحفظ فيه المشروع الذي حددناه مسبقا .. سنجد ملف له نفس اسم المشروع بامتداد hex ... اي سنجد ملف اسمه first_experiment.hex .

اسم المشروع في

كتابة (حرق) البرنامج على البك :-

البك لن يعمل في الدائرة التي كوناها لأنه ليس عليه برنامج يحدد وظيفته لذلك سنكتب عليه البرنامج الذي كتبناه والذي تحول إلى ملف hex وسنتبع الخطوات التالية :

سنضع البك في جهاز البرمجة كما بالشكل

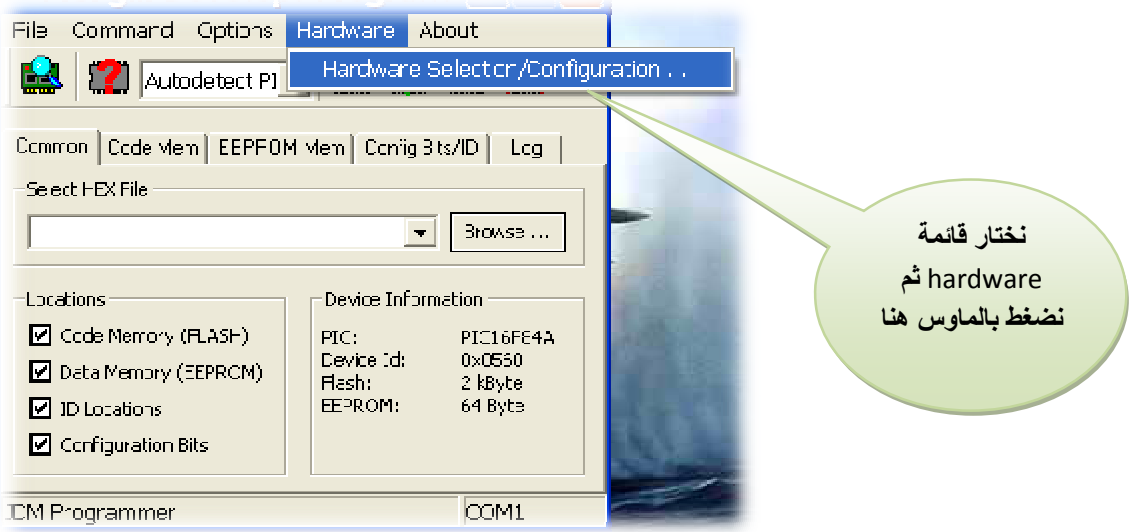


وإذا كنت تستخدم جهاز برمجة آخر اتبع تعليماته وضع البك بالطريقة الصحيحة في النوع الذي نستخدمه من أجهزة البرمجة في هذا الكتاب يجب وضع البك بهذا الشكل ويجب الانتباه إلى النصف دائرة التي ترشدنا للطريقة الصحيحة للتركيب .

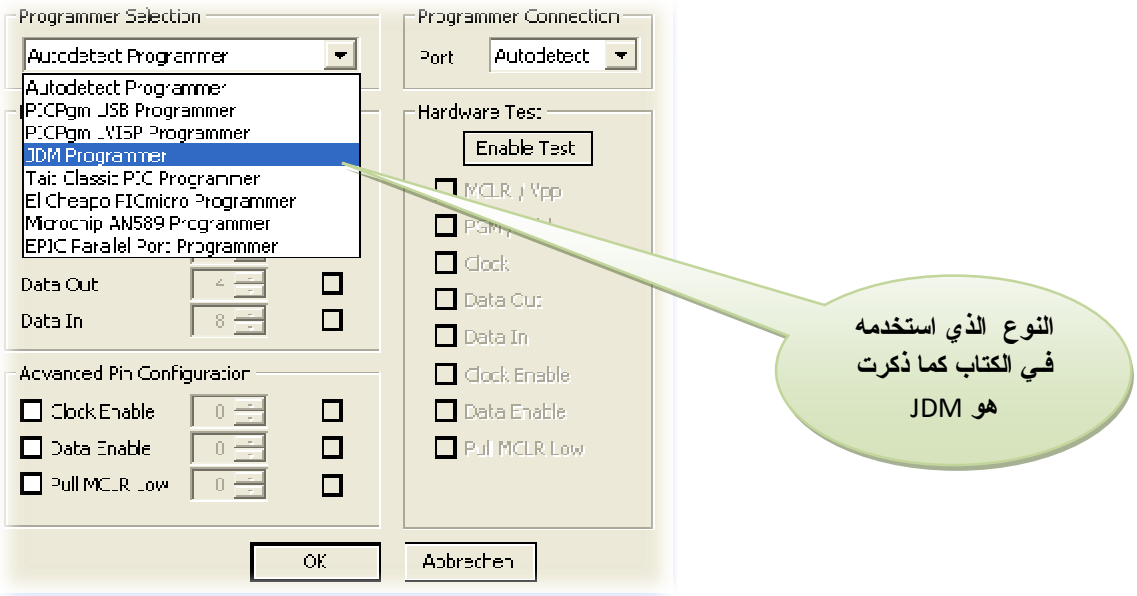
وطبعا توصيل جهاز البرمجة بالكمبيوتر عن طريق الكابل خطوة يجب أن لا تنسى .

الخطوة التالية هي تشغيل برنامج PICPgm Programmer إذا لم تكن تمتلكه ارجع للجزء الخاص بالبرامج والذي ستجده في نهاية الكتاب .

هذا البرنامج هو الذي سيكتب على البك . هذا البرنامج سيقوم بأخذ ملف الـ hex وسيقوم بكتابته على البك ، ولكن يجب ضبط إعدادات البرنامج أولا ... في حالتنا هذه جهاز البرمجة نوعه JDM لذلك سنضبط الاعدادات باتباع الخطوات التالية .



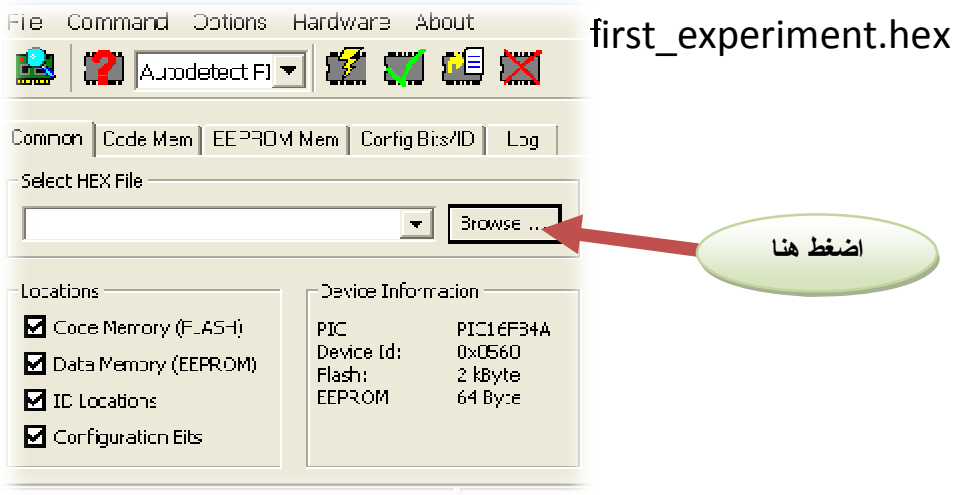
نختار نوع جهاز البرمجة



بعد ذلك اضغط على زر OK

ثم اضغط على زر Browse وحدد المكان الذي يوجد فيه المشروع ومن ثم حدد ملف الهيكس حيث سيكون

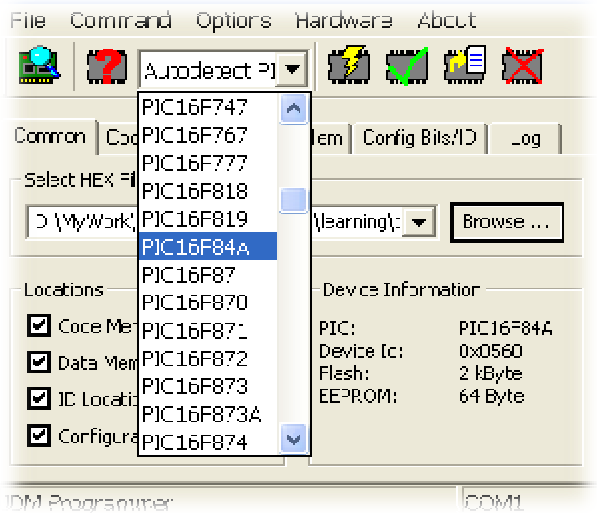
في تجربتنا هذه اسمه first_experiment.hex



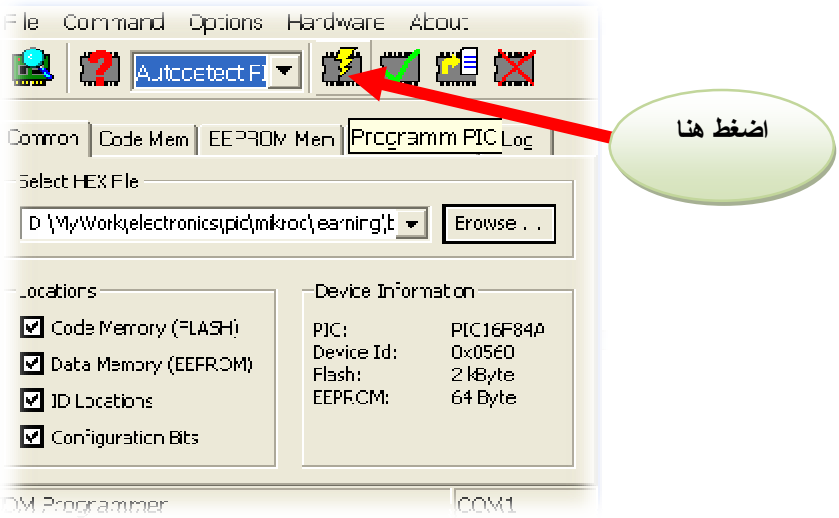
نختار الملف ثم نضغط على زر Open

ونختار نوع البك المستخدم وهو في هذه

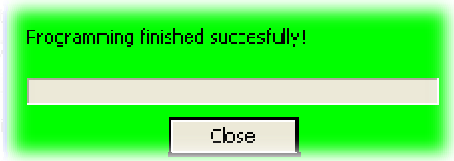
التجربة من النوع PIC16F84A



ثم نضغط على الزر الذي سيبدأ بعملية الكتابة وهو  كما هو واضح بالشكل



وانتظر إلى أن تنتهي عملية الكتابة .. حيث ستظهر هذه الشاشة بعد الانتهاء

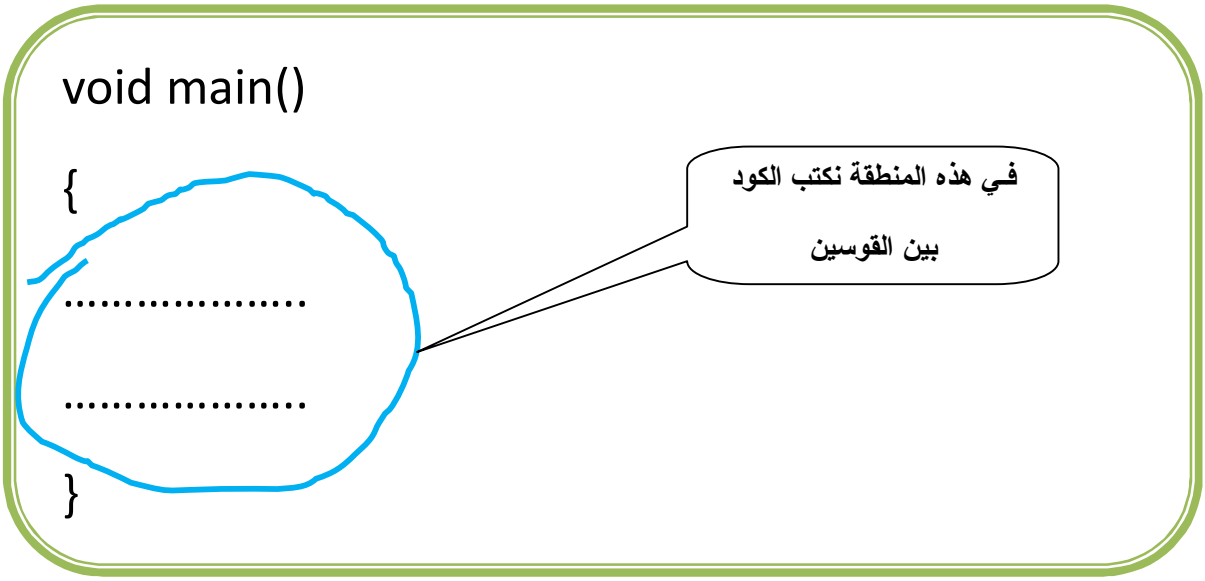


بهذا تكون انتهت عملية الكتابة على البك يمكنك الآن أن تقوم بفك البك من جهاز البرمجة بحذر وتركيبه في الدائرة الالكترونية الخاصة بالمشروع ومن الأفضل أن تفصل التيار الكهربي في الدائرة عند تركيب البك وبعد تركيب البك يمكنك توصيل التيار الكهربي .

آمل أن تكون قد نجحت في تنفيذ التجربة . إذا واجهتك أي مشاكل يمكنك الإستعانة بأهل الخبرة في المجال أو مراسلتي على إيميلي الخاص .

والآن سنشرح الكود الذي كتبناه الذي جعل البك يشغل الليد . وسنقوم بطرح تجارب أخرى مع تغيرات بسيطة في الأوامر لنفهم أكثر وأكثر ولنحترف هذا المجال بإذن الله تعالى .

في أي برنامج بلغة الـ MikroC سنضع الكود بالشكل التالي



ولكن ماذا تعني void main() ؟؟ إنها تعني أن هذه هي الدالة الرئيسية للبرنامج ... دالة ؟؟ نعم ، إن البرنامج أو الكود يمكن تقسيمه إلى دوال حيث أن كل دالة تحتوي على مجموعة من الأوامر . وكلمة void هي نوع هذه الدالة وتعني أنها لا ترجع قيمة وسنشرح الدوال بالتفصيل فيما بعد إن شاء الله... إذا لم تفهم الشرح السابق جيداً لا تقلق كل ما عليك أن تعرفه أنه في أي برنامج سنقوم بكتابته سنكتب في بدايته هذا السطر void main()

ولا ننسى الأقواس . فقط اعلم هذه المعلومة البسيطة . . . مؤقتاً .

اعلم عزيزي القارئ : أن البك لا يستطيع أن يخرج جهد على أحد أطرافه ... إلا في حالة أن أخبره أن هذا الطرف سيكون خرج .... كيف سأخبره ؟؟ .. عن طريق الأمر أو المسجل TRIS فمثلا لو أردت أن أجعل أي رجل (طرف) من PORTB تخرج جهد كهربي لابد من استخدام TRISB وإذا أردت أن أجعل رجل من PORTA تخرج جهد كهربي لابد من استخدام TRISA وهكذا ...

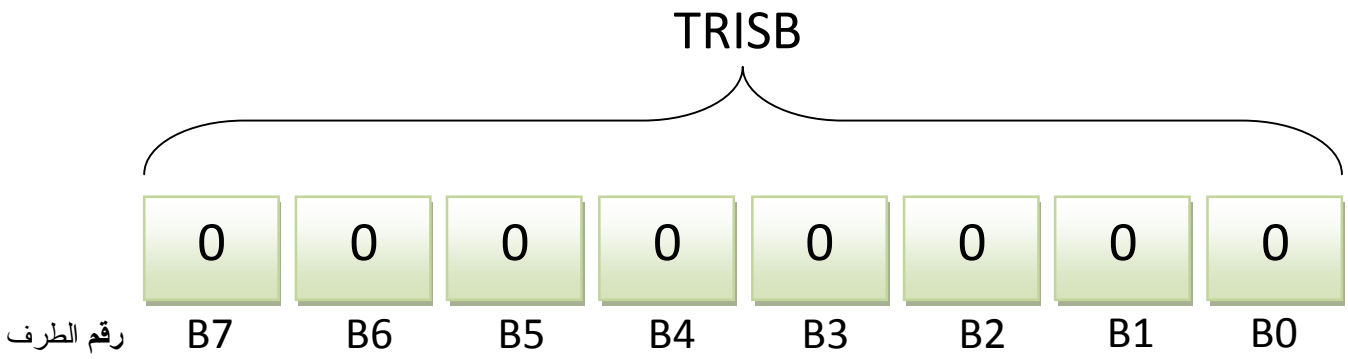
احفظ معي هذه العلاقة التي تخص TRIS والتي ستساعدنا على فهم الكود الذي كتبناه

0 - - - - - - - - - - تعني خرج OUTPUT

1 - - - - - - - - - - تعني دخل INPUT

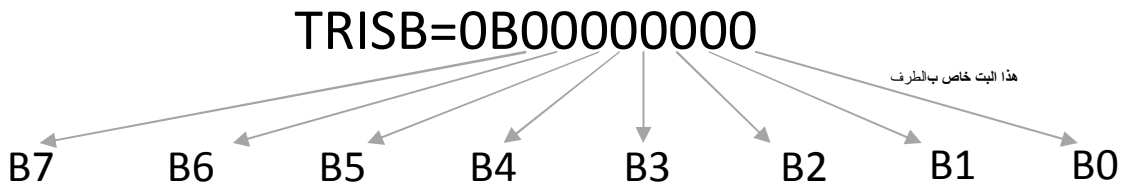
في مشروعنا كنا نريد تشغيل الليد الموجود في PORTB حيث سيكون الليد موصل بـ B0 لذلك سنستخدم المسجل أو الأمر TRISB حيث سنجعل الطرف رقم B0 خرج ليتسنى لنا بعد ذلك إخراج جهد كهربي منها.. إذا سنجعل الطرف رقم B0 بصفر لتصبح خرج ..... تذكر العلاقة السابقة.

ولكن كيف سأجعل B0 بصفر ؟؟؟ ... الأمر بسيط الآن المسجل TRISB يتكون من ٨ أجزاء أو يمكن أن نقول عنهم ٨ بت (BIT) كل بت خاص بطرف معين إذا كان هذا البت يساوي واحد سيكون الطرف الخاص به دخل وإذا كان البت يساوي صفر سيكون الطرف الخاصة به خرج . في الرسم التالي نرى مكونات المسجل TRISB يتكون من ثمان أجزاء كل جزء أو كل بت مرتبط برجل معينة يستطيع جعلها دخل أو خرج



في مشروعنا جعلناهم جميعا بأصفار أي أن الأطراف من B0 إلى B7 وظفناهم كخرج للمايكروكنترولر.

كيف وظفناهم كخرج؟؟ .. بجعل قيمة البت الخاص بكل طرف تساوي صفر من خلال كتابتنا للأمر TRISB=0B00000000; انظر للشكل التالي ليتضح الأمر أكثر



لعلك لاحظت 0B بعد علامة = وهي تعني أن الرقم الذي سيكتب بعدها سيكون بالنظام الثنائي binary .

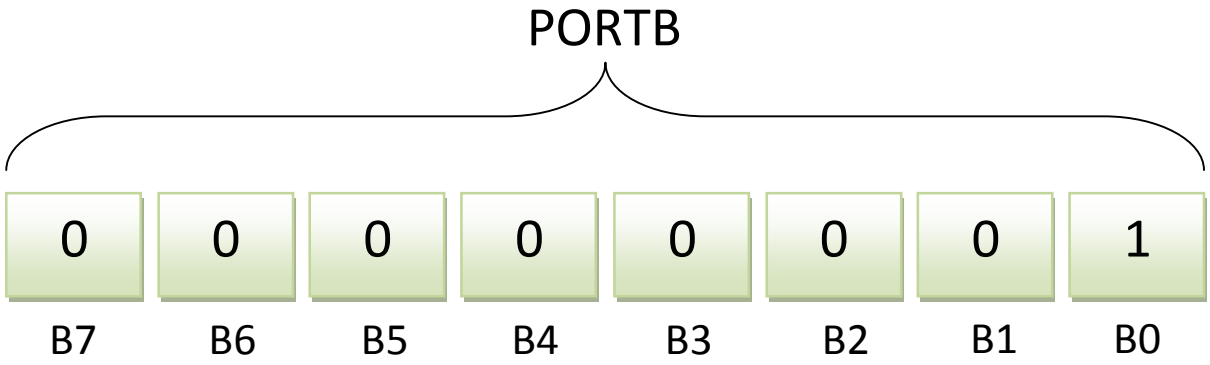
بعد أن وظفنا الطرف رقم B0 كخرج نستطيع الآن أن نجعل البك يخرج جهد كهربي عليها من خلال استخدامنا لأمر آخر (مسجل آخر) اسمه PORT ..

ولكي تفهمه جيداً احفظ معي العلاقة التالية الخاصة بـPORT :

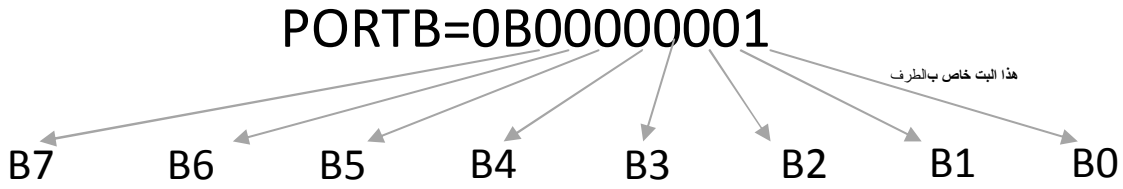
1 - تعني جهد موجب خمسة فولت (HI) 0 - تعني جهد صفر فولت (سالب البطارية) LOW .

لكي يعمل الليد سنخرج على الطرف B0 جهد موجب HI وذلك باستخدام الأمر أو المسجل PORTB حيث سنجعل B0 بواحد .... واحد تجعله يخرج جهد موجب خمسة فولت .... تذكر العلاقة السابقة.

PORTB مثل TRISB ينقسم إلى ثماني أجزاء (٨ بت) كل بت خاص بطرف معين .. نستطيع من خلال أي بت أن نخرج جهد موجب أو صفر على أي طرف من أطراف البك من B0 إلى B7 ... وذلك بأن نجعل قيمة البت =١ لكي يخرج جهد موجب خمسة أو نجعل قيمة البت = ٠ لكي يخرج صفر فولت .



في تجربتنا جعلنا أول بت في PORTB بواحد لكي يخرج جهد كهربي على هذه الطرف ليضيء الليد أما باقي الأطراف فلم نستخدمها (لم نوصل بها ليدات أو عناصر أخرى) لذلك جعلناهم جميعا بأصفار . حيث كتبنا الأمر التالي : PORTB=0B00000001;



ملحوظة في غاية الأهمية : لا يكفي أن أكتب PORTB=0B00000001 بل يجب أن أضع في نهاية السطر (نهاية الأمر) علامة ; كالتالي PORTB=0B00000001; وكذلك الحال بالنسبة للأمر TRIS فنكتب TRISB=0B00000000; ..... هذه العلامة وغيرها من العلامات مثل { } تسمى هذه الأشياء بـ syntax اللغة .. أي الطريقة التي يجب أن يكتب بها الكود وإذا أخطأنا في الكتابة لن تتم عملية الترجمة ولن نحصل على ملف الهيكس hex .

والآن لنقوم بمراجعة سريعة على الكود الذي كتبناه .

```
void main()
{
TRISB=0B00000000;
PORTB=0B00000001;
 }
```

السطر الأول void main() نكتبه في بداية أي برنامج ثم نفتح قوس ونبدأ بكتابة الكود

السطر الثالث سيوظف جميع الأطراف من B0 إلى B7 على أنها خرج .من خلال TRISB حيث صفر تعني خرج وواحد تعني دخل (دخل هذه نستخدمها مثلاً عندما نوصل مفتاح switch بطرف الميكروكنترولر).

السطر الرابع سيجعل الطرف رقم B0 يخرج أو يوصل جهد موجب خمسة فولت . وذلك من خلال الأمر أو المسجل PORT حيث صفر تعني وصل بسالب البطارية و واحد تعني وصل بموجب خمسة فولت .

السطر الخامس هو أن نغلق القوس . ولا ننسى علامة ;

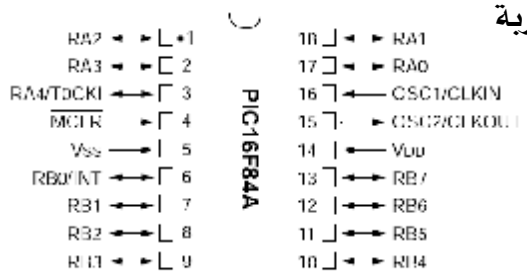
كيف سيتم تنفيذ البرنامج (الكود) ؟

تقوم لغة البرمجة MikroC بترجمة الكود الذي كتبناه وعندما ينفذه البك سينفذه سطر سطر... أي ينفذ الأمر الذي في السطر الأول ثم الثاني ثم الثالث ... وهكذا ..

بعد أن انتهينا من التجربة الأولى سنقوم بعمل تجربة أخرى تجعلنا نتأكد من قوة فهمنا للتجربة الأولى وبالطبع سنشعر أن الأمر أصبح أكثر سهولة ومتعة ....

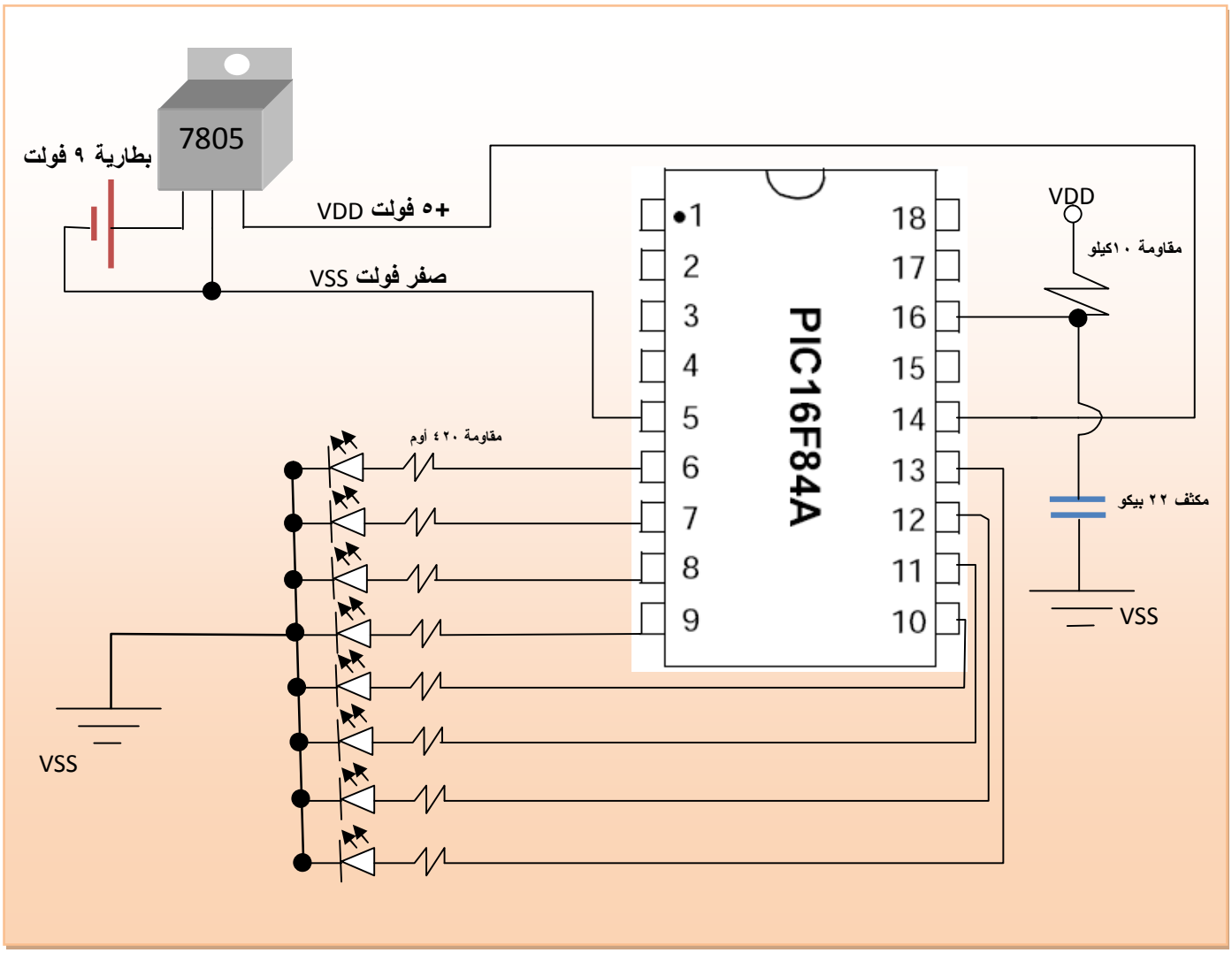
التجربة (٢)

الهدف من التجربة هو تشغيل الليدات الموصلة بالطرف رقم B0 و B1 و B5 و B7 أما بقية الليدات فسنجعلها لا تعمل . سنتعلم أشياء جديدة وممتعة في هذه التجربة



توصيل الدائرة :-

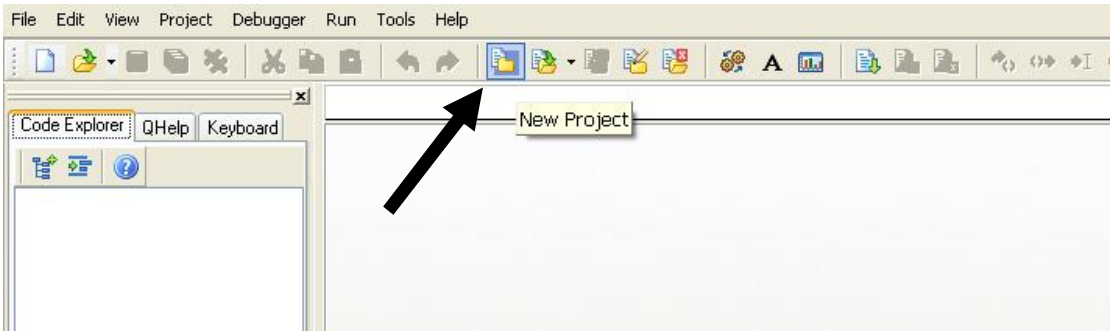
الخطوة الأولى في التجربة هي توصيل الدائرة التالية:



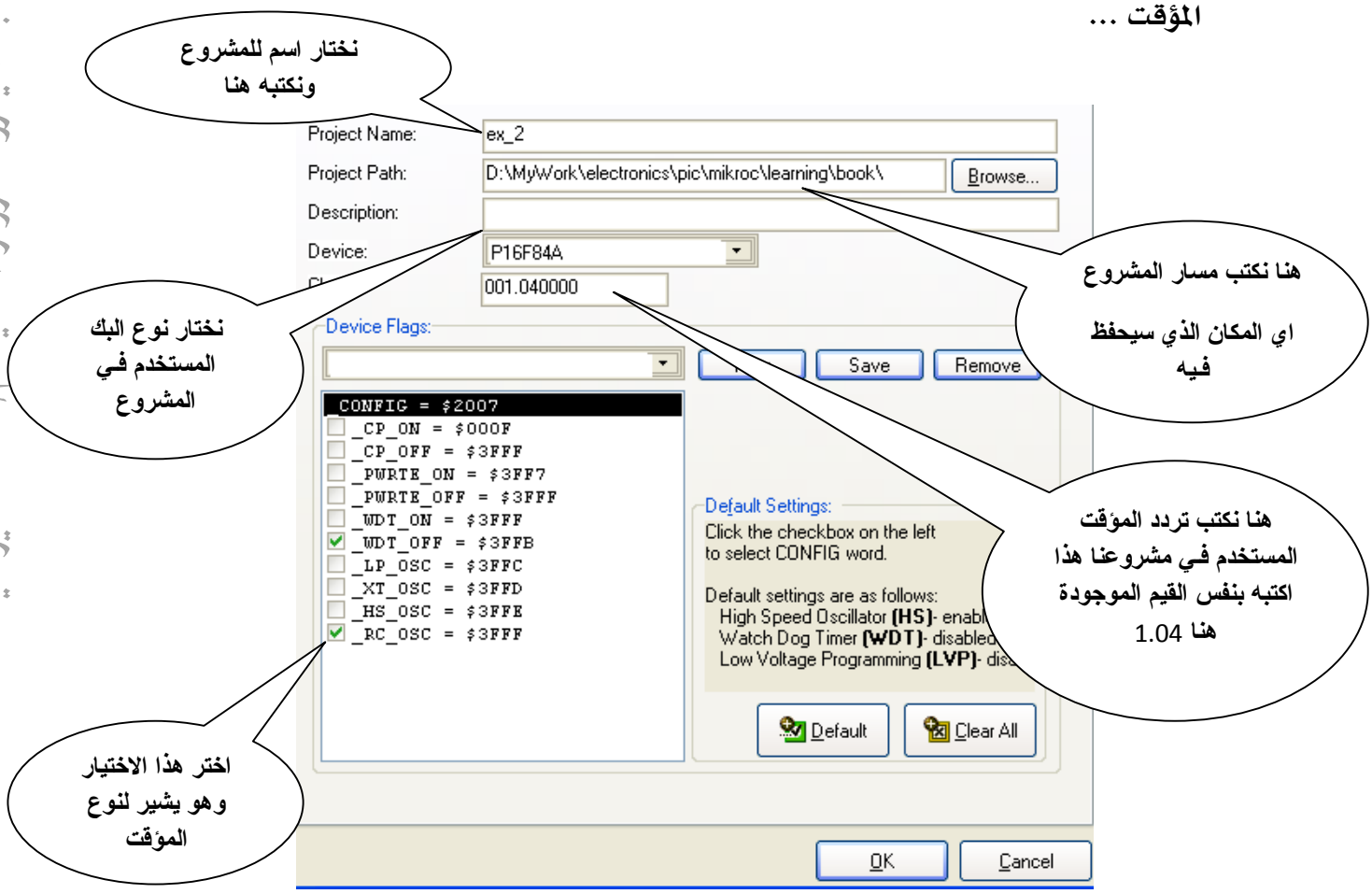
كتابة البرنامج :

سنقوم بإنشاء مشروع جديد كما فعلنا بالتجربة الأولى تماما ولكن سنكتب اسم آخر للمشروع

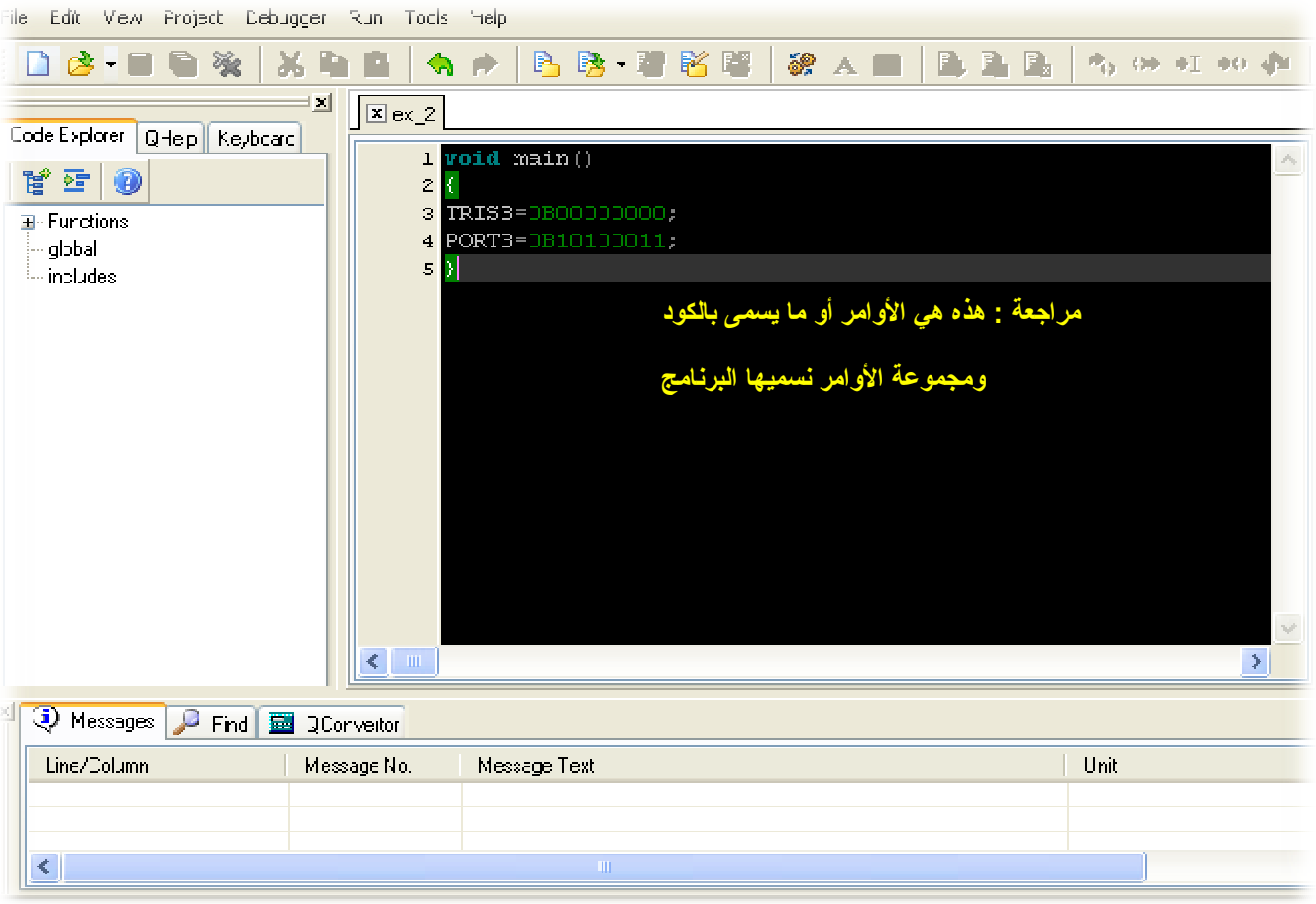
١- نقوم بفتح البرنامج ثم إنشاء مشروع جديد بالضغط بالماوس على  كما بالشكل .



٢- نقوم بكتابة اسم المشروع ونحدد المكان الذي سيحفظ فيه ونحدد نوع البك المستخدم وكذلك المؤقت ...



٣- الآن ستظهر لنا شاشة سنكتب فيها البرنامج كما بالشكل



إليك الكود بشكل أوضح

```
void main()
{
TRISB=0B00000000;
PORTB=0B10100011;
 }
```

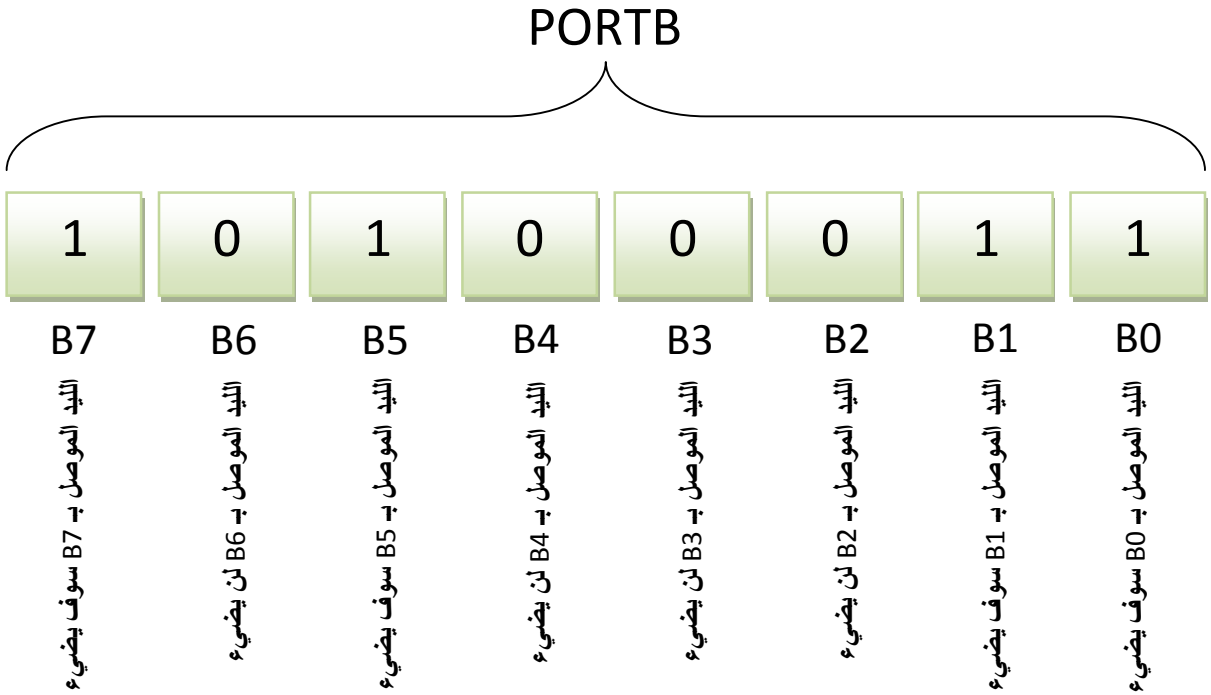
ثم نقوم بالترجمة بالضغط على أو Ctrl+F9 ونراجع الأخطاء . ثم نتبع نفس الخطوات التي اتبعناها بعد ذلك في التجربة الأولى وعند تطبيق الدائرة عملياً سنجد أن الليدات المطلوبة تضيء .

تحليل الكود الذي كتبناه :

طبعا السطر الأول متفقين على وضعه في بداية أي برنامج وكذلك الأقواس

السطر ;TRISB=0B00000000 سيوظف الأطراف من B0 إلى B7 على أنها خرج وطبعا هذا يعتبر إذن أو تصريح للميكروكنترولر يمكنه فيما بعد من تطبيق(أو إخراج) جهد موجب أو سالب على هذه الأطراف .

السطر ;PORTB=0B10100011 سيجعل الأطراف B0 و B1 و B5 و B7 توصّل جهد ٥ فولت وبالتالي ستعمل (ستضيء) الليدات الموصلة بها . أما الأطراف B2 و B3 و B4 و B6 ستوصل جهد صفر فولت (سالب البطارية) وبالتالي لن تعمل الليدات الموصلة بها .



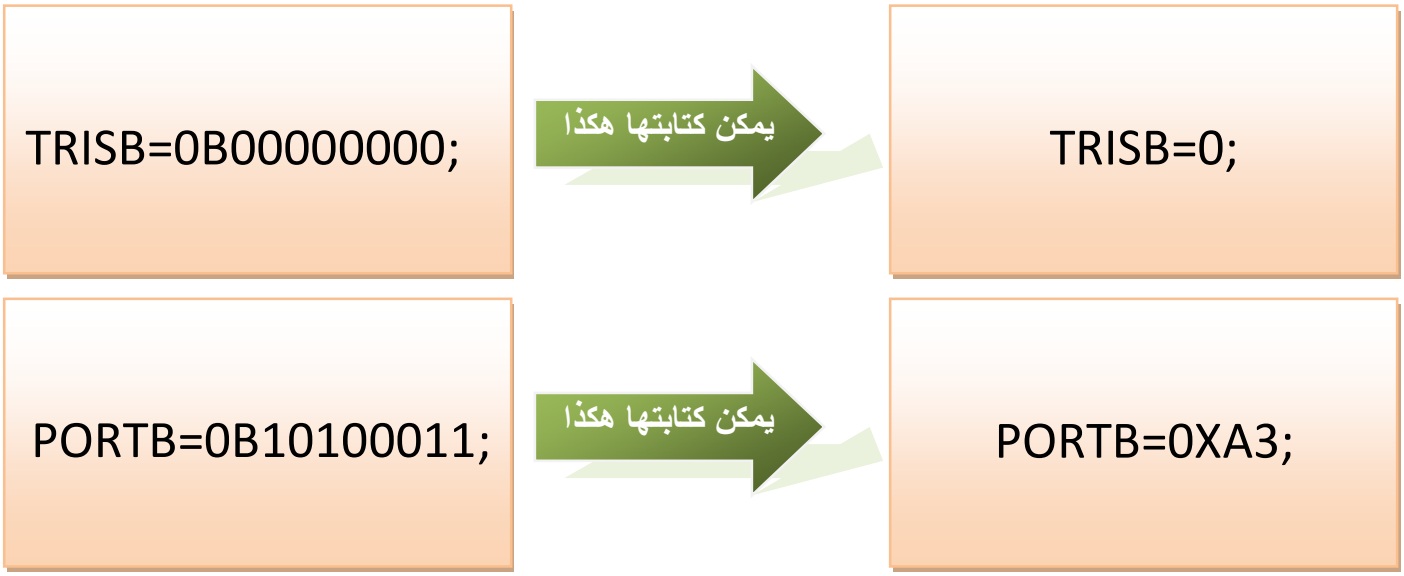
معلومات جديدة في هذه التجربة :

بالنسبة للسطر TRISB=0B00000000; بما أن كل البتات (Bits) قيمتها بأصفار نستطيع أن نكتب هذا السطر البرمجي بطريقة أخرى كالتالي TRISB=0; وسيؤدي نفس الوظيفة .

بالنسبة للسطر PORTB=0B10100011; نستطيع كتابته بشكل آخر كالتالي :

PORTB=0XA3; حيث أن 0x تعني أن الرقم الذي بعدها سيكون بنظام السادس عشري ومن المعلوم أن a3 بالسادس عشري تساوي 10100011 بنظام الثنائي البايناري binary .

خلاصة ماسبق يوضحه الشكل التالي :-



وبالتالي يمكن أن يكون الشكل النهائي للكود كما يلي :-

```
void main()
{
TRISB=0;
PORTB=0XA3;
 }
```

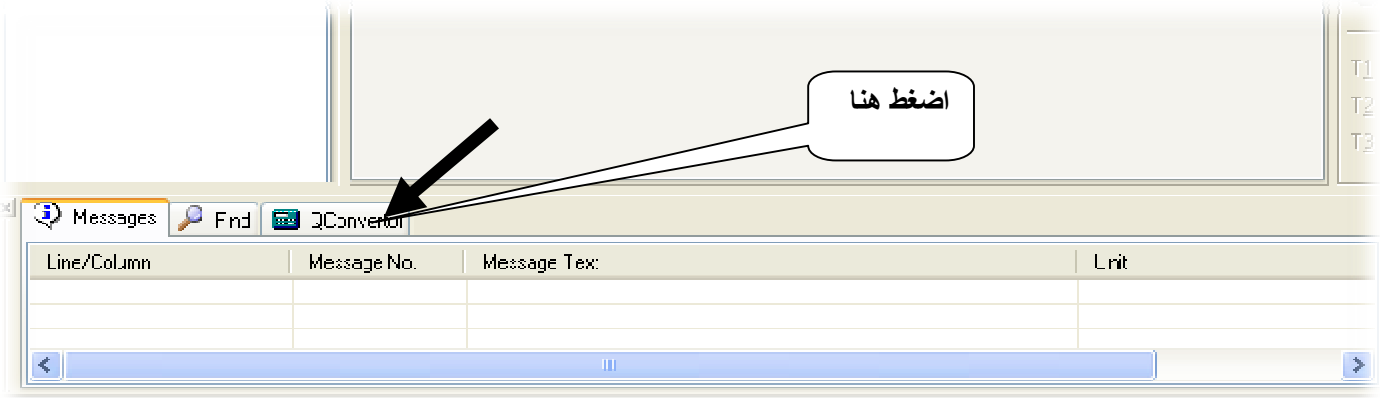
إذن لو كتبنا بعد علامة تساوي 0b فهذا يعني أن الرقم الذي بعدها سيكون بالنظام الثنائي ، وإذا استخدمنا 0x بدلا من 0b فهذا يعني أن الرقم سيكون بالسادس عشري أما إذا لم نضع أي شيء بعد علامة تساوي وكتبنا الرقم مباشرة فهذا يعني أن الرقم سيكون بالنظام العشري .

إمكانية جميلة في لغة Mikroc وهي التحويل من نظام رقمي لآخر ، حيث يمكنك التحويل من أي من النظم التالية إلى بعضها البعض : النظام العشري decimal النظام الثنائي binary النظام السادس عشري hexadecimal وذلك بكل سهولة ويسر

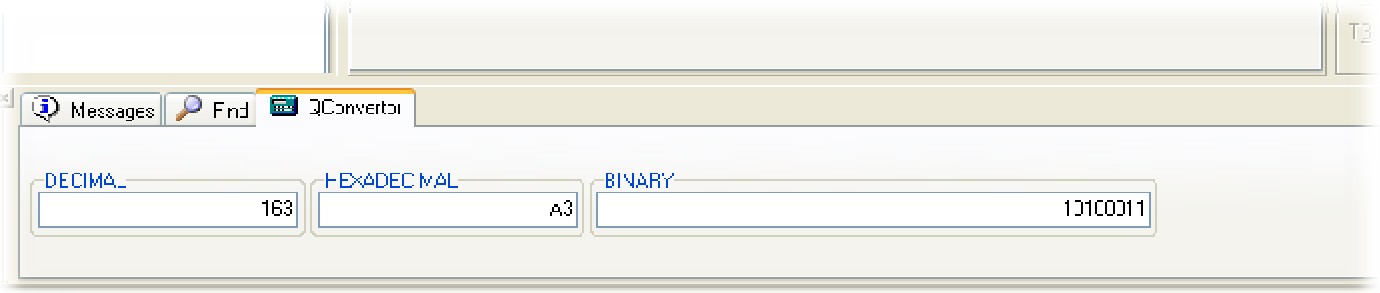
مثال على التحويل من ثنائي (باياري) إلى سادس عشري ... حوّل الرقم التالي إلى سادس عشري

10100011

في أسفل البرنامج اضغط على QConvertor كما بالشكل



بمجرد أن تكتب في خانة binary سيتم التحويل تلقائيا إلى النظامين الآخرين



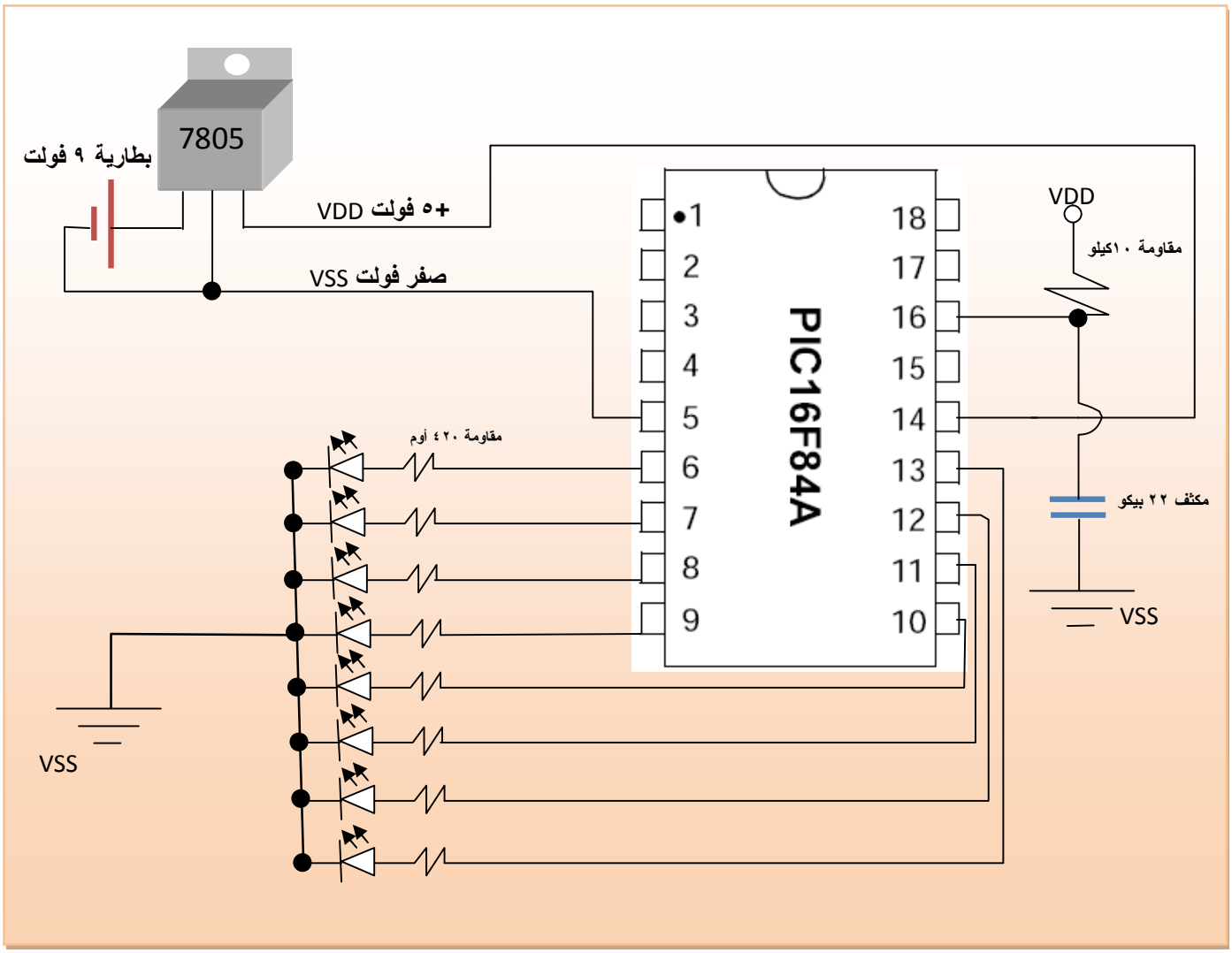
وهكذا عندما تريد التحويل من أي نظام من هؤلاء إلى الآخر فقط اكتب في الخانة المخصصة فمثلا إذا أردت التحويل من عشري إلى ثنائي فقط اكتب في خانة العشري .. وسيتم التحويل مباشرة للنظام الثنائي والسادس عشري .

التجربة (٣)

في هذه التجربة الأمر سيصبح أكثر إثارة ومتعة ، فالهدف من هذه التجربة هو إضاءة الليدات الموصلة بالأطراف من B0 إلى B7 ثم إطفاءهم ثم تكرار هذه العملية بإستمرار .

توصيل الدائرة :-

الخطوة الأولى في التجربة هي توصيل الدائرة وهي نفس الدائرة التي في التجربة السابقة :



لعلك استنتجت من التجربتين السابقتين أن الفرق بين أي تجربة وأخرى هو توصيل الدائرة والكود أو البرنامج لذلك في هذه التجربة سنكتب البرنامج مباشرة والخطوات الأخرى معروفة ..

كتابة البرنامج :-

من المعلوم لديك أننا في هذه التجربة سنجعل جميع الأطراف التي في PORTB والتي هي من B0 إلى B7 كخرج للمايكروكنترولر عن طريق استخدام TRISB وبما أنهم كلهم خرج إذن سنكتب السطر التالي :

```
TRISB=0;
```

وبما أننا سنضيء جميع الليدات ثم نغلقها إذن سنكتب الأمرين التاليين : الأمر الأول الذي سيخرج جهد خمسة فولت على الأرجل كلها ;PORTB=0B11111111 أو ;PORTB=0XFF والأمر الثاني هو الذي سيوصل سالب البطارية (أي سيطفئ الليدات) ;PORTB=0 ...

ولكن إذا كتبنا الأمرين وراء بعض مباشرة سيتم تنفيذ الأمرين بسرعة كبيرة لدرجة أننا لن نستطيع أن نرى الليدات وهي تعمل وسنجدها مطفئة دائما مع أننا كتبنا الأمر الذي يشغلها . من المعلوم أن البك سينفذ الأوامر التي كتبناها أمر وراء أمر وهكذا .. ولكن هل تعلم أن مدة تنفيذه للأمر صغيرة جداً حيث ستكون أقل من ملي ثانية (أي جزء من ألف جزء من الثانية) على حسب تردد المذبب الذي نستخدمه . لذلك عندما ينفذ الأمر الذي سيجعل الليدات تضيء . سينفذ بعده مباشرة الأمر الذي يليه وهو إطفاء جميع الليدات كل هذا في منتهى السرعة فبذلك لن ترى الليدات تضيء ... انظر للبرنامج وتخيل الأمر

```
Void main()
{
TRISB=0;
PORTB=0XFF;
PORTB=0;
}
```

البك سينفذ الأمر ثم الذي يليه ثم الذي يليه وهكذا في منتهى السرعة ..

ولكن ماذا سيحدث إذا غيرنا ترتيب البرنامج أي جعلنا الأمر ;PORTB=0 قبل ;PORTB=0XFF ما الذي سيحدث يا ترى ؟؟ .. بما أن الأمر الذي يطفئ جميع الليدات هو الأول إذن سينفذ البك هذا الأمر أولاً

وبمنتهى السرعة سينفذ الأمر الذي يليه وهو تشغيلهم أي أننا لن نرى الليدات قد أطفئت أساساً بل بمجرد توصيلنا للدائرة سنجد الليدات كلها مضيئة ..!!!

إذن ما الحل ؟؟ ... الحل في هذه المشكلة هو أن نعطي للبك أمر يجعله ينتظر قليلاً فنجعل أمر الإنتظار هذا بين الأمرين السابقين (أمر تشغيل الليدات و أمر إطفاء الليدات) .. فمثلا إذا أردنا أن ينتظر البك زمن مقداره نصف ثانية تقريبا أي (٥٠٠ ملي ثانية) سنكتب الأمر التالي :

Delay_ms(500);

وإذا أردنا جعل زمن الإنتظار يساوي ربع ثانية (٢٥٠ ملي ثانية) سنكتب الأمر التالي :

Delay_ms(250);

أي أننا نكتب بين القوسين السابقين مدة الإنتظار بالملي ثانية . إذا سنعدل البرنامج ويصبح بالشكل التالي

```
Void main()
{
TRISB=0;
PORTB=0xFF;
delay_ms(1000);
PORTB=0;
}
```

ملحوظة : لا تكتب الأمر ;(delay_ms(500 بحروف كبيرة(كابتال) .

الآن لنتابع ماذا سيحدث في هذا الكود .. سيقوم البك بتوظيف الرجول (الأطراف) المرادة على أنها خرج ثم يقوم بتشغيل الليدات ثم ينتظر ثانية ثم يطفئ الليدات .... جميل لقد تم حل مشلكة السرعة .. ولكن انتظر هناك مشكلة أخرى وهي أن البك سيضي الليدات ثم ينتظر ثم يطفئها مرة واحدة فقط ولن يكرر هذه العملية بإستمرار ... إذن ما هو الحل ؟؟ ... الحل هو جعل البك ينفذ هذه الأوامر وبعد الإنتهاء من تنفيذها ينفذها مرة أخرى وهكذا .. كيف سنكتب الكود ؟؟؟ تابع معي ..

إذا أردت أن تكرر مجموعة من الأوامر بإستمرار كل ما عليك فعله هو جعل هذه الأوامر بين أمرين كما بالشكل التالي :

```
LOOP:
.....
.....
.....
goto LOOP;
```

في هذه المنطقة نكتب الكود الذي نريده أن يتكرر بإستمرار .

معنى هذ الأمر اذهب إلى العنوان loop وابدأ تنفيذ الأوامر التي بعده

إذن سيكون شكل البرنامج النهائي كالتالي

```
void main()
{
TRISB=0;
Loop:
PORTB=0XFF;
delay_ms(1000);
PORTB=0;
Delay_ms(1000);
goto loop;
}
```

هذا الأمر سيوظف جميل الرجول في PortB كخرج

هذا هو العنوان الذي سيرجع إليه البك مرة أخرى

هذا الأمر سيجعل جميع الليدات تضيء

هذا الأمر سيجعل البك ينتظر لمدة ثانية ثم ينفذ الأمر الذي يليه

هذا الأمر سيجعل جميع الليدات تطفئ

هذا الأمر سيجعل البك ينتظر لمدة ثانية ثم ينفذ الأمر الذي يليه

هذا الأمر سيجعل البك يذهب لينفذ الأوامر التي بعد العنوان الذي اسمه loop

وهكذا الكود السابق يكون هو الصورة النهائية للبرنامج الذي سيشغل جميع الليدات ثم ينتظر ثم يطفئها ثم ينتظر ويكرر هذه العملية بإستمرار .....

ملحوظة مهمة : لعلك لاحظت أنني كتبت بعد السطر البرمجي PORTB=0; كتبت بعده أمر للإنتظار لأنه في حالة عدم كتابته سيقوم البك بتشغيل الليدات ثم ينتظر ثم يطفيهم ثم يضيئهم من جديد بسرعة دون أن نلاحظ عملية الإطفاء ... أي أننا كل ما سنلاحظه هو أن الليدات جميعا مضيئة ولا تنطفئ.

ملحوظة أخرى : يمكنك جعل العنوان بأي إسم آخر ... فمثلا نسميه Ahmad وبالتالي السطر الذي تكون جملة التكرار بالشكل التالي .

```
ahmad:
.....
.....
.....
goto ahmad;
```

في هذه المنطقة نكتب الكود الذي نريده أن يتكرر بإستمرار .

قبل الإنتقال إلى التجربة الرابعة لابد من التلميح إلى بعض خصائص Syntax اللغة (طريقة كتابة الكود في اللغة) .

بالنسبة للمفتاح Enter في لوحة المفاتيح يمكنك الإستغناء عنه في كثير من الحالات فمثلا بعد كتابة القوس الذي في بداية البرنامج يمكنك كتابة البرنامج مباشرة كما بالشكل التالي

```
void main()
{ TRISB=0;
PORTB=0XFF;}
```

هل لاحظت الأقواس ؟؟ .... جيد .. كما يمكنك أن تضم الأوامر لبعضها أي بعد علامة ; يمكنك أن لا تضغط على Enter كما يلي :

```
void main()
{ TRISB=0;PORTB=0XFF;}
```

أو أن تجعله في سطر واحد كما يلي كما يلي

```
void main() { TRISB=0;PORTB=0XFF;}
```

ويمكنك تطبيق هذه الخصائص في أي برنامج .

فلو أردنا تطبيقها في برنامج التجربة الثالثة يمكننا جعل البرنامج بالشكل التالي .

```
void main()
{ TRISB=0; Loop:  PORTB=0XFF;delay_ms(1000);PORTB=0;
Delay_ms(1000);Goto loop;}
```

ويمكن جعلهم في سطر واحد كما يلي :

```
void main(){TRISB=0; Loop:  PORTB=0XFF;delay_ms(1000);PORTB=0;Delay_ms(1000);Goto loop;}
```

شيء جميل برنامج سطر واحد ...

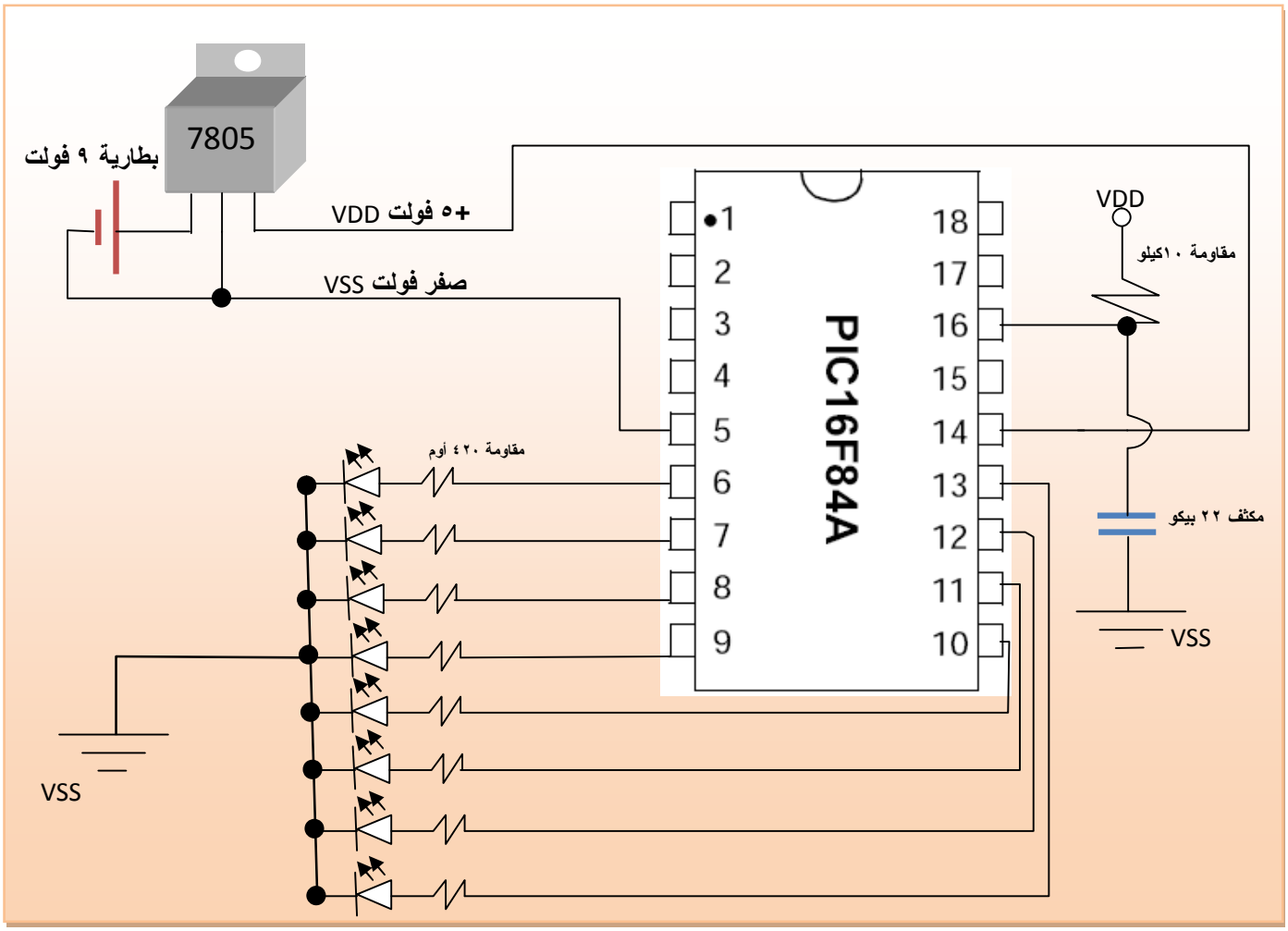
ولكن احذر ..!! لا تحاول الإكثار من هذه الطريقة فيجب عليك أن تحافظ على تنسيق الكود لكي يتسنى لك وللآخرين فهمه بسهولة عند قراءته ..

التجربة (٤)

في هذه التجربة سنتعلم مزيد من قوة التحكم فالهدف من هذه التجربة هو إضاءة الليدات الموصلة بالأطراف من B0 إلى B7 وإطفاءها ١٨ مرة فقط .. أو أي عدد آخر نريده

توصيل الدائرة :-

الخطوة الأولى في التجربة هي توصيل الدائرة وهي نفس الدائرة التي في التجربة السابقة :



كتابة البرنامج : من المعلوم لديك أننا في هذه التجربة سنجعل الأطراف من B0 إلى B7 على أنها خرج للدائرة عن طريق استخدم TRISB فنكتب الأمر ;TRISB=0ثم نضيء الليدات عن طريق استخدام PORTB من خلال جعل كل بت يحتويه بواحد فنكتب ;PORTB=0B11111111 أو PORTB=0XFF; ثم نجعل

البك ينتظر مدة معينة ولتكن ثانية فنكتب ;(delay_ms(1000 ثم نطفئ الليدات باستخدام ;PORTB=0 . ونجعل البك ينتظر مرة أخرى لمدة ثانية . المشكلة كيف سنجعل إضاءة الليدات وإطفاءها تتكرر ١٨ مرة ...

يمكننا أن نكرر كتابة الأوامر ١٨ مرة لتأدية الغرض (لكن هذه لا تعتبر برمجة ويعتبر حل غبي نوعا ما) وستكون المشكلة أكبر لو كان العدد ١٠٠ مرة مثلا أو أكثر

عزيزي القارئ : الأمر بسيط .. توجد في لغة البرمجة أوامر تمكننا من عمل تكرار بالعدد الذي نريده ومن هذه الأوامر أمر for وأحيانا تسمى جملة for . وإليك الطريقة التي ستكتب بها الكود

```
For(x=0;x<18;x++)
{
.....
.....
}
```

هنا نكتب الأوامر التي نريد تكراراها ١٨ مرة بين هذين القوسين

ولكن لا ننسى في بداية البرنامج أن نعرف المتغير x بأن نكتب السطر التالي في بداية البرنامج ;int x

أو ;char x وسنشرح ماذا تعني int وكذلك char فيما بعد . كل ما عليك أن تعرفه الآن هو أنه يجب عليك إضافة ذلك السطر في بداية البرنامج .. وبالتالي يكون الشكل النهائي للبرنامج المطلوب في هذه التجربة هو

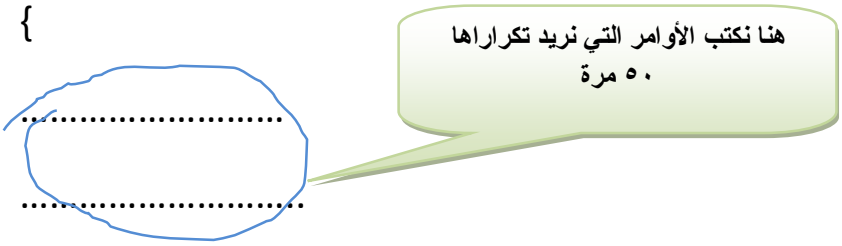
```
void main(){int x; TRISB=0;
for(x=0;x<18;x++)
{ PORTB=0XFF; delay_ms(1000); PORTB=0; ; delay_ms(1000);}
}
```

والآن ركز معي لنحاول فهم كيف تعمل جملة for ؟

أولا : يمكنك أن تستخدم جملة for لتكرار مجموعة من الأوامر عدد من المرات (هذا العدد تختاره أنت)

فعلى سبيل المثال لو أردنا أن نكرر مجموعة من الأوامر ٥٠ مرة سنكتب

```
for(x=0;x<50;x++)
{
.....
.....
}
```



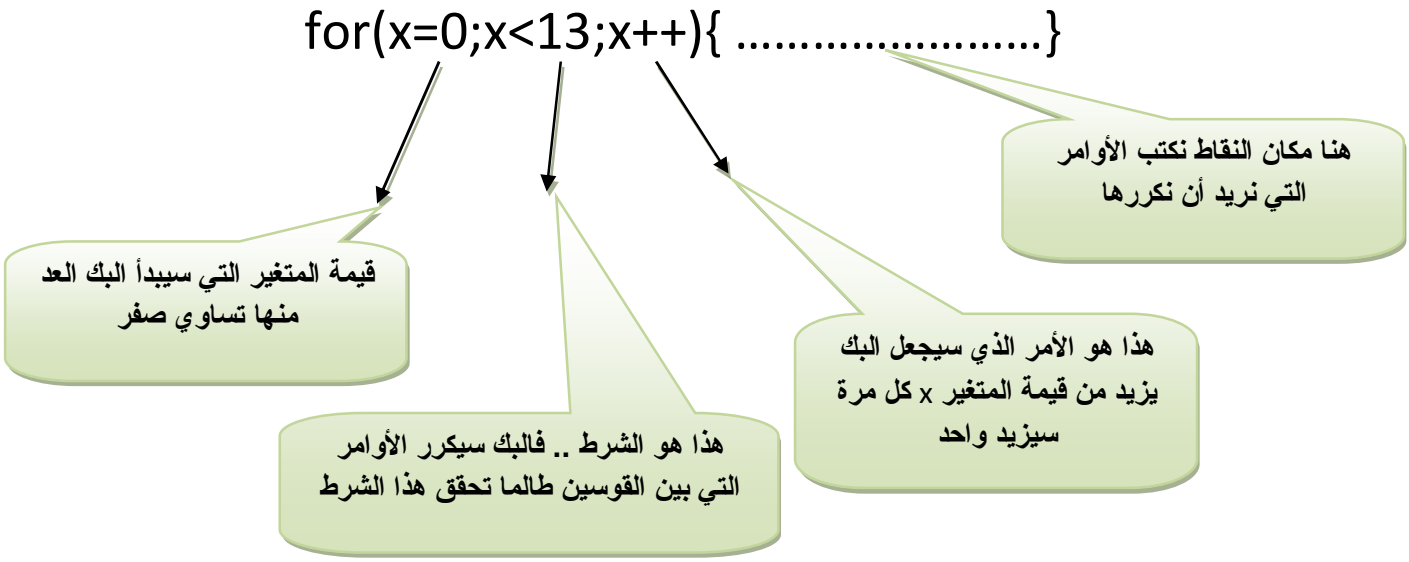
هل لاحظت أننا غيرنا الرقم ١٨ وجعلناه ٥٠ .. هذه هي الطريقة .. فإذا أردت أن تكرر مجموعة من الأوامر برقم معين تضع هذا الرقم بدلا من ١٨ ... كما يمكنك عزيزي القارئ أن تغير اسم المتغير فبدلا من أن يكون X يمكنك أن تجعله بأي حرف آخر أو أي اسم آخر ولكن لا تنسى أن تغير اسمه أيضا في بداية البرنامج فمثلا يمكن أنه نسميه moh فيكون الكود بالشكل التالي

```
void main(){char moh;
for(moh=0;moh<50;moh++) { PORTB=0XFF; delay_ms(1000);PORTB=0;
delay_ms(1000);}}
```

ثانيا : إليك عزيزي القارئ الطريقة التي تعمل بها جملة for وسنطبقها على الكود السابق .. كما أخبرتك فإن الأوامر التي بين القوسين الخاصين بجملة for سيتم تكرارهم ٥٠ مرة .. ولكن كيف سيحدث ذلك؟؟ ... عندما يبدأ البك في تنفيذ جملة for ستكون قيمة المتغير moh تساوي صفر وسيبدأ بتنفيذ الأوامر التي بين القوسين وبعد تنفيذها سيجعل قيمة moh تساوي واحد ثم ينفذ الأوامر التي بين القوسين مرة أخرى وبعد تنفيذها سيجعل moh تساوي ٢ ثم ينفذ الأوامر التي بين القوسين ... وهكذا يكرر هذه العملية طالما تحقق الشرط أي طالما كانت moh أصغر من خمسين . وبما أنه بدأ العد من صفر إذاً سيتوقف عن عملية العد بعد أن تكون moh تساوي ٤٩ لأنه لن يجعل moh تساوي خمسين لأن الشرط يشترط أن تكون قيمة moh أصغر من خمسين ... لاحظ معي لقد بدأ العد من صفر حتى وصل إلى ٤٩

لذلك ستكون عدد مرات التكرار (عدد مرات تكرار تنفيذ الأوامر) تساوي خمسين لأنه حسب الصفر كمرة.

وإليك الشكل التوضيحي التالي كمثال آخر



يجب عليك أن تعلم أن جملة for من أساسيات لغة البرمجة فهي في غاية الأهمية .. أرجو أن تكون فهمتها جيداً ... والآن أجب عن الأسئلة التالية التي تختبر مدى فهمك لجملة for :

في البرامج التالية كم مرة سيتم تكرار الأوامر التي بين القوسين { } ؟

1. for(y=0;y<12;y++) { ..... }
2. for(ahmad=3;ahmad<18;ahmad++) { ..... }
3. for(m=5;m<10;m++) { ..... }
4. for(x=10;x>0;x--) { ..... }
5. for(y=0;y<=12;y++) { ..... }

ملحوظة : x-- تعني أننا في كل مرة سنقلل من قيمة المتغير ثم ننفذ الأوامر التي بين القوسين ثم نقلل قيمة المتغير مرة أخرى وننفذ الأوامر التي بين القوسين وهكذا ...

علامة =< تعني أقل من أو يساوي وعلامة => تعني أكبر من أو يساوي .....

حاول حل الأسئلة الخمسة السابقة ثم انظر للإجابة في الصفحة التالية .

1 for(y=0;y<12;y++) { ..... }

هنا ستكون قيمة المتغير y بصفر ثم سينفذ البك الأوامر التي بين القوسين ثم يزود قيمة Y ويجعلها بواحد ثم ينفذ الأوامر التي بين القوسين مرة أخرى وهكذا إلى أن يصل للرقم ١١ فينفذ الأوامر التي بين القوسين ولن تزيد قيمة المتغير y وتصبح بـ ١٢ لأن الشرط يخبرنا أن y أصغر من ١٢ ... إذن ستكون عدد مرات التكرار للأوامر التي بين القوسين تساوي ١٢ مرة ... لأننا بدأنا العد من صفر (الصفر مرة والواحد مرة ..وهكذا)

2 for(ahmad=3;ahmad<18;ahmad++) { ..... }

هنا ستكون قيمة المتغير الابتدائية بثلاثة ثم ينفذ البك الأوامر التي بين القوسين ثم يزود قيمة المتغير ahmad ويجعلها بـ ٤ ثم ينفذ الأوامر التي بين القوسين وهكذا إلى أن يصل للقيمة ١٧ فينفذ الأوامر التي بين القوسين ولا يجعل قيمة المتغير تساوي ١٨ لأن الشرط يخبرنا بأن قيمة المتغير ahmad أصغر من ١٨ ... وبهذا تكون عدد مرات التكرار اللأوامر التي بين القوسين تساوي ١٥ مرة (١٨- ٣) لماذا ... لأننا بدأنا العد من ثلاثة (ثلاثة تحسب مرة) وانتهينا بـ١٧ ... (يمكنك أن تحسبها بعقلك بدون أي قانون)

3 for(m=5;m<10;m++) { ..... }

تعتبر مثل المثال السابق إذن عدد مرات التكرار تساوي (١٠ - ٥) أي خمس مرات . لأننا بدأن العد من ٥

4 for(x=10;x>0;x--) { ..... }

هنا سيجعل البك القيمة الإبتدائية للمتغير تساوي ١٠ ثم ينفذ الأوامر التي بين القوسين وبعد ذلك يقلل (ينقص) البك من قيمة المتغير فتصبح ٩ ثم ينفذ الأوامر التي بين القوسين ثم ينقص قيمة المتغير فيجعلها تساوي ٨ ثم ينفذ الأوامر التي بين القوسين ... وهكذا إلى أن يصل إلى واحد وينفذ الأوامر التي بين القوسين ولن يجعل قيمة المتغير تساوي صفر لأن الشرط يخبرنا بأن المتغير قيمته أكبر من صفر .

5 for(y=0;y<=12;y++) { ..... }

هنا سيجعل البك القيمة الإبتدائية للمتغير تساوي صفر ثم ينفذ الأوامر التي بين القوسين ثم يزيد قيمة المتغير ويجعلها ٢ ثم ينفذ ما بين القوسين ... وهكذا إلى أن يجعل البك قيمة المتغير بـ ١١ ثم ينفذ ما بين القوسين ثم يجعل قيمة المتغير تساوي ١٢ (لاحظ في هذه المرة الشرط يخبرنا بأن المتغير أصغر من أو

يساوي ١٢) لهذا سيسمح البك بجعل قيمة المتغير تساوي ١٢ ثم ينفذ ما بين القوسين . وبهذا يكون عدد مرات تكرار تنفيذ الأوامر ١٣ مرة .

مما سبق يتضح لنا أننا لو أردنا على سبيل المثال عمل تكرار تنفيذ مجموعة من الأوامر ١٠ مرات فيمكن كتابة الكود بعدة طرق نختار منها ما نشاء .. من هذه الطرق ما يلي

```
for(x=0;x<10;x++) { ..... }
for(x=1;x<=10;x++) { ..... }
for(x=10;x>0;x--) { ..... }
for(x=10;x>=1;x--) { ..... }
```

ستجدني في هذا الكتاب أكثر من استخدام الطريقة الأولى ... لأنني إعتدت عليها .

وفي نهاية هذه التجربة إليك هذه المعلومة الصغيرة ... إذا أردت أن تكرر تنفيذ أمر واحد فقط عدد من المرات وليس مجموعة من الأوامر فلا يشترط أن تضع قوسين .. فمثلا لو أردنا أن نجعل البك ينتظر خمس ثواني بأن يكرر الأمر delay_ms(1000); خمس مرات سنكتب الكود التالي :

```
for(x=0;x<5;x++) delay_ms(1000);
```

أو نكتبها بالشكل التالي

```
for(x=0;x<5;x++) { delay_ms(1000);}
```

كما يمكننا ضغط مفتاح enter فنجعل الكود أكثر تنسيقاً

```
for(x=0;x<5;x++)
delay_ms(1000);
```

قد يخطر في بالك وأن تقول لماذا نستخدم جملة for ويمكننا مباشرة كتابة delay_ms(5000); .. نعم هذا صحيح لكنني فعلت ذلك لتوضيح الأمر بضرب مثال تستطيع فهمه الآن .. أي أن الأمر للإيضاح فقط.

التجربة (٥)

هل تتذكر التجربة الثالثة ... لقد كان الهدف منها هو إضاءة الليدات الموصلة على الأطراف من B0 إلى B7 ثم اطفاءها حيث كتبنا الكود التالي

```
void main()
{ TRISB=0;
 Loop:
PORTB=0XFF;delay_ms(1000);PORTB=0;Delay_ms(1000);
Goto loop;}
```

في هذه التجربة سنكتب برنامج يؤدي نفس الوظيفة لكن بطريقة أخرى

```
void main()
{ TRISB=0;PORTB=0;
Loop:
PORTB=~PORTB; delay_ms(1000);
goto loop;}
```

هل لاحظت ذلك الأمر الجديد PORTB=~PORTB; ؟؟

في بداية البرنامج جعلنا قيمة جميع محتويات PORTB بأصفار وذلك باستخدام الأمر PORTB=0; العلامة ~ تعني اعكس جميع قيم البتات التي في المسجل PORTB أي البت الذي قيمته بصفر ستصبح قيمته بواحد والبت الذي قيمته بواحد ستصبح قيمته بصفر . والآن هيا بنا نعرف كيف سيعمل البرنامج؟

في أول مرة ينفذ فيها البك الأمر PORTB=~PORTB; بما أن كل البتات BITS قيمتها بأصفار إذن سيعكس البك الحالة التي عليها هذه الـ bits أي سيجعل قيمة كل بت بواحد بدلا من صفر أي ستضيء

جميع الليدات ..ثم ينتظر البك لمدة ثانية بسبب الأمر ;(delay_ms(1000 و الأمر ;goto loop سيجعله يعيد تنفيذ الأمر ;PORTB=~PORTB مرة أخرى وبالتالي سيعكس حالتها (لقد كانت حالة كل bit تساوي واحد) إذن سجعل كل bit يساوي صفر وبالتالي ستنطفئ جميع الليدات .. وهكذا

في الدروس السابقة علمت أننا نستطيع عمل تكرار لمجموعة من الأوامر بإستمرار بإستخدام الطريقة التالي

```
:

LOOP:

.....
.....
.....

goto LOOP;
```

في هذه المنطقة نكتب الكود الذي نريده أن يتكرر بإستمرار .

وهناك طريقة أخرى تفضل في الإستخدام عن هذه الطريقة وتكون بالشكل التالي :

```
while (1)
{
.....
.....
}
```

في هذه المنطقة نكتب الكود الذي نريده أن يتكرر بإستمرار .

وهناك طريقة أخرى وهي

```
for(;;)
{
.....
.....
}
```

في هذه المنطقة نكتب الكود الذي نريده أن يتكرر بإستمرار .

التجربة (٦)

في هذه التجربة سنتعلم أشياء جديدة أرجو أن تستفيد و تستمتع بها . من الآن لست مضطراً لتطبيق التجربة في الواقع وتوصيل الأسلاك وو.... سنرى هل ستعمل الدائرة أم لا وسنرى كيف ستعمل بإستخدام أحد برامج المحاكاة فمن خلال برنامج المحاكاة سترسم الدائرة وتضع عليها البرنامج وتختبر دائرتك بكل سهولة .... واعلم عزيزي القارئ أن برامج المحاكاة لها أهمية كبيرة فهي توفر الكثير من الوقت كما أنك تستطيع من خلالها أن تستخدم قطع الكترونية لا تمتلكها ... والعديد من الميزات الأخرى ..

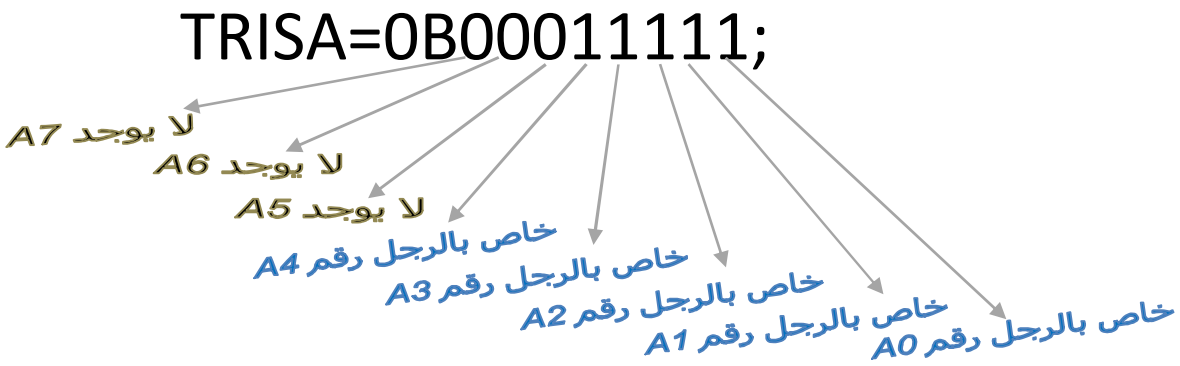
في هذه التجربة المطلوب منا تشغيل الليد الموصل على B0 عند الضغط على المفتاح(switch) و في حالة عدم الضغط على المفتاح يتم إطفاء الليد .علما بأن المفتاح موصل على الطرف رقم A0 .

كتابة البرنامج :-

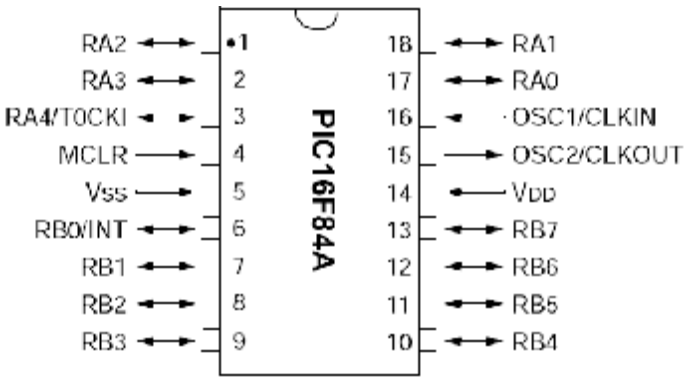
في هذه التجربة لن تكون الخطوة الأولى هي توصيل الدائرة ولكن خطوتنا الأولى هي كتابة البرنامج . وإليك هذه المعلومات التي ستساعدك على كتابة البرنامج .. أولاً عندما نبرمج نأخذ في الإعتبار أن المفتاح يكون دخل وليس خرج ... وهذا شيء ثابت يجب أن لا ننساه فأي مفتاح نستخدمه في أي تجربة سنجعله دخل. ثانيا : المفتاح موصل بالطرف A0 لذلك سنستخدم TRISA لتوظيف الدخل والخرج حيث أن أول بت وهو الخاص بـ A0 ستكون قمته بواحد ... هل تتذكر العلاقة الخاصة بـ TRIS

1 تعني دخل 0 تعني خرج

هنا سنكتب الأمر التالي : TRISA=0B00011111;



هل لاحظت الشكل السابق..؟؟ كما تعلم أن أول بت خاص باللرجل رقم A0 وبما أنه بواحد أي ستصبح الطرف A0 دخل أما بقية الرجول فهي لا تعنينا لأننا لن نوصل بها أي شيء فيمكننا جعلها بواحد أو بصفر كما نشاء ... في النوع الذي نستخدمه من المايكروكنترولر PIC16F84A لا يوجد A5,A6,A7 انظر للشكل التالي

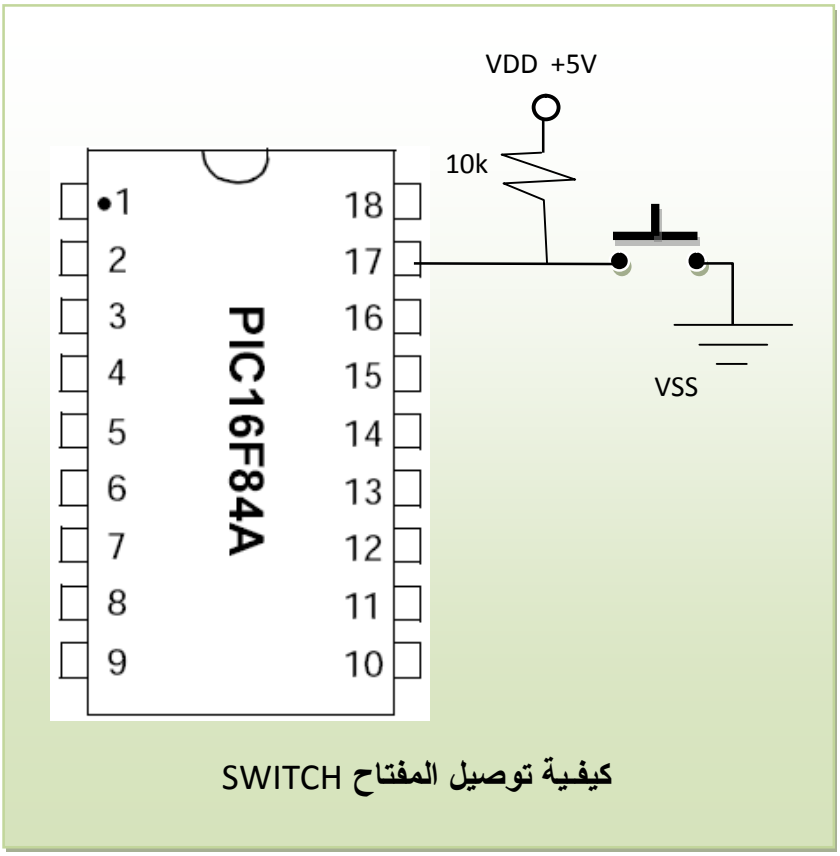


ومع ذلك لابد أن نضع قيمة لهذه الـ BITs ولا نتركها فارغة فنضع إما صفر وإما واحد وفي أي الحالتين لن يؤثر ذلك على عمل المايكروكنترولر ... لكن يجب الإنتباه أننا نملأ المسجل TRISA كاملاً لجميع الـ BITs سواءاً احتجناها جميعا أو احتجنا بعضها كما في حالتنا هذه.

ملحوظة : يفضل جعل A4 بدخل وليس خرج ... وسنشرح ذلك لاحقاً.

كيف سنوصل السوتش ؟؟

سنوصله كما بالشكل التالي ←



كيفية توصيل المفتاح SWITCH

لماذا هذه التوصيلة ؟؟ .. إذا كان المفتاح محرراً(لم يتم الضغط عليه) سيكون الجهد الوصل إلى البك عن طريق الطرف A0 جهد موجب أو بمعنى أصح hi وإذا تم الضغط على المفتاح فسيصل للبك جهد صفرlow وهذا يعني أن البك إذا أراد أن يعرف هل المفتاح محرراً أم لا سينظر إلى الجهد الذي على الطرف A0 فلو كان الجهد hi أي في لغتنا واحد سيعلم أن المفتاح محرراً. وإذا كان الجهد low أي في لغتنا صفر سيعلم البك أن المفتاح مضغوط عليه (غير محرر) .. إذن عندما نبرمج البك ونريد منه أن ينفذ أمر معين عند الضغط على السويتش فنقول للبك لو الطرف رقم A0 تساوي صفر (تم الضغط على المفتاح) نفذ الأمر الفلاني وإذا كانت الطرف رقم A0 تساوي واحد (أي أن المفتاح محرر) نفذ الأمر الفلاني.

ولكن كيف سأكتب الكود لتنفيذ ذلك ؟؟؟ ... ببساطة سنستخدم جملة if حيث سنكتب الكود التالي:

```
void main()
{ TRISA=0B00011111;
TRISB=0;
loop:
If(PORTA.F0==0)
    {
     PORTB.F0=1;
     }
If(PORTA.F0==1)
    {
     PORTB.F0=0;
     }
Goto loop;
}
```

توظيف الطرف A0 على أنها دخل

لأن الليد موصل بـ B0 إذن سنجعل B0 خرج

هذا العنوان سيرجع إليه البك دائما لنجعل البك يختبر المفتاح بشكل دائم وليس مرة واحدة فقط في منتهى

إختبر هل تم الضغط على السويتش .. إذا حدث ذلك نفذ الأوامر التي بين القوسين الخاصين بجملة IF

اجعل الليد يضيء ... ولكن هذا الأمر لن ينفذ إلا في حالة تحقق الشرط وهو أن نضغط على المفتاح

اختبر هل السويتش محراراً .. إذا حدث ذلك نفذ الأوامر التي بين القوسين التاليين

اجعل الليد ينطفئ ... ولكن هذا الأمر لن ينفذ إلا في حالة تحقق الشرط وهو ان المفتاح محرر

اذهب للعنوان loop وذلك لتختبر مرة أخرى هل المفتاح محراراً أم لا

والآن لنشرح بالتفصيل الأمر if أو ما نسميه جملة if .

If تعني لو . أي لو حدث الشيء الفلاني (تحقق الشرط) نفذ الأمر الفلاني. والشكل العام لجملة if كما يلي

```
If (الشرط) { الأوامر التي نريد تنفيذها إذا تحقق الشرط}
```

ففي المثال السابق استخدمنا جملتين if الجملة الأولى هي If(PORTA.F0==0) هنا الشرط هو أن تكون الطرف A0 تساوي صفر (أي تم الضغط على المفتاح) والأمر الذي سيتم تنفيذه في حالة تحقق الشرط هو PORTB.F0=1; أي سيتم إضاءة الليد

ملحوظة مهمة: البك عندما يأتي لينفذ جملة IF سيختبر الشرط وفي حالة عدم تحققه لن يلتفت إلى الأوامر التي بين القوسين الخاصين بهذه الجملة.

ويوجد شكل آخر لجملة IF وهذا الشكل نسميه (if .... else) والجديد هنا هو else وهي تعني أنه في حالة عدم تحقق الشرط نفذ الأوامر التي تلي كلمة else والتي ستكون بين قوسين.

```
If (الشرط) { نفذ هذه الأوامر في حال تحقق الشرط}
else { وفي حالة عدم تحقق الشرط السابق نفذ هذه الأوامر}
```

وإذا أردنا استخدام هذا الشكل في برنامجنا السابق سنكتب الكود التالي

```
void main() { TRISA=0B00011111; TRISB=0;
loop : if (PORTA.F0==0)
            {          PORTB.F0=1;       }
else {
       PORTB.F0=0;
      }
goto loop; }
```

لعلك تعلم جيداً أن الأقواس ومكان وجودها (أي استخدمنا enter أم لا وكذلك المسافات) كل ذلك لك فيه مطلق الحرية . وكما قلنا في جملة FOR إذا كان الأمر الذي سينفذ أمر واحد فقط وليست مجموعة أوامر فلا يشترط وضع الأقواس فهكذا الحال مع if وبهذا يمكن أن نكتب الكود بالشكل التالي:

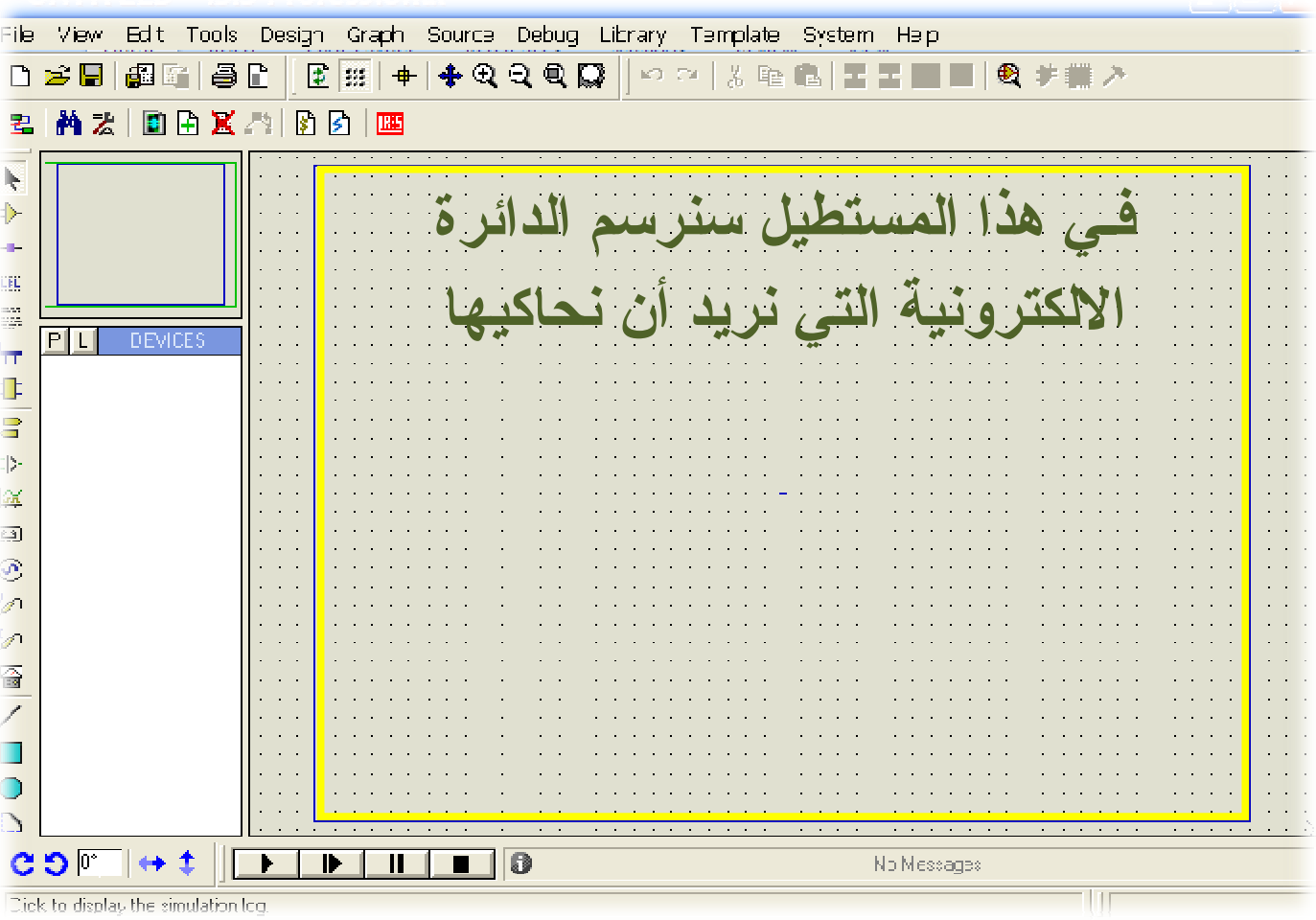
```
void main() { TRISA=0B00011111; TRISB=0;
loop:
if (PORTA.F0==0) PORTB.F0=1;
else PORTB.F0=0;
goto loop;
}
```

... ولعلك لاحظت أننا داخل الشرط كتبنا علامة تساوي مرتين ومن الخطأ أن نكتبها مرة واحدة وهذه تعتبر ضمن syntax اللغة أي يجب أن تكتب هكذا ولكن هذا في حالة الشرط فقط وليس في الأوامر الأخرى العادية .

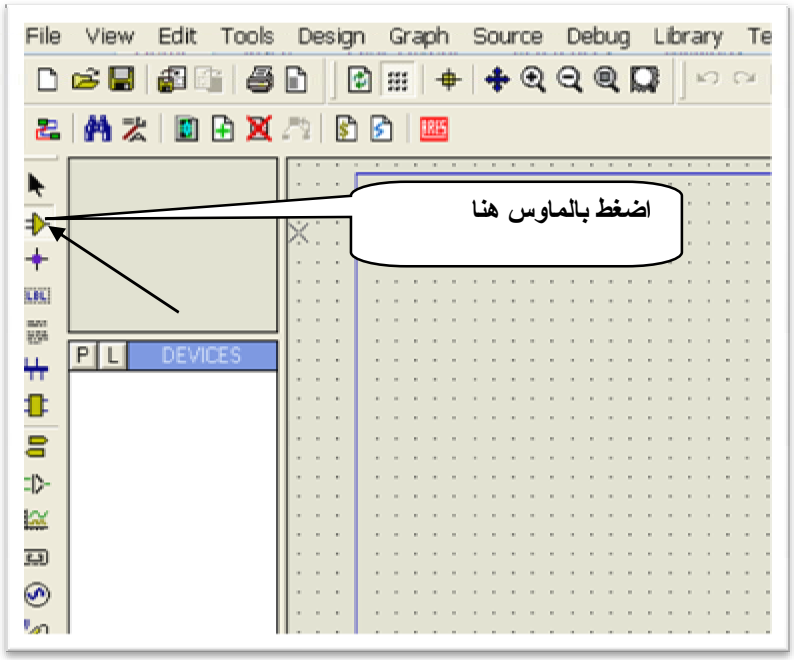
وبهذا نكون انتهينا من شرح البرنامج ... والآن هيا بنا لنحاول تشغيل الدائرة ولكن باستخدام برنامج للمحاكاه اسم البرنامج proteus (انظر للملحق الخاص بالبرامج) أثناء بدء البرنامج تظهر الشاشة التالية

قد تختلف هذه الشاشة في حالة استخدامك لإصدار آخر من البرنامج .

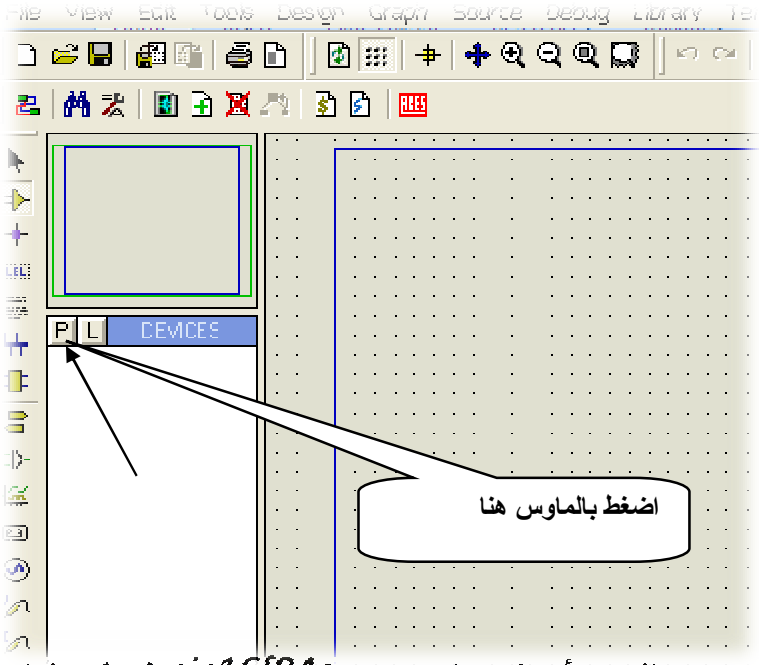
وهذه الشاشة التي ستجدها بعد فتحك للبرنامج



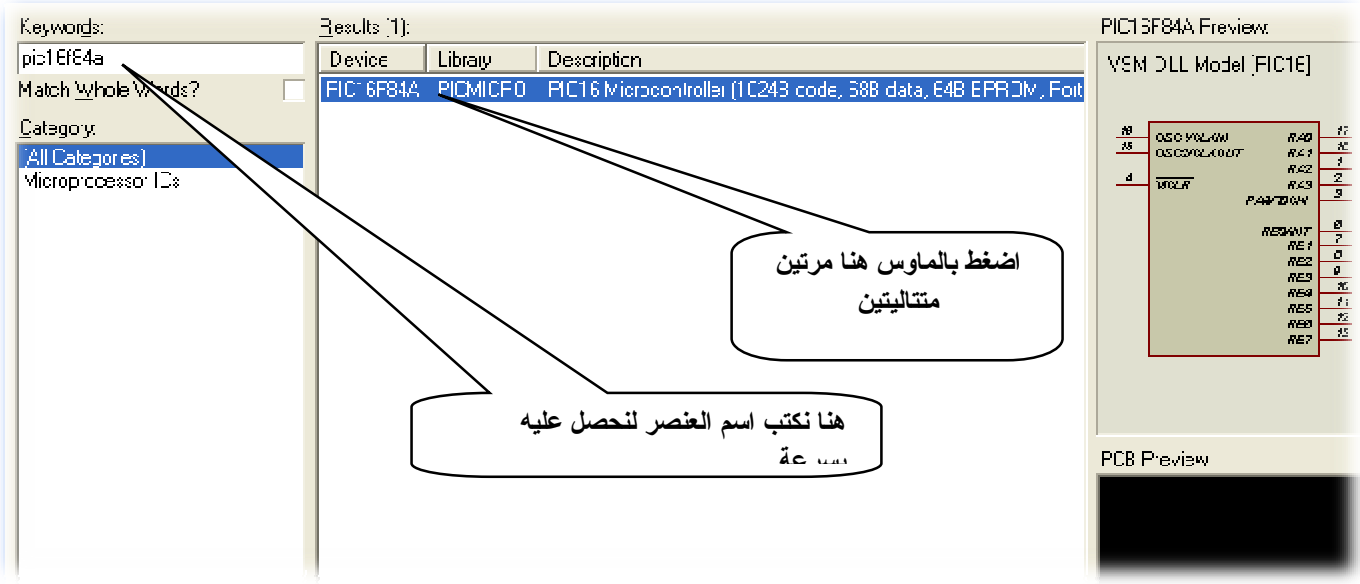
سنختار بعد ذلك component mode وذلك بالضغط بالماوس على  كما بالشكل



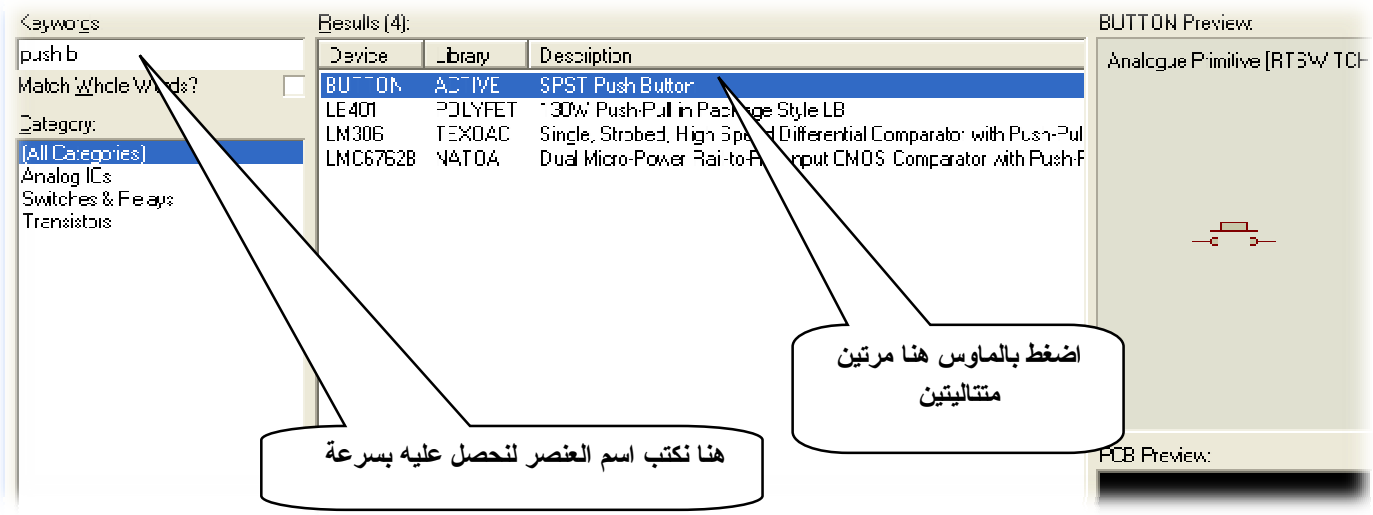
ثم نضغط على حرف الـ p الذي ستجده في الجزء اليسار من البرنامج حيث يرمز هذا الحرف إلى pick devices والذي من خلاله سنحضر المكونات أو القطع الإلكترونيات .

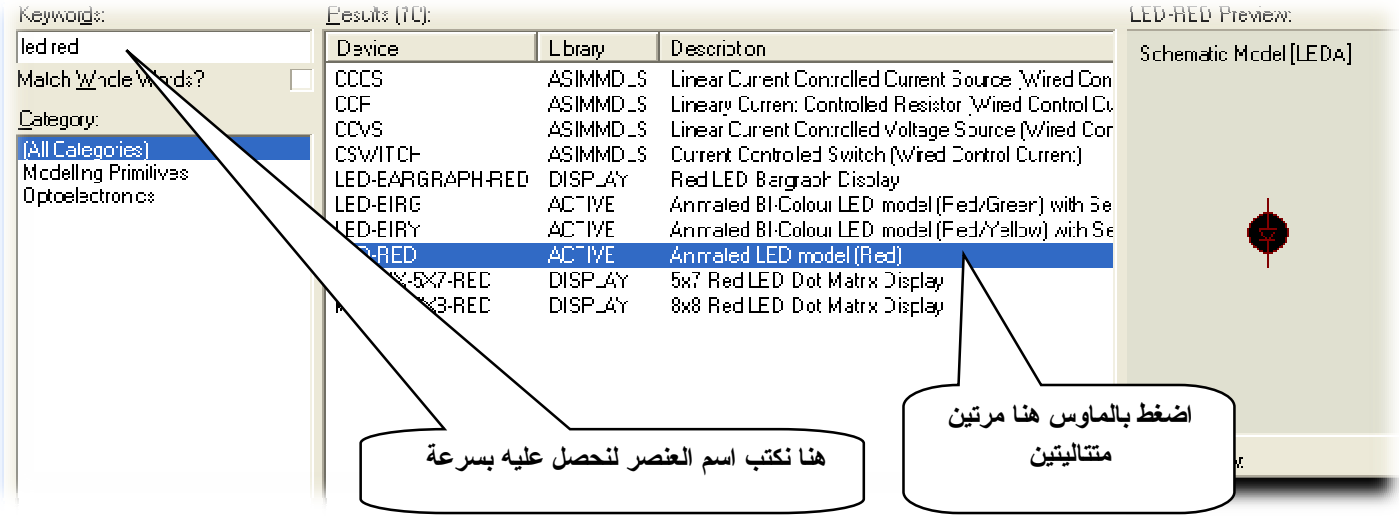


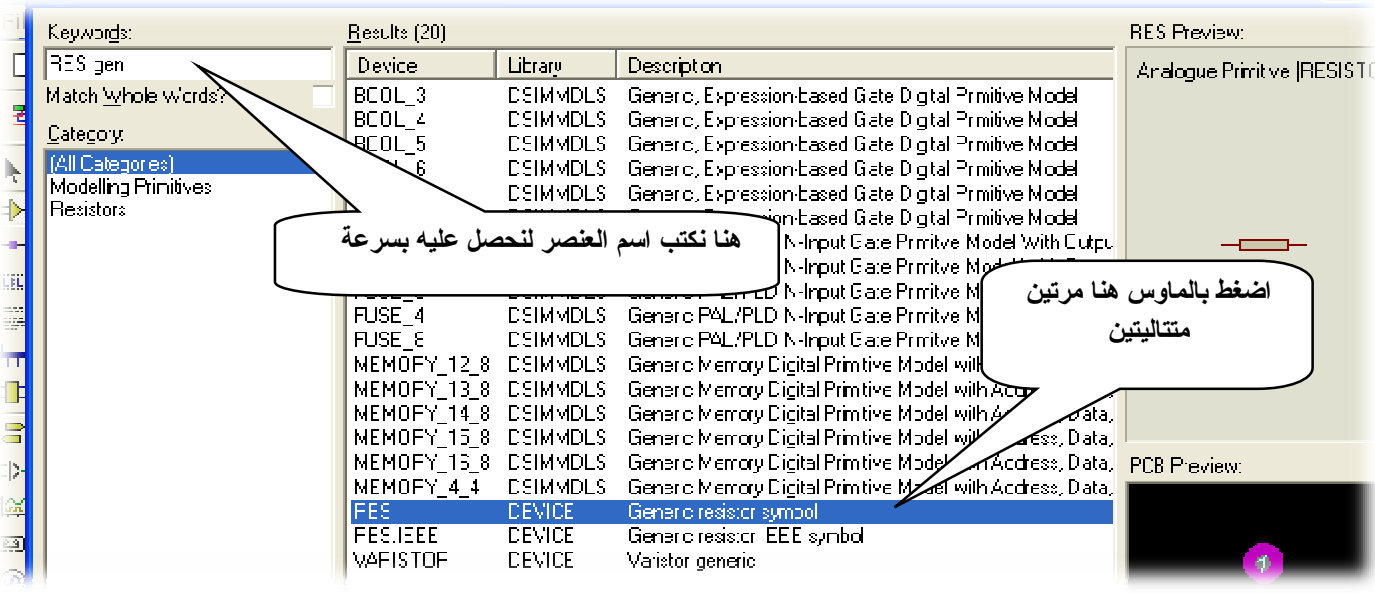
ثم نبدأ بإختيار أول عنصر وذلك بأن نكتب اسمه هو pic16f84a فيظهر لنا على اليمين فنضغط بالماوس مرتين متتاليتين عليه.



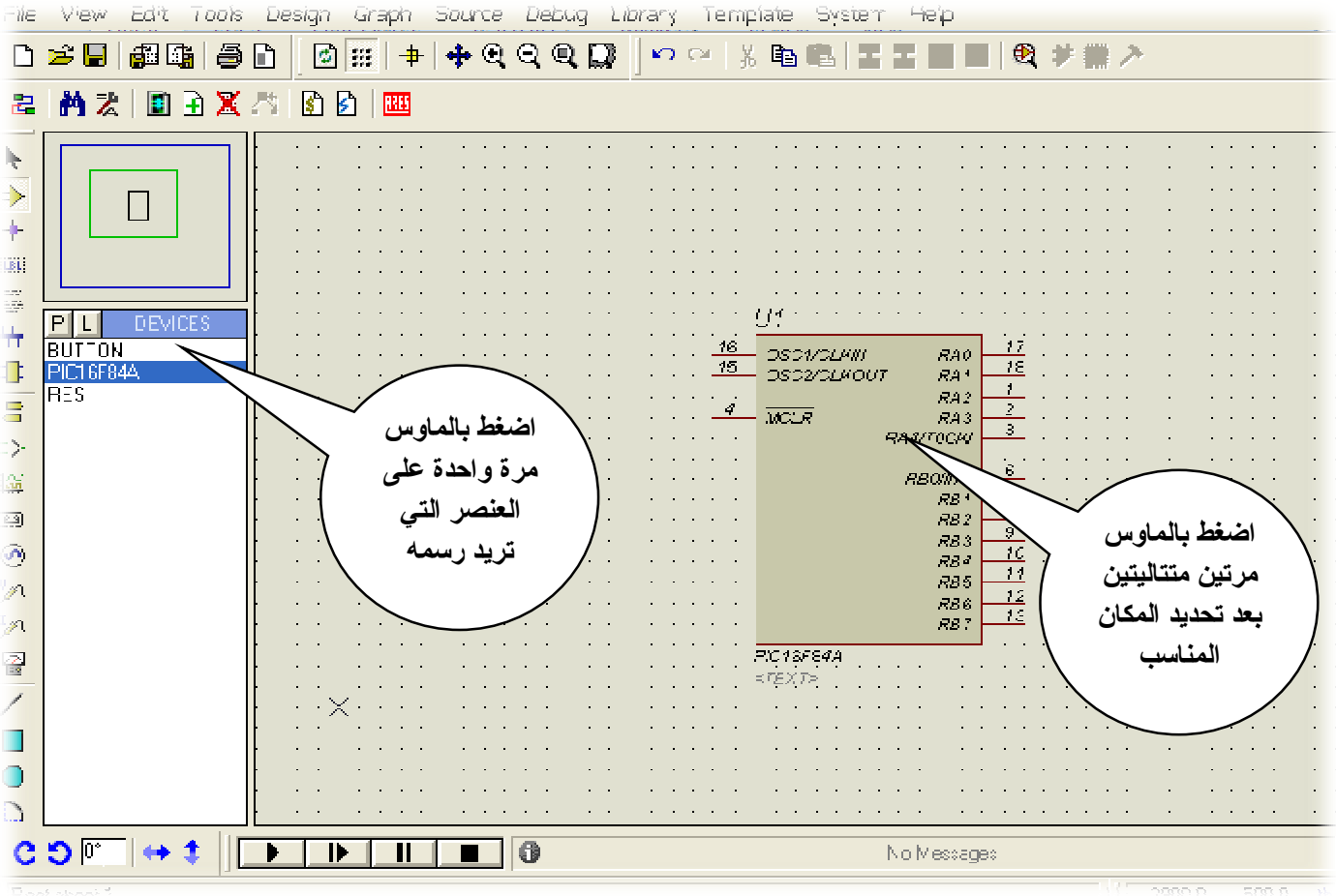
وبنفس الطريقة سنختار بقية عناصر الدائرة push button و resistor 10k وresistor 470 و red led كما بالأشكال التالية


Keywords:
push b
Match Whole Words?
Category:
(All Categories)
Analog ICs
Switches & Relays
Transistors
Results (4):
Device
Library
Description
BUTTON
ACTIVE
SPST Push Button
LE401
POLYFET
LM306
TEXOAC
LMC6762B
NATOA
BUTTON Preview:
Analogue Primitive [RTSW TCH
PCB Preview:
هنا نكتب اسم العنصر لنحصل عليه بسرعة
اضغط بالماوس هنا مرتين متتاليتين

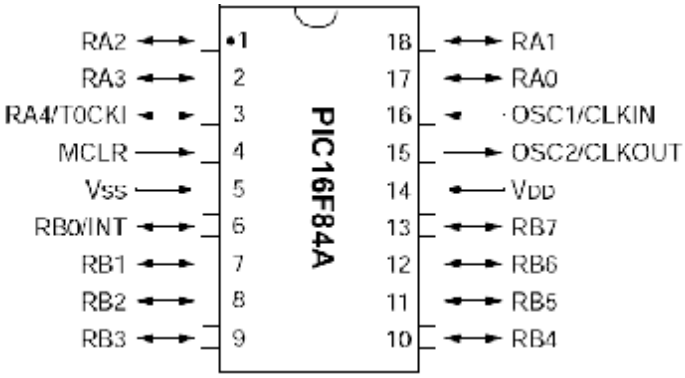

Keywords:
led red
Match Whole Words?
Category:
(All Categories)
Modelling Primitives
Optoelectronics
Results (70):
Device
Library
Description
CCCS
ASIMMDLS
Linear Current Controlled Current Source (Wired Con
CCF
ASIMMDLS
Linear Current Controlled Resistor (Wired Control Cu
CCVS
ASIMMDLS
Linear Current Controlled Voltage Source (Wired Cor
CSWITCH
ASIMMDLS
Current Controlled Switch (Wired Control Current)
LED-BARGRAPH-RED
DISPLAY
Red LED Bargraph Display
LED-BIRG
ACTIVE
Annealed Bi-Colour LED model (Red/Green) with Se
LED-BIRY
ACTIVE
Annealed Bi-Colour LED model (Red/Yellow) with Se
ACTIVE
Annealed LED model (Red)
DISPLAY
5x7 Red LED Dot Matrix Display
DISPLAY
8x8 Red LED Dot Matrix Display
LED-RED Preview:
Schematic Model [LEDA]
هنا نكتب اسم العنصر لنحصل عليه بسرعة
اضغط بالماوس هنا مرتين متتاليتين


Keywords:
RES gen
Match Whole Words?
Category:
(All Categories)
Modelling Primitives
Resistors
Results (20)
Device
Library
Description
BCOL_3
DSIMMDLS
BCOL_4
BCOL_5
FUSE_4
FUSE_8
MEMORY_12_8
MEMORY_13_8
MEMORY_14_8
MEMORY_15_8
MEMORY_16_8
MEMORY_4_4
RES
DEVICE
Generic resistor symbol
RES.IEEE
DEVICE
Generic resistor IEEE symbol
VARISTOR
DEVICE
Varistor generic
RES Preview:
Analogue Primitive [RESIST
PCB Preview:
هنا نكتب اسم العنصر لنحصل عليه بسرعة
اضغط بالماوس هنا مرتين متتاليتين

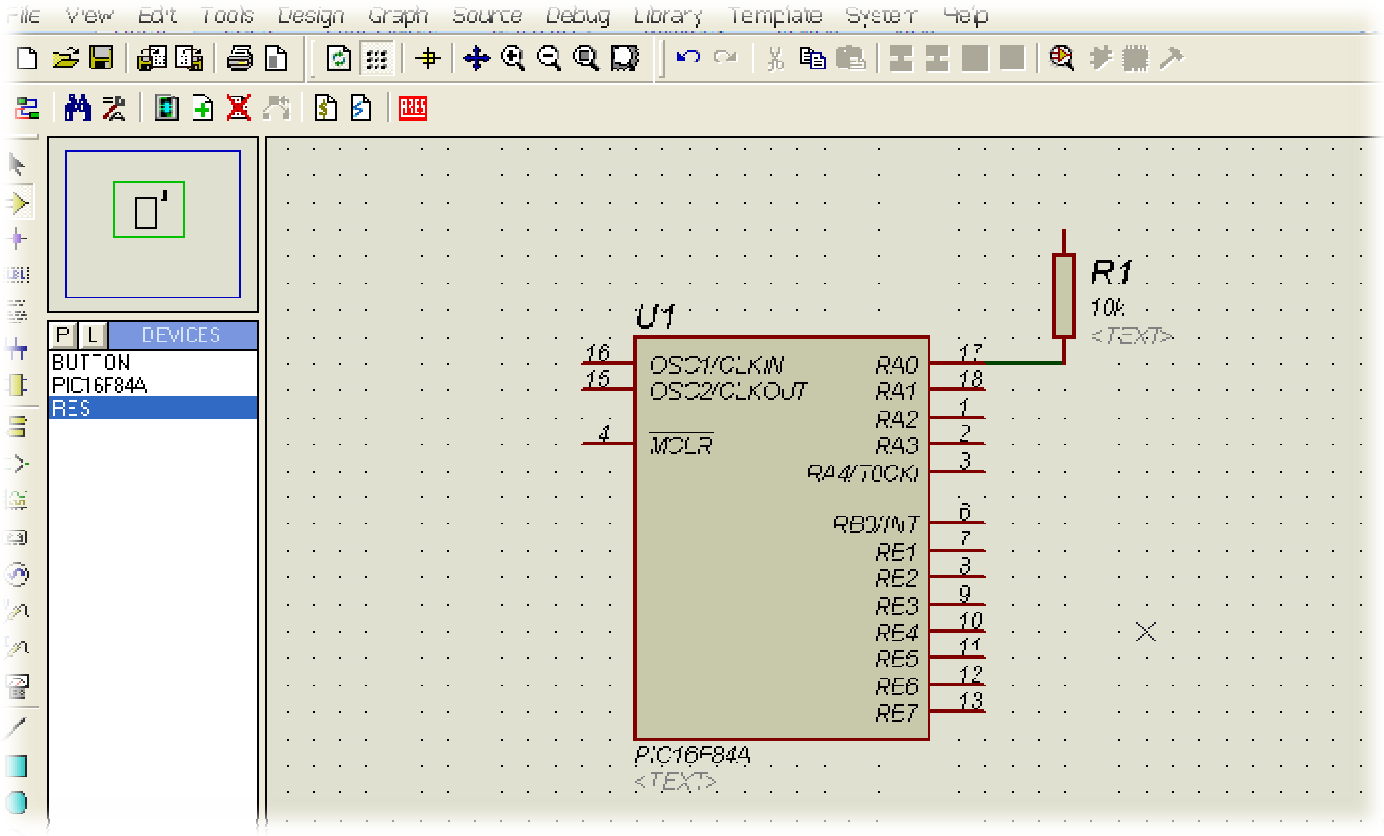
والآن نغلق هذه النافذة ونبدأ برسم الدائرة فسنجد قائمة على اليسار بها العناصر التي اخترناها فنضغط بالماوس على العنصر الذي نريد رسمه وليكن pic16f84a ثم نأتي في المنطقة المخصصة للرسم ونضغط ضغطتين متتاليتين بعد أن نحدد المكان المناسب .



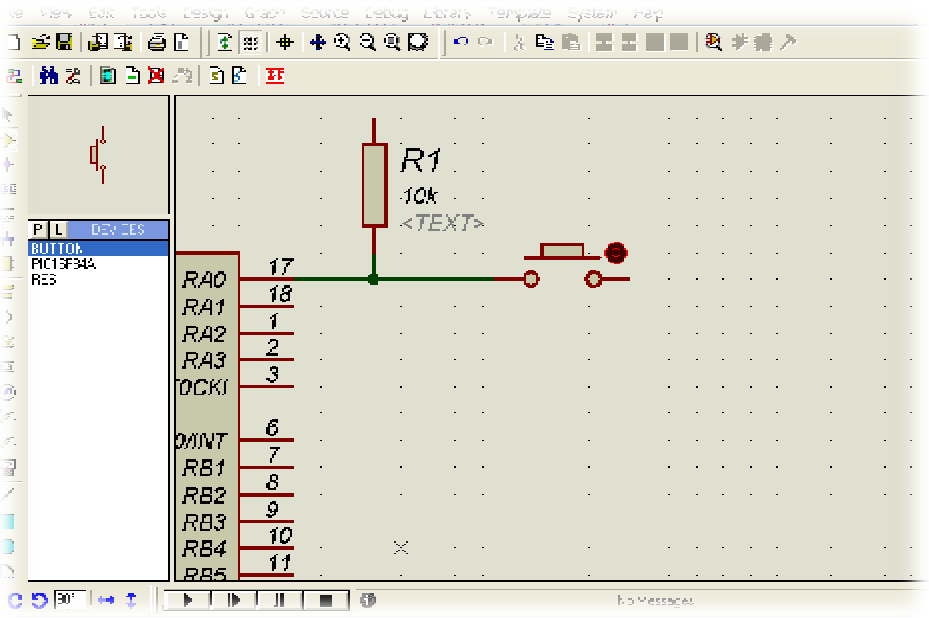
كما يمكننا التكبير والتصغير zoom عن طريق بكرة الماوس التي بين الزرين الأيمن والأيسر وستلاحظ أن أطراف المايكروكنترولر ليست مرتبة كما في الحقيقة . انظر للشكل التالي لتعرف الفرق ولكن هذا الترتيب الجديد أسهل في عملية توصيل الدائرة .



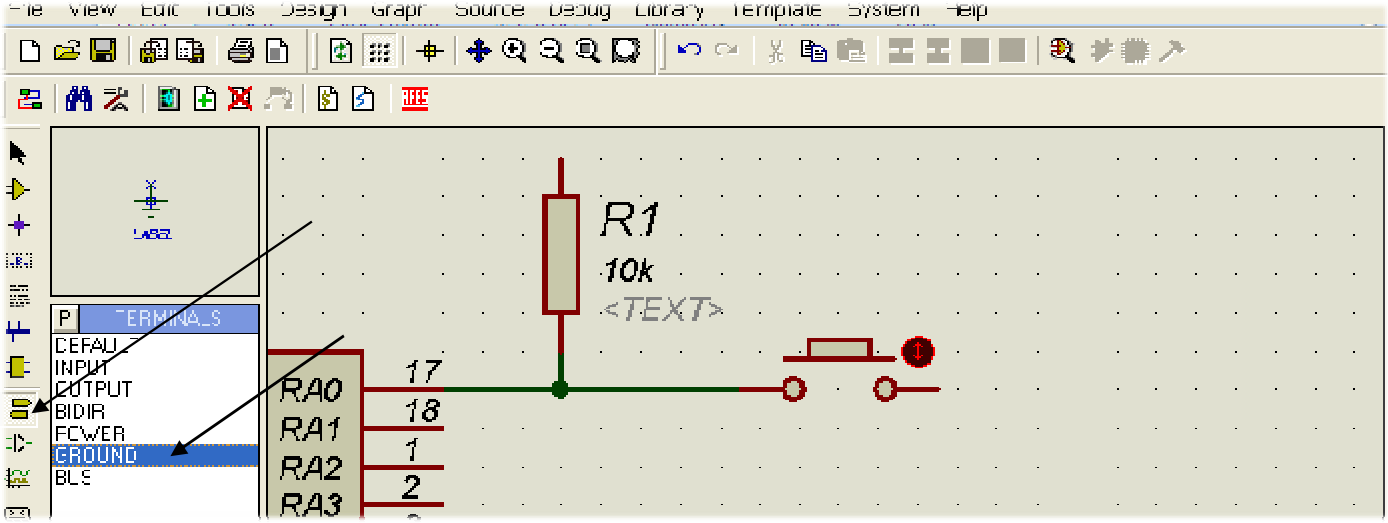
إذا اقتربت من من نهاية أي طرف سظهر قلم تستطيع من خلاله رسم أسلاك التوصيل . والآن لنرسم المقاومة سنضغط عليها من القائمة ثم نضغط مرة على المكان المخصص لرسم الدائرة ثم نحدد المكان المناسب لها ويمكننا تغيير وضعها من أفقي لرأسي أو العكس بالضغط على الزر + في يمين لوحة المفاتيح ثم ثم نضغط مرة أخرى. ثم نوصل A0 بطرف المقاومة كما بالشكل التالي.



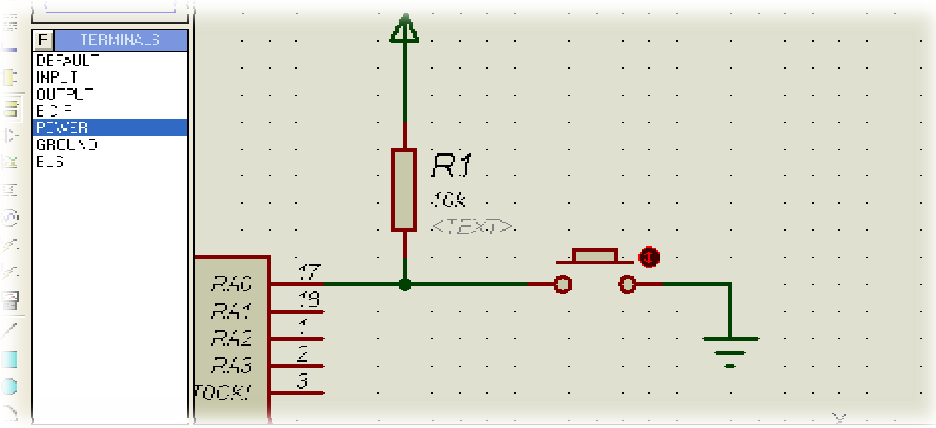
وبنفس الطريقة نرسم المفتاح



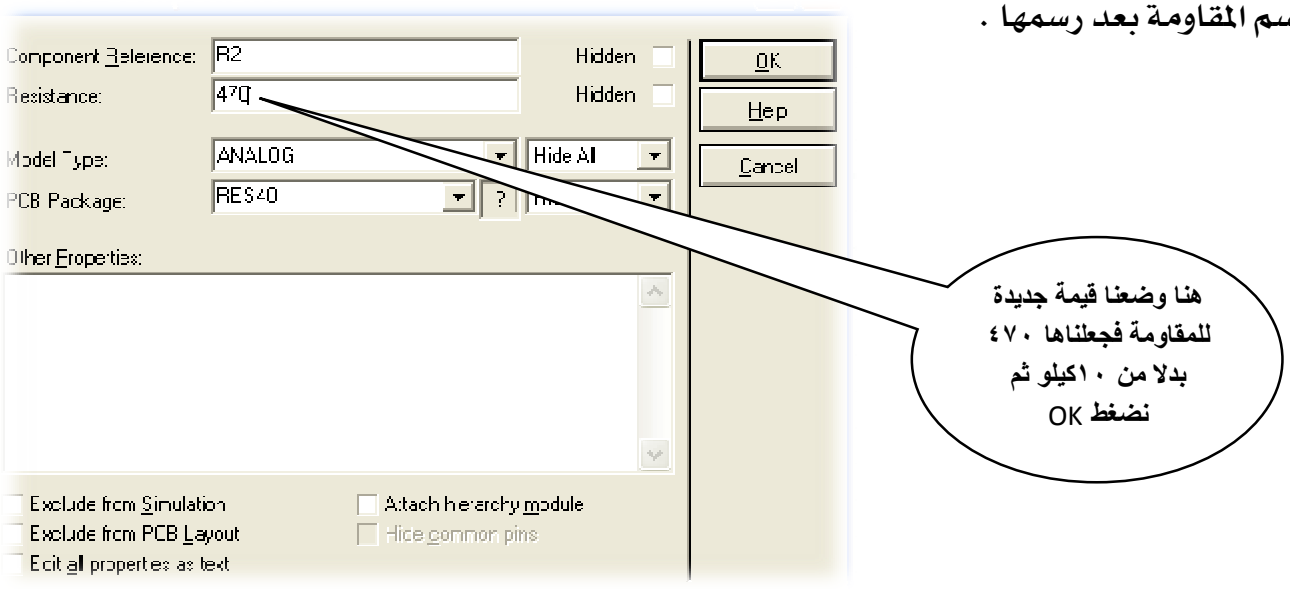
ولرسم الأرضي VSS و VDD نختار بالماوس من الأدوات التي على يسار البرنامج  ثم نختار GROUND عند رسم الأرضي ونختار POWER عند رسم VDD



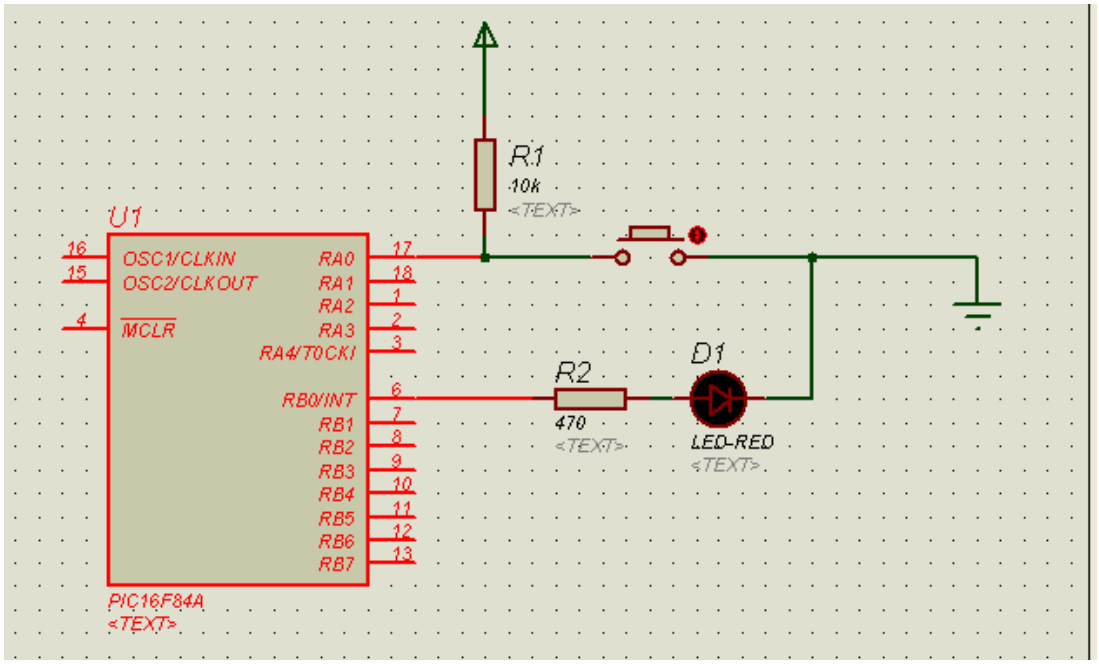
ونضغط على المكان المخصص للرسم مرة ثم نحدد المكان والشكل المناسب ثم نضغط مرة أخرى كما بالسابق



وبقي أن نرسم الليد وعند رسمه ووضع له مقاومة يمكننا تغير قيمة المقامة ونجعلها ٤٧٠ أوم مثلا بالضغط على جسم المقاومة بعد رسمها .



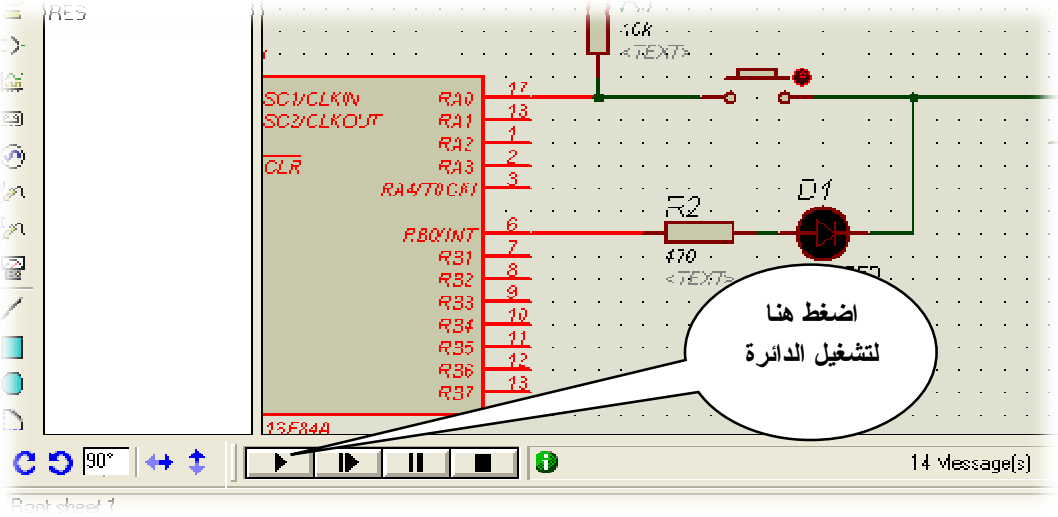
ونكمل بقية الرسم فيصبح الشكل النهائي كالتالي .



ملحوظة تساعدك في التحكم فيما تراه من الدائرة : ستجد أعلى قائمة المكونات التي اخترتها مربع اضغط عليه مرة بالماوس ثم حرك الماوس ببطء حتى يظهر لك الشكل المناسب ثم اضغط بالماوس مرة أخرى.

الخطوة الأخيرة في عملية التصميم هي أن تضغط بالماوس على جسم البك مرتين متتاليتين فتظهر لك نافذة تحدد فيها تردد المذبذب المستخدم وهو ١,٠٨ وتحدد مكان ملف الهيكس (ستجده في المكان الذي حفظت فيه المشروع البرمجي) سيكون ملف بإسم التجربة وامتداده hex فمثلا إذا سميت المشروع E_6 وحفظته على سطح المكتب مثلا .. اختار سطح المكتب بعد الضغط على browse وستجد ملف E_6.hex اختاره واضغط

والآن اضغط على علامة  وهكذا ستعمل الدائرة



من المفترض أنك إذا ضغطت الآن على المفتاح سيضيء الليد وإذا رفعت يدك من على الماوس (لم تضغط على المفتاح) سينطفئ الليد .

ملحوظة إذا لم تكن غيرت قيمة المقاومة وجعلتها صغيرة بدلا من ١٠ كيلو لن تلاحظ إضاءة الليد. ولعلك لاحظت أننا لم نوصل vdd وكذلك vss الخاص بالبك ولم نوصل أيضا التوصيلة الخاصة بالمذبذب في الواقع إن برنامج المحاكاة هذا يدعم هذه الأشياء تلقائيا دون الحاجة لهذه التوصيلات .. ولكن إذا أردنا تطبيق الدائرة في الواقع فيجب علينا حينذاك بتوصيل هذه التوصيلات .

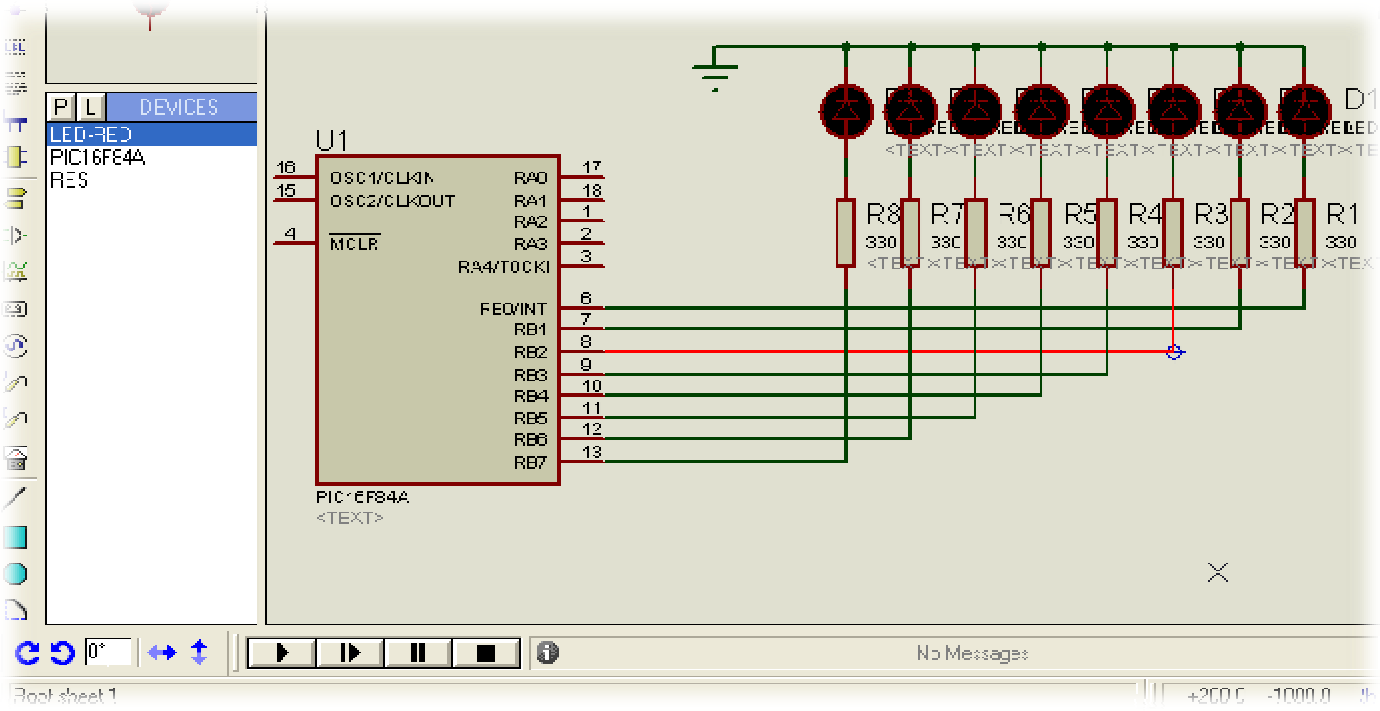
نفذ الخطوات السابقة خطوة خطوة وتابع الفيديو المرفق..... ستجد عندما تعمل الدائرة أن الأمر في غاية الروعة والمتعة.

التجربة (٧)

في هذه التجربة المطلوب منا تشغيل الليدات بأشكال مختلفة شكل يلي الشكل الآخر

الشكل الأول يضيء ليد ثم الذي يليه ثم الذي يليه وهكذا ثم يطفئهم جميعا .. والشكل الثاني يجعل الإضاءة تبدأ من المنتصف إلى الأطراف . والشكل الثالث يضيئهم جميعا ثم ينتظر ثم يطفئهم جميعا... ويكرر البرنامج السابق بإستمرار.

١- سنفتح برنامج proteus ونرسم به الدائرة التالية .



٢- الآن سفتح برنامج MikroC وننشئ مشروع جديد ونكتب الكود التالي . اقرأ الكود التالي(في الصفحة التالية) وحاول أن تتخيل ماذا سينفذ البك أثناء تنفيذ كل أمر. وستلاحظ في الكود التالي وجود علامتي // وهذه العلامة لن تؤثر على عمل البرنامج (أثناء تخيلك لما سينفذه البك اعتبر هذه العلامة وما بعدها في نفس السطر غير موجودين) . إنها تدل على أنها وما بعدها من الكود في نفس السطر لا ينفذه البك . وفائدة هذه العلامة أن نضع تعليق أو ملاحظات على كل سطر وأنت حر فيما تكتبه ولست متقيد بأي شيء . ويستخدم المبرمجون

هذه العلامة دائما لتوضيح ما يكتبوه في أي سطر ليفهمه غيرهم أو لتنظيم الكود وفهمه بسهولة

.

```
void main()
{TRISB=0; PORTB=0;
loop:
PORTB=0B00000001; delay_ms(500);          // L0 ON
PORTB=0B00000011; delay_ms(500);          //L0,L1 ON
PORTB=0B00000111; delay_ms(500);          //L0,L1,L2 ON
PORTB=0B00001111; delay_ms(500);          //L0,L1,L2,L3 ON
PORTB=0B00011111; delay_ms(500);          //L0,L1,L2,L3,L4 ON
PORTB=0B00111111; delay_ms(500);          //L0,L1,L2,L3,L4,L5 ON
PORTB=0B01111111; delay_ms(500);          //L0,L1,L2,L3,L4,L5,L6 ON
PORTB=0B11111111; delay_ms(500);          // L0,L1,L2,L3,L4,L5,L6,L7 ON
PORTB=0B00000000; delay_ms(500);          // ALL LEDs OFF
        PORTB=0B00011000; delay_ms(500);  // L3,L4 ON
        PORTB=0B00111100; delay_ms(500); //L2-L5 ON
        PORTB=0B01111110; delay_ms(500); //L1-LS ON
        PORTB=0B11111111; delay_ms(500); // ALL LEDs ON
        PORTB=0B00000000; delay_ms(500); // ALL LEDs OFF
PORTB=0B11111111; delay_ms(500);          // ALL LEDs ON
PORTB=0B00000000; delay_ms(500);          //  ALL LEDs OFF
PORTB=0B11111111; delay_ms(500);          // ALL LEDs ON
```

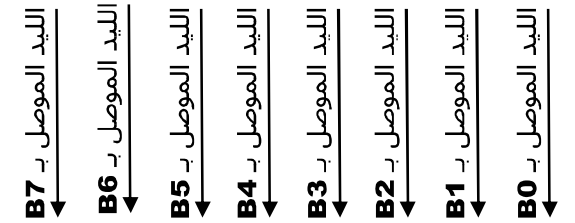
```
PORTB=0B00000000; delay_ms(500);          //  ALL LEDs OFF

goto loop;}
```

هل تخيلت ما سيحدث جيداً عند تنفيذ كل أمر . سأعتبر أن أول سطر بعد كلمة loop هو السطر الأول السطر الأول سيجعل الليد الموصل بـ B0 يضيء وأنا اقترحت اسم لهذا الليد وسميته L0 لذلك تجدني كتبت ملاحظة بعد علامتي // حيث كتبت L0 ON وكأني أقول أن هذا الأمر سيجعل الليد الموصل بـB0 سيضيء .. (أأكد على أن // وما بعدها في نفس السطر لن يلتفت إليهم البك ولن يحاول تنفيذهم فهو سيعتبرهم مجرد ملاحظات خاصة بك أنت . وبإعتبار أن الليد الموصل بـ B1 اسمه L1والليد الموصل بـ B2 اسمه L2 وهكذا ... ويمكنك طرح أي تسمية تشاءها ويمكنك أيضاً أن تكتب أي يشاء يجعلك تفهم أكثر أو يجعل الآخرين يفهموا الكود الخاص بك بسهولة .

وتلاحظ أيضاً في الكود السابق أنني جعلت قبل مجموعة من السطور مسافات وهو شيء للتنسيق ففي برنامجنا هناك ثلاثة أشكال للإضاء فجعلت الشكل الثاني قبله مسافات لأعلم أن الشكل الذي قبله هو الأول والذي بعده هو الثالث . كما يمكنني أن أضيف تعليق يخص هذا الأمر (طبعا بعد علامتي//) .

والآن لنرى ماذا سينفذ البك في السطور التسعة الأولى التي بعد كلمة :loop .



PORTB=0B00000001; delay_ms(500);

PORTB=0B00000011; delay_ms(500);

PORTB=0B00000111; delay_ms(500);

PORTB=0B00001111; delay_ms(500);

PORTB=0B00011111; delay_ms(500);

PORTB=0B00111111; delay_ms(500);

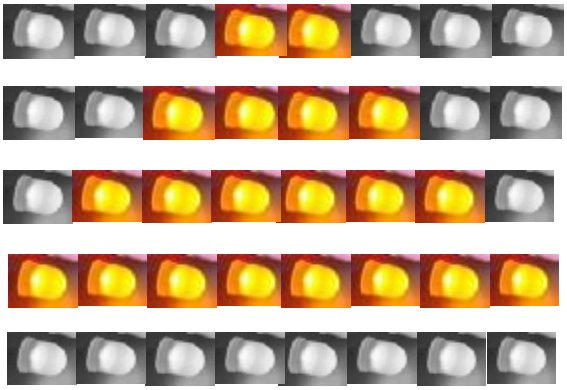
PORTB=0B01111111; delay_ms(500);

PORTB=0B11111111; delay_ms(500);

PORTB=0B00000000; delay_ms(500);

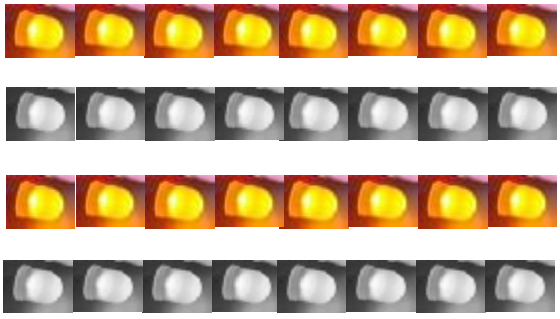
وهذا هو الشكل الأول .. والآن لنرى الشكل الثاني وهو عبارة عن الخمس سطور التي تلي السطور السابقة .

```
PORTB=0B00011000; delay_ms(500);
PORTB=0B00111100; delay_ms(500);
PORTB=0B01111110; delay_ms(500);
PORTB=0B11111111; delay_ms(500);
PORTB=0B00000000; delay_ms(500);
```



أما الشكل الثالث وهو الأربعة سطور التي تلي السطور السابقة فهو كما يلي .

```
PORTB=0B11111111; delay_ms(500);
PORTB=0B00000000; delay_ms(500);
PORTB=0B11111111; delay_ms(500);
PORTB=0B00000000; delay_ms(500);
```



بعد كتابة الكود وترجمته نرجع مرة أخرى للدائرة التي رسمناها في برنامج بروتس ونضغط على جسم البك ضغطتين متتاليتين ونحدد ملف الهيكس وكذلك التردد ... وبعد ذلك نشغل الدائرة ونرى هل ستعمل بالشكل الذي تخيلناه أم لا ؟

توجد طريقة أخرى لكتابة تعليق وهي أن نكتب /* ثم نكتب الكلام أو التعليق الذي لا نريد أن ينفذه البك ويمكن أن يكون هذا الكلام عبارة عن عدة أسطر ثم بعد ذلك نكتب */ كما يوضح الشكل التالي :

```
/* program : led flasher

   Programmed by : Ahmad samir fayed

*/
```

فيمكن جعل التعليق كمقدمة للمشروع تشرح وظيفته وملاحظات التوصيل واسم المبرمج ونحوه .

تابع الفيديو المرفق .

تابع الفيديو المرفق الخاص بهذه التجربة .

أرجو أن تكون قد استفدت واستمتعت خلال هذه التجربة .. وفي التجربة القادمة نكمل رحلتنا في تعلم البك وسنتعرف على مزيد من الأوامر البرمجية .

التجربة (٨)

في هذه التجربة المطلوب منا عمل عداد ثنائي (بايناري Binary) أتوماتيكي باستخدام الليدات.

هل تعرف ترتيب الأعداد بالباياناري؟؟ ... من المؤكد أنك تعرف العد بالعشري وهو ١ ٢ ٣ ٤ ٥ ٦ .... إلخ لكن بالباياناري (الثنائي) يكون كالتالي :

00000000

00000001

00000010

00000011

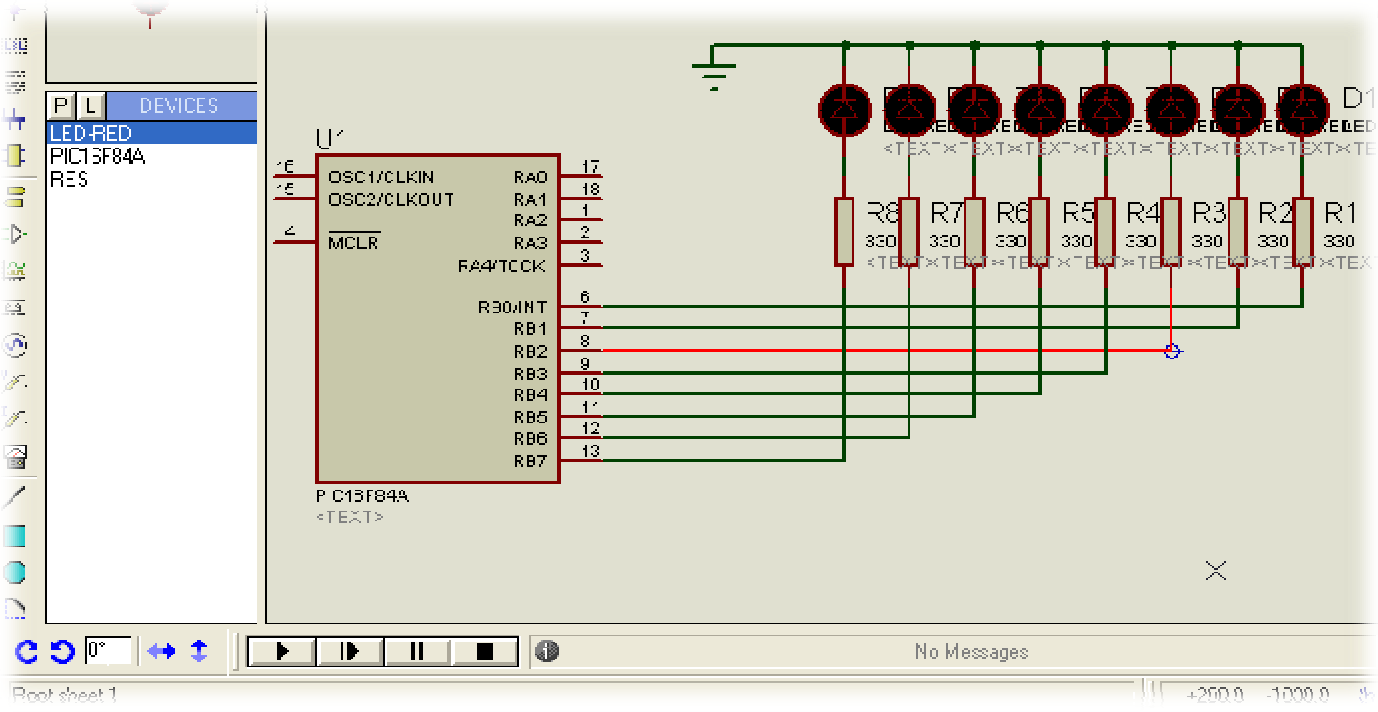
00000100

.....

.....

11111111

العداد المطلوب يتكون من ثمانية ليدات (وبالتالي ثمانية بت) يمكنه العد من صفر إلى ٢٥٥ حيث أن الرقم 11111111 (بالباياناري أو الثنائي) يساوي ٢٥٥ بالعشري. وبالتالي نستخدم نفس رسمة المشروع السابق



نقوم بإنشاء مشروع جديد ونكتب فيه الكود الذي سيجعل البك يقوم بعمل هذا العداد المطلوب .

```
void main()
{ TRISB=0;
PORTB=0;delay_ms(1000);
Loop:
        PORTB++;
        Delay_ms(1000);
Goto loop;
}
```

والآن لنشرح الكود السابق ::

كما نعلم سنوظف جميع الرجول على أنها خرج عن طريق الأمر ;TRISB=0 ليتسنى لنا بعد ذلك إضاءة الليدات . ثم نجعل جميع الليدات مطفأة عن طريق الأمر ;PORTB=0 وبهذه الطريقة تأكدنا أن البرنامج سيبدأ العد من صفر (حيث أن جميع الليدات مطفئة) . ستلاحظ الأمر الجديد وهو ;++portb هذا الأمر وظيفته أنه يضيف واحد (يجمع واحد) على القيمة السابقة لـ portb... لنحلل الأمر أكثر وننظر ماذا سينفذ البك خطوة خطوة ...

الأمر ;portb=0 سيجعل جميع الليدات مطفئة . إذن القيمة الحالية التي تظهرها الليدات هي

00000000 (حيث صفر يشير إليه الليد المنطفئ ، و واحد يشير إليه الليد المضيء)

وبعد ذلك سيتم الانتظار لمدة ثانية ثم يتم تنفيذ الأمر ;++portb والذي سيزيد قيمة portb السابقة (التي كانت 00000000) سيزيدها بمقدار واحد وبالتالي القيمة التي ستظهر على الليدات هي

00000001 (ليد واحد فقط هو المضيء وباقي الليدات مطفئة).

وبعد ذلك سينتظر البك لمدة ثانية والأمر ;goto loop سيجعله ينفذ الأوامر من جديد مرة أخرى أي سيزيد من قيمة portb السابقة (التي كانت 00000001) سيزيدها بمقدار واحد وبالتالي القيمة الجديدة التي ستظهر على الليدات ستكون

00000010

ثم ينتظر لمدة ثانية ويزيد قيمة portb مرة ثالثة ويجعلها

00000011

وهكذا باستمرار سيستمر بالعد كل مرة يزيد واحد هذا طبعا بالنظام الثنائي (binary) ...

لعلك لاحظت أهمية الأمر PORTB++ فبدونه كنا سنكتب العديد من الأسطر فبدون هذا الأمر قد يخطر على البال أن نكتب مثلا الأوامر التالية : (كنا سنكتب حوالي ٥٠٠ سطر) :

```
Loop:
PORTB=0B00000000;
Delay_ms(1000);
PORTB=0B00000001;
Delay_ms(1000);
PORTB=0B00000010;
Delay_ms(1000);
            ……..
وهكذا إلى ……..
PORTB=0B11111111;
Delay_ms(1000);
Goto loop;
```

فعلاً هذا الأمر وغيره من الأوامر التي تجري بعض العمليات الحسابية مهم جدا .. وبهذه المناسبة أأكد على معلومة مهمة وهي أن قيمة PORTB سواءا كتبناها بالنظام الثنائي أو بالنظام العشري فإنها ستتحول للنظام الثنائي لتظهر على الليدات بنفس الطريقة .. فلو كتبنا الأمر PORTB=3; فهو نفسه لو كتبنا PORTB=0B00000011; .. وأعتقد أن هذه المعلومة تعرفها جيدا .

التجربة (٩)

في هذه التجربة سنصمم عداد ثنائي تصاعدي يعد من صفر إلى 9 وبعد ذلك يبدأ من جديد يعد من واحد إلى عشرة وهكذا .

في هذه الدائرة لن نحتاج إلا لأربعة ليدات فقط سنوصلهم بـ b3,b2,b1,b0 لأن الرقم تسعة عبارة عن 1001 بالنظام الثنائي .

وسنكتب الأوامر التالية .

```
void main()
{   TRISB=0;
    PORTB=0; delay_ms(1000);
while(1){
    PORTB++;
    delay_ms(1000);
    if(portb==9) {portb=0; delay_ms(1000);}
            }
}
```

ليعمل البرنامج باستمرار (لتكرار تنفيذ الأوامر باستمرار) نستخدم loop و goto loop أو نستخدم while(1) كما ذكرت ذلك في التجربة (٥) .

وستلاحظ أن هذا البرنامج يشبه تماما البرنامج السابق ولكننا استخدمنا جملة if لتختبر هل وصلت قيمة portb إلى تسعة... فإذا حدث ذلك (تحقق الشرط) سيجعل البك قيمة portb بصفر ليبدأ العد من جديد من الصفر إلى أن يصل لتسعة وهكذا ...

ولمزيد من المراجعة ولمزيد من الفهم سأشرح طريقة ما سينفذه البك مرة أخرى بالتفصيل .

الآن في بداية البرنامج سيتم توظيف جميع الأطراف من b0 إلى b7 على أنها خرج عن طريق الأمر trisb=0; وبعد ذلك سيتم التأكد أن جميع الليدات مطفئة عن طريق الأمر portb=0; ثم ينتظر البك لمدة ثانية . وبعد ذلك ينتقل البك للأوامر التي ستتكرر باستمرار وهي الموجودة بين القوسين الخاصين بالأمر while(1) ففي المرة الأولى لتنفيذ هذه الأوامر سيضيء أول ليد والموصل بـ b0 وذلك عن طريق الأمر portb++ والذي يزيد من قيمة portb السابقة والتي كانت بصفر فسيجعلها بواحد وبالتالي يضيء الليد الموصل بـ b0 ثم ينتظر البك لمدة ثانية عن طريق الأمر delay_ms(1000); وبعد ذلك يختبر الشرط الخاص بجملة if وهو (هل قيمة portb الآن تساوي ٩) طبعا هنا لن يتحقق الشرط لأن قيمة Portb الآن تساوي واحد وبما أن الشرط لم يتحقق إذن لن يتم تنفيذ الأوامر التي بين القوسين الخاصين بجملة if .. بعد ذلك يعيد البك تنفيذ الأوامر الخاصة بـ while مرة أخرى فيزيد من قيمة portb ويجعلها تساوي ٢ (أي 10 بالنظام الثنائي) وبالتالي سيضيء الليد الموصل بـ b1 فقط ثم ينتظر البك لمدة ثانية ثم يختبر الشرط الخاص بجملة if ولن يتحقق الشرط فيعود البك مرة أخرى لينفذ الأوامر الخاصة بـ while أي سيزيد قيمة portb مرة أخرى وسيجعل قيمتها بثلاثة أي (11 بالثنائي) بعد ذلك ينتظر ثانية ثم يختبر الشرط فلن يتحقق ... وهكذا إلى أن تصل قيمة portb إلى ٩ (1001بالثنائي) فحينها عندما يختبر البك الشرط الخاص بجملة if فيجده قد تحقق وبالتالي سينفذ الأوامر الخاصة بجملة If فأول أمر سينفذه هو portb=0; الذي سيجعل قيمة portb بصفر ثم ينتظر البك ثانية عن طريق الأمر delay_ms(1000); وبعد ذلك يعود البك مرة أخرى لتنفيذ الأوامر الخاصة بـ while فينفذ الأمر الأول الذي يزيد قيمة portb وسيجعلها بواحد ثم ينتظر ثم يختبر الشرط ... وهكذا .

ملحوظة : يمكننا كتابة الكود السابق بشكل آخر باستخدام جملة for للتكرار و دون استخدام جملة if

```
void main(){
char x;
 TRISB=0; PORTB=0;
 delay_ms(1000);
 while(1){
for(x=0;x<9;x++){ PORTB++; delay_ms(1000);}
portb=0;delay_ms(1000);
            }
}
```

في هذه المرة سنكرر تنفيذ الأوامر التي تزيد قيمة portb تسع مرات عن طريق جملة for وبعد ذلك أي بعد تنفيذ جملة for سيجعل البك قيمة portb بصفر ثم ينتظر ثم يكرر تنفيذ جملة for مرة أخر التي بدورها ستقوم بالعد تسع مرات ثم سيجعل البك قيمة portb بصفر مرة أخرى وهكذا ...

ولكن أيهما أصح الكود الأول أم الكود الثاني ؟؟

كلاهما صحيح .. وعندما نقارن بين أي كود وآخر فغالبا ما يكون الأفضل هو الأقل في عدد الأوامر والأقل في عدد المتغيرات المستخدمة وسنتعرف أكثر على المتغيرات بعد أن نقوم بالتجربة القادمة إن شاء الله . فقط وضعت لك طريقتين مختلفتين لكتابة الكود لكي تعلم أن لديك العديد من الاختيارات وهذا يعتمد على فكرك .لذا ليس من العجيب أن تجد نفس البرنامج يكتبه عدة أشخاص بطرق مختلفة .

التجربة (١٠)

في هذه التجربة سنستخدم المتغيرات لنتعرف أكثر على مدى أهميتها . فالمطلوب من هذه التجربة هو عرض الرقم صفر ثم الرقم ٣ ثم الرقم ٦ ثم ٩ ثم ١٢ حتى ٣٠ (جدول الضرب للرقم ٣) ولكن بالنظام الثنائي أي أن الليدات هي التي ستعبر عن رقم الليد. فالليد المضيء يعني ١ بالثنائي والليد المنطفئ يعني ٠ بالثنائي أي أن هذا البرنامج عبارة عن عداد ولكن طريقة عده تكون وفق جدول الضرب للرقم ثلاثة أي سيكون كتالي

سنكتب الكود التالي :

```
void main()                                                    {
char number; char x;
TRISB=0; PORTB=0;delay_ms(1000);
while (1)  { for (x=1;x<=10;x++)     {
                number=3*x;
                portb=number;
                delay_ms(1000);          }
portb=0; delay_ms(1000);                        }                    }
```

طبعا من الأفضل عدم تنسيق الكود بهذا الشكل لكي يسهل فهمه .. (ولكن أحيانا نضطر لذلك بسبب حدود الورقة) .. ولكن إذا اضطررنا لذلك على الأقل نحاول أن نضع الأقواس الخاصة بنفس الأمر تحت بعض مباشرة بقدر الإمكان .

في السطر الرابع من البرنامج استخدمنا الأمر while(1) وبالتالي سيكون ما بداخل القوسين الخاصين بهذا الأمر يتكرر تنفيذهم باستمرار . وبعد ذلك استخدمنا جملة for والتي تبدأ العد من x=1 وحتى x=10 أي أن الأوامر الخاصة بجملة for ستتكرر ١٠ مرات هذه الأوامر التي ستتكرر عشر مرات هي :

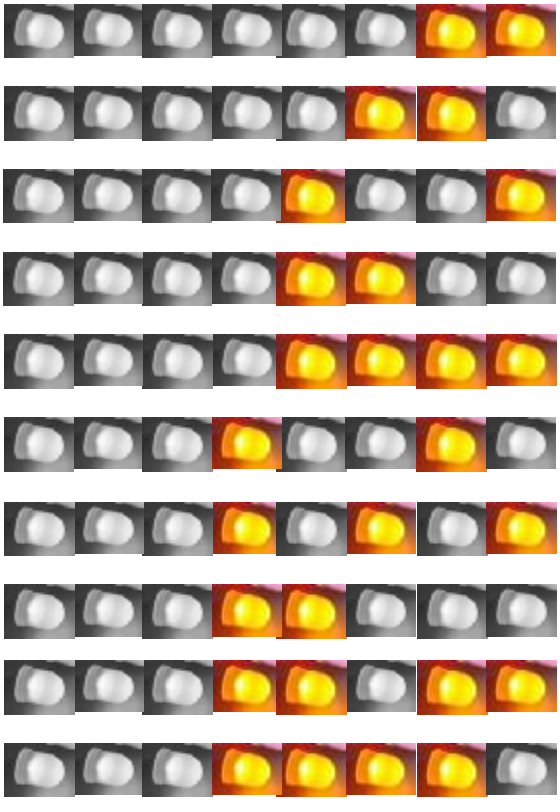
```
number=3*x;
portb=number;
delay_ms(1000);
```

ففي أول مرة تنفذ فيها تلك الأوامر ستكون قيمة x=1 وبالتالي ستكون قيمة المتغير number تساوي نتيجة العملية التالية : (٣ ضرب واحد) (أي ثلاثة) وهذا من خلال الأمر number=3*x; بعد ذلك سينفذ الأمر portb=number; والذي سيجعل الرقم ثلاثة يظهر على الليدات (بالنظام الثنائي) أي أن كل من الليد الموصل بـ b0 و b1 سيضيئا ... وهي نفس النتيجة التي نحصل عليها لو كتبنا الأمر portb=0b00000011; أو portb=3;

ثم ينتظر البك لمدة ثانية من خلال الأمر delay_ms(1000); ثم يعيد تنفيذ الأوامر مرة أخرى وذلك بجعل قيمة x=2 . وهذا يؤدي إلى أن قيمة المتغير number ستصبح 3X2 أي ٦ من خلال الأمر number=3*x; (هنا x=2) ثم تظهر القيمة ٦ على portb من خلال الأمر portb=number; ثم ينتظر البك لمدة ثانية ثم يعيد تنفيذ الأوامر مرة أخرى ولكن مع جعل x=3 .... وهكذا إلى أن تصبح قيمة x تساوي 10 فينفذ الأوامر مرة أخرى ولن يجعل قيمة x بـ ١١ لأن الشرط يخبرنا أن x<=10 . وبهذا تنتهي جملة for ثم ينفذ البك الأمر الذي يلي جملة for وهو portb=0 أي سيطفئ جميع الليدات ثم ينتظر لمدة ثانية من خلال الأمر delay_ms(1000) ثم يعيد تنفيذ جملة for مرة أخرى من جديد ثم ينفذا ما بعد جملة for ثم يعيد تنفيذ جملة for من جديد ... وهكذا ، وبالتالي سنرى أن جدول الضرب للعدد ثلاثة يظهر على الليدات ويتكرر باستمرار . كما يوضح ذلك الشكل التالي في الصفحة التالية :

في كل مرة ستضيء الليدات بشكل معين ثم ينتظر البك لمدة ثانية ثم تضيء الليدات نفسها بشكل آخر

وهكذا...



ملحوظة :: هل تعلم أن portb يتكون من ٨ بت فهو يشبه المتغير char في هذه النقطة لذلك يمكن اعتبار PORTB متغير لكنه يمتاز بخاصية إضافية وهي أن قيمته تظهر مباشرة على الأطراف من B0 إلى B7 .

ولهذا فإن البرنامج السابق كان من الممكن كتابته دون استخدام المتغير number . بالشكل التالي .

```
void main()
{char x;
TRISB=0; PORTB=0;delay_ms(1000);
While (1)  { for (x=1;x<=10;x++)  { portb=3*x;    delay_ms(1000); }
              portb=0; delay_ms(1000);
             }
}
```

مرة أخرى أقول أن portb متغير طوله ٨ بت لكن يمتاز بميزة إضافية عن بقية المتغيرات العادية وهي إخراج جهد معين(٥فولت أو سالب البطارية) على أي طرف من B0 إلى B7 .

المتغيرات ماذا تعني ؟

هل تعرفت بعد على معنى " متغير" ؟؟ إن متغير هذه تعني أنه مكان نحجزه في الذاكرة (ذاكرة البك) نضع فيه قيمة معينة هذه القيمة يمكن أن تتغير لذلك نسميه متغير . والمكان الذي نحجزه في الذاكرة نستفيد منه في عمليات كثيرة جداً ، فمنها مثلا لاستخدام جملة for نقوم بإنشاء متغير حيث نضع فيه قيمة معينة ونكرر تنفيذ مجموعة من الأوامر وعند وصول هذا المتغير لقيمة معينة لن يتم تكرار الأوامر مرة أخرى. أيضا نستخدم المتغيرات في عمليات حسابية كثيرة مثل الجمع والطرح والضرب والقسمة ... إلخ .

لقد ذكرت منذ قليل أن المتغير هو مكان نحجزه في الذاكرة .. ولكن ماهو قدر (أو طول) هذا المكان ؟؟

في الواقع يمكننا أن نحجز ٨ بت (8 bits) أو ما يسمى بـ بايت byte (1 byte = 8 bits) ويمكننا أيضا أن نحجز 16بت أي اثنين بايت أو يمكننا حجز ٤ بايت أي ٣٢ بت .. كل ذلك كما نريد ؟ نعم ،، عن طريق تحديد نوع المتغير ، فيمكننا تحديد طوله بجعله ٨ بت مثلاً عن طريق جعل المتغير من النوع char أو unsigned short كما يمكننا جعله ١٦بت عن طريق جعل نوعه int مثلا . وإليك الآن أنواع المتغيرات وقيمة (طول) كل متغير :

النوع	الحجم بالبايت (byte)	المدى
char	1	0 .. 255
signed char	1	- 128 .. 127
short	1	- 128 .. 127
unsigned short	1	0 .. 255
int	2	-32768 .. 32767
unsigned	2	0 .. 65535
long	4	-2147483648 .. 2147483647
unsigned long	4	0 .. 4294967295

الأنواع التي تسمح بالكسور

Type	Size in bytes	Range
float	4	$\pm 1.17549435082 * 10^{-38}$ .. $\pm 6.80564774407 * 10^{38}$
double	4	$\pm 1.17549435082 * 10^{-38}$ .. $\pm 6.80564774407 * 10^{38}$
long double	4	$\pm 1.17549435082 * 10^{-38}$ .. $\pm 6.80564774407 * 10^{38}$

كما قلت أن المتغير هذا مكان يحجز في الذاكرة ويمكننا أن نحدد طول أو حجم هذا المكان بتحديد نوع المتغير فكما يتضح من الجدول يوجد متغيرات حجمها واحد بايت وأخرى حجمها ٢ بايت وأخرى أربعة .

فمثلا المتغير الذي من النوع char حجمه واحد بايت أي ٨ بت . وبالتالي أقصى قيمة له نحصل عليها إذا كانت قيمته كالتالي 11111111 .هذا بالنظام الثنائي وهو ما يكافئ 255 بالنظام العشري . أي أن مدى هذا المتغير الذي من النوع char من صفر إلى ٢٥٥ . وبمعنى آخر أن هذا المتغير قيمته ستكون صفر أو واحد أو اثنين أو ... ٢٥٥ أي أن قيمته محصورة بين الصفر إلى ٢٥٥ . لذلك نقول أن مداه من صفر إلى ٢٥٥.

في بعض العمليات الحسابية نحتاج إلى متغير قد تكون قيمته سالبة مثل المتغير signed char فسنجد أن مداه من سالب مئة وثمان وعشرون(128-)إلى موجب مئة وسبعة وعشرون (127) . أيضا هناك بعض العمليات الحسابية التي ستحتاج فيها إلى متغير من نوع مختلف يسمح بالنقطة العشرية (أو الكسور) بمعنى أنه لتخزين قيمة معينة ولتكن 32.66 ستحتاج لمتغير به خاصية العلامة العشرية وهو ماستجده في الجدول الثاني الذي عنوانه (الأنواع التي تسمح بالكسور) .

والآن بعد أن أخذنا لمحة سريعة على أنواع المتغيرات هيا بنا لنتعرف على الطريقة التي نكتب بها الكود الذي من خلاله يقوم البك بحجز مكان في الذاكرة لهذا المتغير أو ما نسميه بعملية تعريف المتغير .

لتعريف المتغير (لحجز مكان في ذاكرة البك) علينا ان نكتب نوع المتغير ثم نكتب اسمه ولنا حرية الاختيار في تحديد اسمه . فمثلا لتعريف متغير من النوع char واسم المتغير مثلا ahmad . فسنكتب الأمر التالي :

```
char ahmad;
```

طبعا لا تنسى علامة ; ولعلك تذكر عند استخدامنا لجملة for أننا كتبنا الأمر

```
char x;
```

فهذا الأمر معناه أننا حجزنا ٨ بت في ذاكرة البك ووضعنا اسم لهذا المكان الذي حجزناه وجعلنا اسمه x .

بقي أن تعرف أننا لو أردنا أن نعرف ثلاث متغيرات مثلا من نفس النوع ولتكن هذه المتغيرات هي x , y , z . فإنه يمكننا أن نقوم بذلك بطريقتين الطريقة الأولى كما يلي :

```
Char x;
Char y;
Char z;
```

والطريقة الثانية وهي الأسهل كما يلي :

```
Char x,y,z;
```

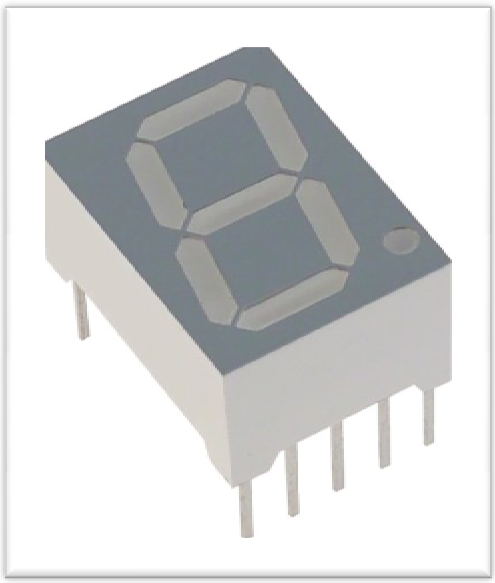
فقط نضع علامة الفاصلة , بين أسماء المتغيرات . ولكن هذا بشرط أن يكونوا جميعا من نفس النوع .

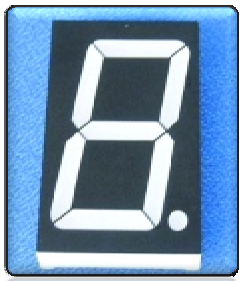
ملحوظة أخيرة : يجب أن يكون تعريف المتغير في بداية البرنامج فإما أن تكتبه قبل void main() أو أن تكتبه داخل الدالة الرئيسية ولكن في البداية أيضا . وعندما نشرح الدوال سنتطرق لهذه النقطة مرة أخرى إن شاء الله .

التجربة (١١)

لابد أنك تعلم أن كثير من الناس لا يعرف الترقيم الثنائي والكثير أيضا يجده صعبا في القراءة لذلك يفضل أن نستخدم في المشاريع التي نقدمها لهم الترقيم العشري بدلا من الثنائي .. ولكن كيف ذلك ؟؟ ببساطة سنستبدل الليدات بشيء آخر يسمى سفن سيجمنت . seven segment (7 segmetns) . وسنقوم في هذه التجربة بعمل عداد يقوم بالعد من صفر إلى تسعة ثم يكرر ذلك باستمرار ولكن بالنظام العشري .

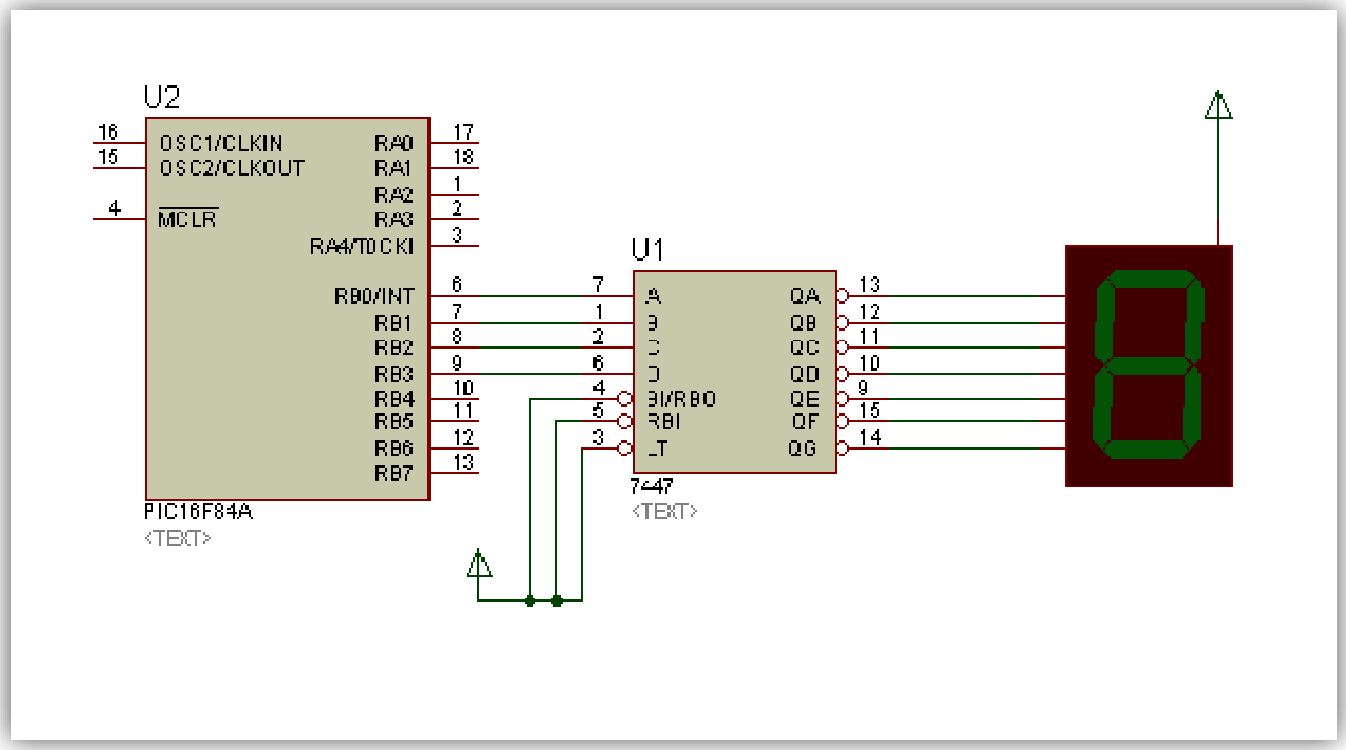
في هذه التجربة سنستخدم نفس الكود الذي استخدمناه في التجربة رقم ٩ ولكن سنقوم بعمل تغيير في الدائرة الالكترونية فقط . سنقوم بتنفيذ هذه التجربة ثم نشرح بالتفصيل فيما بعد في تجارب أخرى . ماهي السفن سيجمنت وتركيبها من الداخل ؟؟ كل ما يجب أن تعرفه في هذه التجربة هو شكل السفن سيجمنت و كيف سيتم توصيلها بالبك وسنتعرف في هذه التجربة على إحدة طرق التوصيل .





بعض أشكال السفن سيجمنت 7 segment

والآن لنرى كيف سنقوم بتوصيل الدائرة .



لاحظ أن السفن سجمنت من النوع (المصعد المشترك) common anode .

كما أن ترتيب الأطراف في دائرة المحاكاه ليس كالذي في الحقيقة لذا يجب الرجوع لـ datasheet .

لعلك تتسائل وتقول كيف ستعمل الدائرة بنفس الكود الذي يقوم بعمل عداد ثنائي .. ببساطة إن الدائرة المتكاملة 7447 هي المسؤولة عن هذه العملية حيث تقوم بتحويل الرقم الثنائي إلى ما يناظره على السفن سيجمنت .. وسنشرح ذلك بشكل أوسع بإذن الله فيما بعد .. فقط استمتع بمتابعة الفيديو المرفق وجرب الدائرة بنفسك على برنامج المحاكاه .

تحذير ..!! : لا تحاول تجربة هذه الدائرة في الحقيقة الآن إلا لو كانت السفن سيجمنت التي لديك تعمل بـ خمسة فولت .

التجربة (١٢)

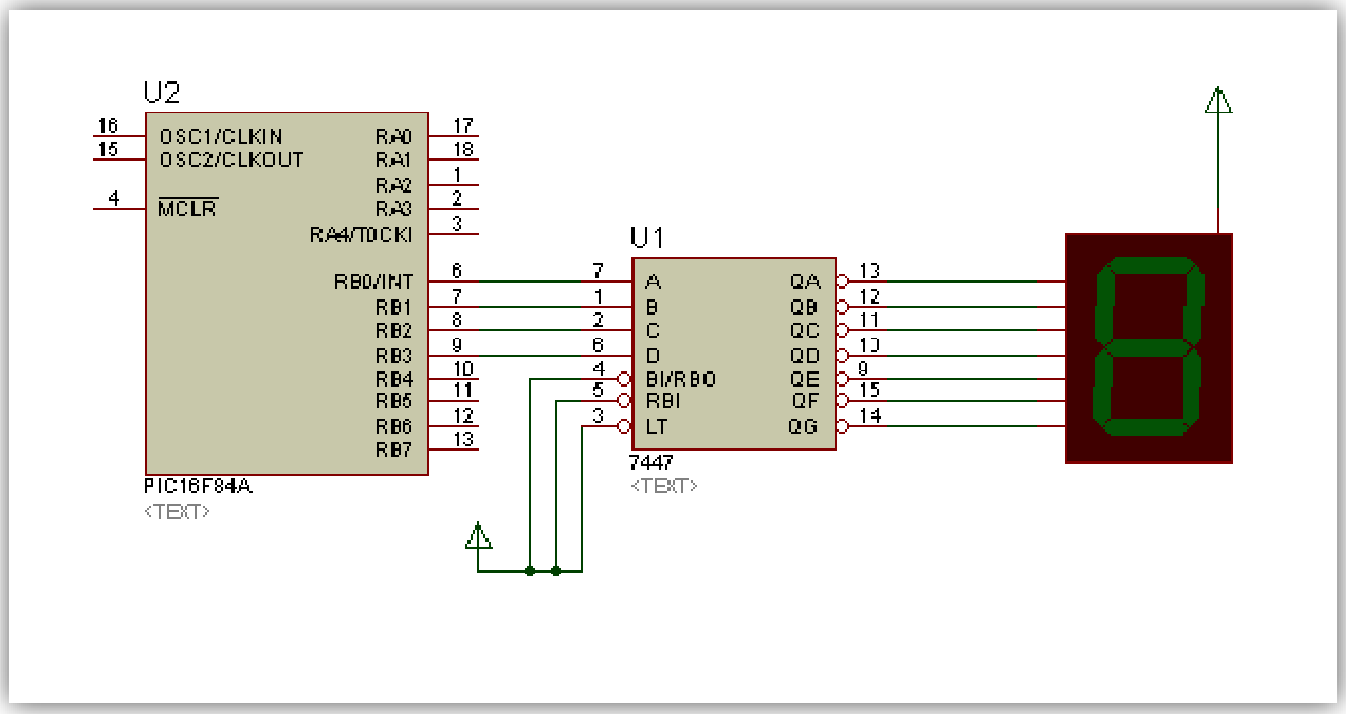
في هذه التجربة سنتعرف على مزيد من الأوامر الحسابية ففي هذه المرة سنقوم بعمل عداد تنازلي وليس تصاعدي يقوم بالعد من تسعة إلى صفر ويكرر ذلك باستمرار .. وهذا أيضا باستخدام السفن سيجمنت.

هل تتذكر أهم أمر في برنامج العداد التصاعدي .. إنه الأمر portb++ ، حيث أن هذا الأمر هو المسؤول عن عملية زيادة العدد الموجود في portb . الآن نحن نريد العكس فبالتالي سنستخدم أمر آخر مسؤول عن عملية النقصان(الطرح) في العدد الموجود في portb . انظر للكود التالي ولاحظ التغييرات .

```
void main()
{   TRISB=0;
    PORTB=9; delay_ms(1000);
while(1){
    PORTB--;
    delay_ms(1000);
    if(portb==0) {portb=9; delay_ms(1000);}
            }
}
```

لعلك لاحظت الأمر portb--; إنه ينظر للقيمة السابقة لـ portb وينقص(يطرح) منها واحد . في بداية البرنامج جعلت قيمة portb تساوي ٩ لكي يتم بداية العد منها ثم بعد ذلك ينتظر البك ثم يقوم بتنقيص أو تقليل العدد ٩ ويجعله ٨ عن طريق الأمر portb--; ثم ينتظر البك لمدة ثانية ويتم تكرار ذلك باستمرار حتى تصل قيمة portb إلى صفر حينها تتحق جملة if ويتم تنفيذ الأوامر التي بين القوسين الخاصين بجملة if وبدورهم سيتم بدء العد من جديد من الرقم ٩ . .. وهكذا .

نفس الدائرة التي في التجربة السابقة يمكن استخدامها هنا أيضا .



كما أخبرتك سابقا إن الكود الأفضل هو الأقل في عدد الأوامر .. وعند تأملي للكود السابق وجدت أنه يمكن تعديله ليصبح أفضل حيث لاحظت أنني استخدمت أمر الإنتظار ;(delay_ms(1000 ثلاث مرات لذلك تم تعديل الكود السابق إلى الشكل التالي : (مع العلم أن كلاهما سيؤدي نفس الوظيفة)

```
void main()
{   TRISB=0;
    PORTB=9;
while(1){   delay_ms(1000);
            PORTB--;
            if(portb==0) { delay_ms(1000); portb=9;}
         }
}
```

تأمل في الكود جيداً وانتبه لجميع التغيرات وخاصة ترتيب الأوامر وحاول تخيل ما سينفذه البك ..

في الكود السابق لقد أصبح الأمر ;delay_ms(1000) متكرر مرتين فقط وليس ثلاث مرات لذلك فالكود السابق أفضل من الكود الأول .

(تمرين) : ولكن هل يمكن أن نستخدم أمر الإنتظار مرة واحدة وليس مرتين ؟؟ .. فكر جيداً كيف سنفعل ذلك واعط لنفسك الوقت وإذا أردت أن تعرف الإجابة التي لدي انظر للصفحة التالية .

```
void main()
{char x;  TRISB=0;
portb=9;
while(1){
        for(x=10;x>0;x--)
        {
        delay_ms(1000);
        portb--;
        }
portb=9;
    }
}
```

تمرين جميل .. أليس كذلك ؟؟ . مع العلم أنه ربما توجد حلول أخرى كثيرة . والآن في هذا الكود استخدمنا متغير إضافي X وكما ذكرت لك سابقاً إن الكود الأفضل هو الأقل في عدد الأوامر وفي عدد المتغيرات المستخدمة . لذلك حاول أن تفكر في تعديل هذا الكود لكي لا نستخدم المتغير X . إذا أردت أن تعرف الإجابة التي أقترحها انظر للصفحة التالية ، ولكن قبل ذلك اعط لنفسك وقت للتفكير والمحاولة.

```
void main()
{  TRISB=0;
while(1){
                for(PORTB=9;PORTB>0;PORTB--)
                {
                 delay_ms(1000);
                 }
          }
}
```

ألم أخبرك سابقاً أن PORTB متغير لكن من نوع خاص .. لقد قمت باستغلال هذه المعلومة في هذا الكود . ويمكنك أن تلاحظ أنني لم أكتب في بداية البرنامج char portb مثلا لأن portb لا يحتاج إلى ذلك وهذا ينطبق أيضا على porta و ... إلخ .

لكنني عندما كتبت الكود السابق ظهر خطأ أثناء تنفيذ البرنامج وهو أنه بعد الوصول إلى الرقم ١ لن يظهر الرقم صفر وسيظهر الرقم تسعة مباشرة . لذلك فكرت أن أقوم بتعديل جملة for لتصبح كما يلي :

```
for(PORTB=9;PORTB>=0;PORTB--)
```

جرب فعل ذلك ستواجه مشكلة وهي أنه بعد ظهور الرقم صفر تبدأ السفن سيجمنت بإظهار أشياء غريبة غير الأرقام لذلك يجب أن نفكر في تعديل آخر ...

ولكن قبل ذلك ما هو سبب هذه المشكلة ؟ الآن تخيل معي ما سيحدث عند تنفيذ البك لجملة for . إن البك في بداية تنفيذه لجملة for سيعرض الرقم تسعة من خلال الأمر portb=9 ثم بعد ذلك ينفذ ما بداخل جملة for وهو أمر الإنتظار ثم بعد ذلك يقوم بتقليل القيمة تسعة حيث يظهر الرقم ٨ عن طريق الأمر Portb-- ثم يختبر الشرط وهو هل قيمة portb أكبر من أو يساوي صفر إذا كان الجواب نعم ، سيكرر الأوامر مرة أخرى وهو أمر الإنتظار في هذا المثال ثم يقوم البك بالتقليل مرة أخرى بإظهار الرقم ٧

عن طريق الأمر --Portb ثم يختبر الشرط فيجده تحقق فيقوم بالإنتظار ... وهكذا إلى أن تصبح قيمة portb تساوي صفر حينها يقوم البك بتنقيص(تقليل) هذه القيمة ولكن ماذا سيفعل هل سيجعل قيمة Portb تساوي سالب واحد ثم يختبر الشرط فيجده لم يتحقق فيخرج من جملة for ؟؟

في الواقع هذا لن يحدث .. لماذا ؟؟ .. لأن portb يشبه المتغير الذي من النوع char وهذا المتغير لا يسمح بالأرقام السالبة وعند رجوعك إلى الجدول الخاص بالمتغيرات ستلاحظ أن مداه من صفر إلى ٢٥٥ لذلك عندما ينقص البك من قيمة portb بعد أن كان بصفر سيصبح ٢٥٥ وليس سالب واحد . وهذا معناه أن البك لن ينتهي أبداً من تنفيذ جملة for لأن الشرط سيظل متحققاً دائماً لأنه بعد أن تصل القيمة إلى صفر تصبح بعد ذلك بـ ٢٥٥ والشرط كان يقول أن قيمة portb أكبر من أو يساوي صفر وطبعا 255 أكبر من الصفر لذلك ستستمر جملة for وتنقص قيمة portb وتصبح ٢٥٤ إلى أن تصبح صفر ثم تصبح 255 وهكذا. إن سبب ظهور الأشياء الغريبة (وليس أرقام) على السفن سيجمنت أن الدائرة المتكاملة 7447 مهيأة أن تستقبل الأرقام الثنائية من صفر إلى تسعة لكي تستطيع عرضها على السفن سيجمنت وفيما عدا هذه الأرقام فإنها لن تقوم بالعرض على السفن سيجمنت بالشكل الصحيح . وهذا يحدث بمجرد أن تصبح قيمة portb بـ ٢٥٥ .

والآن عرفنا سبب المشكلة ونريد حلاً ...فكر في الحل ، وإذا أردت أن تعرف إجابتي المقترحة انظر للصفحة التالية.

```
void main()
{  TRISB=0;
while(1){
        for(PORTB=9;PORTB>=0;PORTB--)
        {
        delay_ms(1000);
       if(portb==0) portb=10;
        }
      }
}
```

وبهذا لا يوجد أي مشكلة ، فالكود ليس به تكرار لكتابة أي أمر ولم نستخدم فيه متغير إضافي وليس به أخطاء ويؤدي الوظيفة المطلوبة .

فعندما تصل قيمة portb إلى صفر وقبل أن يجعل البك هذه القيمة بـ 255 ستقوم جملة if بجعل portb=10 ولأن الأمر الذي سينفذ بعد ذلك هو portb-- وليس بينهم أمر انتظار لن نلاحظ أي تغيير ملفت و portb ستصبح بـ ٩ في منتهى السرعة. (أقصد لن تظهر السفن سيجمنت أشياء غريبة حيث أن الرقم ١٠ خارج نطاق الأرقام من صفر إلى تسعة لأن البك سينفذ الأوامر بسرعة أما لو كان هناك أمر انتظار بعد الأمر portb=10 كان سيظهر شيء غريب على السفن سيجمنت) .

انظر للكود الذي كتبناه من قبل ولم نجعل هناك علامة تساوي فيه .

```
void main()
{  TRISB=0;
while(1){
                for(PORTB=9;PORTB>0;PORTB--)
                {
                 delay_ms(1000);
                 }
          }
}
```

كانت مشكلة هذا الكود أنه يقوم بالعد من ٩ إلى واحد ثم يظهر الرقم ٩ مباشرة دون المرور بالرقم صفر لهذا قمنا بإضافة علامة تساوي ثم بعد ذلك وضعنا جملة if . ولكن هل تعرف ما هو السبب في أن الرقم صفر لم يظهر ؟؟ لنتأمل ما سيحدث ونتخيل كيف سينفذ البك جملة for في هذا الكود .

سيظهر البك الرقم تسعة عن طريق الأمر portb=9 ثم يقوم بتنفيذ ما بين القوسين وهو الانتظار لمدة ثانية ثم ينقص القيمة ويجعلها ٨ ثم يختبر الشرط فيجده متحققا فيكرر الانتظار مرة أخرى وهكذا إلى أن تصبح قيمة portb تساوي ١ حينها سيقوم بإنقاص هذه القيمة ويجعلها بصفر وبسرعة يختبر الشرط فيجده لم يتحقق فيخرج من جملة for ولا يجد أوامر أخرى وبما أن جملة for داخل جملة while سينفذ البك جملة for مرة أخرى كل ذلك في منتهى السرعة وجملة for ستجعل قيمة portb بتسعة . لهذا لن تلاحظ أبداً أن الرقم صفر ظهر على السفن سيجمنت . لذلك يمكن حل المشكلة السابقة بطريقة أخرى وهي بإضافة أمر إنتظار بعد جملة for كما في الكود التالي :

```
void main()
{  TRISB=0;
while(1){
                for(PORTB=9;PORTB>0;PORTB--)
                {
                 delay_ms(1000);
                 }
delay_ms(1000);
          }                    }
```

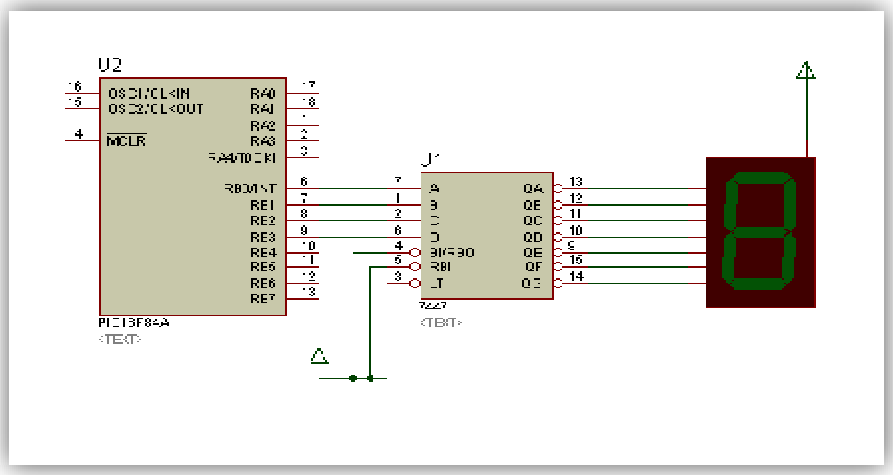
وبهذا بعد أن تصبح قيمة portb بصفر سيخرج البك من جملة for وينفذ الأمر الذي يليها وهو الانتظار لمدة ثانية فيظهر الرقم صفر على السفن سيجمنت ثم يتم تكرار تنفيذ هذه الأوامر مرة أخرى لأنهم ضمن جملة while(1) .

ولكن طبعا هذا الحل لا نريده لأننا في الأساس كنا نريد استخدام أمر الانتظار مرة واحدة فقط .

عزيزي القارئ : إن الكلام السابق والحلول هذه كان من الممكن الاستغناء عنها فلن تحدث مشكلة أثناء تنفيذ هذه التجربة عند تطبيقنا لأول كود كتبناه ففي مثل هذه المشاريع البسيطة ليست هذه مشكلة إطلاقاً . ولكن في بعض المشاريع قد تحتاج لتغيير الكود بسبب قلة الذاكرة ونحوه لهذا كانت هذه محاولة مني لتوسيع الأفق البرمجي لديك أتمنى من الله أن توفق في حل مشاكلك البرمجية القادمة بكل مهارة وأتمنى أن تكون استفدت من هذا الدرس واستمتعت به و أن تكون قوة فهمك لما سينفذه البك خطوة خطوة قد ازدادت لديك .. وتأكد أنك إذا فهمت هذا الدرس جيداً فأنت الآن تعد ضمن قليل من المبرمجين في العالم الذين يمكلون هذه المهارة الاحتراف. فهذه نقطة إحترافية وإلى مزيد من الإحتراف أكثر وأكثر إن شاء الله فتابع الكتاب لعله يساعدك في طريقك للإحتراف بإذن الله . والله سبحانه وتعالى هو الموفق وهو مسبب كل نجاح وتفوق واحتراف . فاللهم ارزقنا وإياكم العلم النافع .

التجربة (١٣)

في هذه التجربة سنقوم بعمل عداد تصاعدي يقوم بعد الأعداد الزوجية فقط . وسأقوم بشرح بعض الأوامر الحسابية .



نفس الدائرة السابقة .

من المعلوم أن الفرق بين أي عدد زوجي وآخر يساوي اثنين . بمعنى أن العدد أربعة مثلاً عدد زوجي والعدد الزوجي الذي يليه هو ستة (٤+٢ يساوي ٦) لذلك إذا بدأنا من اثنين وأضفنا كل مرة الرقم ٢ إلى العدد السابق سنحصل بذلك على عداد للأعداد الزوجية . إذن يمكن كتابة الكود التالي ليؤدي هذه الوظيفة :

```
void main()
{  TRISB=0;portb=2;
while(1){
        delay_ms(1000);
        portb++;
        portb++;
if (portb==8){ delay_ms(1000);portb=2;}
         }
}
```

بما أن الأمر portb++ يزيد من القيمة السابقة بمقدار واحد إذن فإن كتابة هذا الأمر مرتين يجعل البك يزيد من القيمة السابقة اثنين . وهذا هو المطلوب .

ولكن ماذا لو أردنا أن نزيد القيمة خمسة أو مئة أو أي رقم كبير .. إن عملية تكرار الكود في هذه الحالة يعد من الغباء البرمجي كما أسميه أنا . لذلك من المؤكد أن هناك حلول أخرى أو أوامر أخرى تساعد على هذا الأمر . قد يخطر بذهنك فكرة أخرى وهي استخدام جملة for وداخل جملة for نكتب الأوامر التي نريد تكرارها وهذه فكرة جيدة . ولكن هناك حل آخر أسهل وأفضل .

الآن عندما نريد أن نجمع الرقم ٢ على القيمة السابقة لـ Portb فإنه يمكن فعل ذلك بسهولة بكتابة الأمر:

```
portb=portb+2;
```

هذا الأمر سينظر للقيمة السابقة لـ Portb ويضيف إليها ٢ . وبنفس الطريقة يمكن كتابة أمر يضيف على القيمة السابقة لـ Portb الرقم ٩٠ .

```
portb=portb+٩٠;
```

وبهذه الطريقة يمكن كتابة الكود المطلوب بالشكل التالي :

```
void main()
{  TRISB=0;portb=2;
while(1){
        delay_ms(1000);
        portb=portb+2;
        if (portb==8){ delay_ms(1000);portb=2;}
         }
}
```

وهنا يجب التنبيه إلى نقطة مهمة جداً وهي أننا في حالة كتابتنا لمثل هذه المعادلات فإن الطرف الأيسر من المعادلة هو الذي نريد تغيير قيمته أما الطرف الأيمن فلن يحدث له أي تغيير .

وسأضرب لك هذا المثال : نفرض أن porta=3; ونريد أن نجعل قيمة portb تساوي خمسة فإننا سنكتب الكود التالي :

```
portb=porta+2;
```

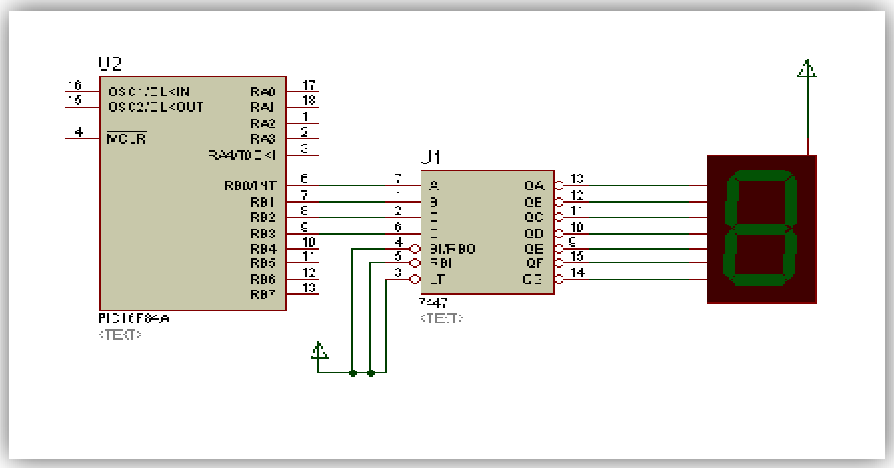
الطرف الأيسر هو الذي سيتغير أي أن portb ستصبح قيمته تساوي قيمة porta مضافاً إليها ٢ . وبالتالي ستصبح قيمة portb بخمسة . أما الطرف الأيمن فلن يحدث له أي تأثير بمعنى أن المتغير أو المسجل Porta يظل كما هو أي لازالت قيمته تساوي ٣ . ولكن عندما نكتب portb=portb+2; هنا نجد أن الطرف الأيمن والأيسر كلاهما بهما Portb فهنا المقصود من portb التي على اليمين أنها القيمة السابقة لـ Portb وأما الطرف الأيسر فإن portb هي القيمة الجديدة التي ستكون بإجراء تنفيذ العملية الحسابية التي في الطرف الأيمن (وهي القيمة السابقة لـ Portb + ٢) .

في التجارب السابقة استخدمنا الأمر portb++ هذا الأمر يكافئ الأمر portb=portb+1 . فكلاهما يؤدي نفس الوظيفة وهي إضافة أو جمع واحد على القيمة السابقة لـ portb .

أيضاً هذه العملية يمكن تطبيقها على أي متغير فمثلا لو كان لدينا المتغير x . يمكننا أن نكتب x=x+6 مثلا .

التجربة (١٤)

في هذه التجربة المطلوب عمل عداد تنازلي للأعداد الزوجية يبدأ العد من ٨ وينتهي بـ ٢ .



نفس الدائرة السابقة

في التجربة السابقة كتبنا الأمر Portb=portb+2 . هنا سنعكس هذه العملية وسنكتب

portb=portb-2 وسنقوم بتعديلات بسيطة في الأرقام .. اترك لك اكتشافها .

```
void main()
{  TRISB=0;portb=٨;
while(1){
        delay_ms(1000);
        portb=portb-2;
        if (portb==٢){ delay_ms(1000);portb=٨;}
         }
}
```

بقي أن تعرف أن الأمر portb=portb-2 يمكن كتابته بشكل آخر وهو

```
portb-=2 ;
```

والأمر portb=portb+2 يمكن كتباته بشكل آخر أيضا

```
Portb+=2;
```

أيضا لو أردنا أن نضرب القيمة السابقة في ٥ مثلا نكتب الأمر portb=portb*5; يمكن أيضا أن يكتب :

```
Portb*=5
```

وبهذا نكون قد تعلمنا بعض الأوامر الحسابية المهمة جدا في العديد من التطبيقات .

ولنفترض الآن أن لدينا مشروع معين وداخل هذا الكود نريد القيام بعملية حسابية معينة ونضع النتيجة في متغير اسمه mm مثلاً .

ونفترض أن هذه العملية الحسابية هي ٣٥٠ زائد ١٢ مطروحا منها قيمة المتغير yy كل ذلك مقسوما على ١٠ فإننا نكتب الأمر التالي :

```
mm=(350+12-yy)/10;
```

لعلك لاحظت وجود الأقواس فهي تعني أن ما بداخلها من عمليات حسابية سيتم تنفيذه أولا ثم بعد ذلك نقسم هذا الناتج على ١٠ أما لو كتبنا الأمر التالي :

```
mm=350+12-yy/10;
```

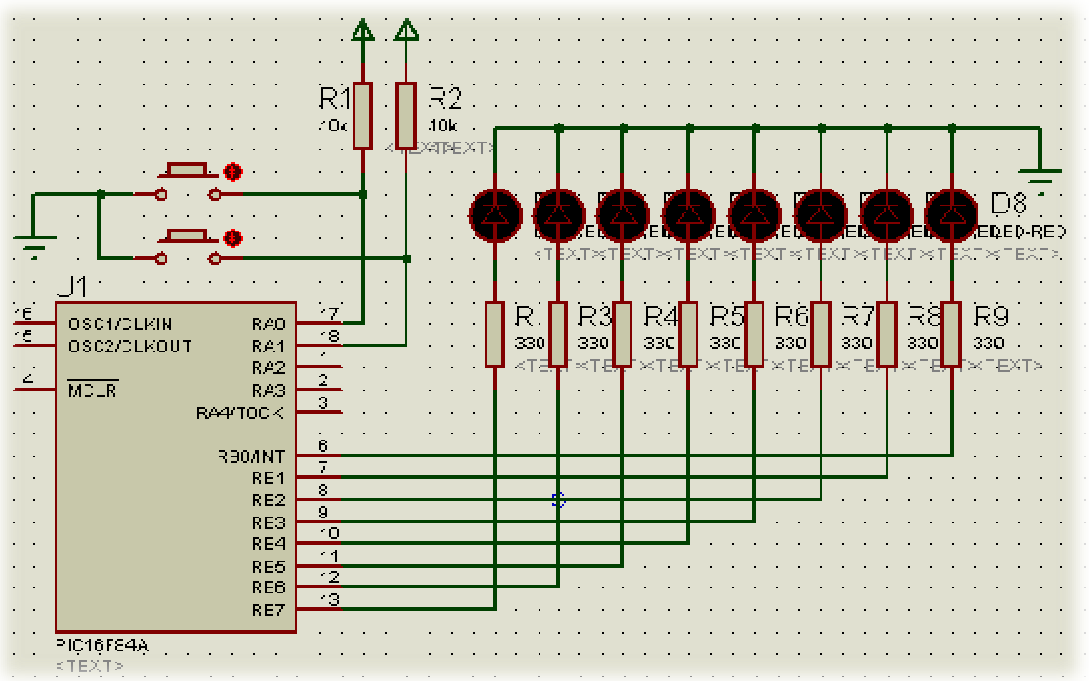
فهذا يعني أن المتغير yy فقط هو الذي سيقسم على ١٠ . لذلك الأقواس مهمة جداً خاصة في العمليات التي بها بسط ومقام فيفضل أن نجعل البسط بين قوسين و نجعل المقام أيضا بين قوسين إذا كان المقام به عملية حسابية كالجمع والطرح .. ونحوه .

توجد عمليات حسابية أخرى سنتعرف عليها في تجارب أخرى بإذن الله . . طريقة هذا الكتاب خطوة خطوة نأخذ معلومة صغيرة تليها معلومة صغيرة وهكذا .. أرجو من الله أن تستفيد وتستمتع بهذه الطريقة وأن يكون التعلم من خلال هذا الكتاب سهل وميسر للجميع .

التجربة (١٥)

في هذه التجربة المطلوب عمل عداد ثنائي بالسويتشات حيث سيكون هناك سويتش يزود القيمة الحالية. وسويتش آخر ينقص القيمة الحالية .

نرسم الدائرة بهذا الشكل



هيا بنا نكتب الكود !!

لنفكر سويا كيف سنكتب هذا الكود . . .؟!!

الآن في مشروعنا هذا لدينا سويتشين (مفتاحين) أريد أن آأمر البك وأقول له لو تم الضغط على السويتش الأول افعل كذا .. ولو تم الضغط على السويتش الثاني إفعل كذا . . إذن سنستخدم جملتين if الجملة الأولى خاصة بالمفتاح الأول والجملة الثانية خاصة بالمفتاح الثاني كما يلي :

if(porta.f0==0) { نفذ هذه الأوامر إذا تم الضغط على السويتش الأول }

if(porta.f1==0) { نفذ هذه الأوامر إذا تم الضغط على السويتش الثاني }

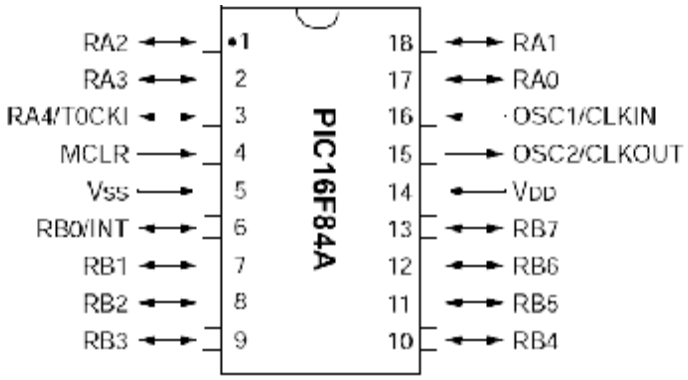
ولكن ما هي الأوامر التي نريد تنفيذها إذا تم الضغط على السويتش الأول ؟؟ سنكتب الأمر الذي يجعل قيمة portb تزداد بمقدار واحد فمثلا يمكننا أن نكتب portb++ وهذا الأمر كما ذكرنا سابقا سيجعل قيمة portb تزداد بمقدار واحد عن القيمة السابقة .

وماهو الأمر الذي نريد تنفيذه إذا تم الضغط على السويتش الثاني ؟؟ سنكتب الأمر الذي يجعل قيمة portb تنقص بمقدار واحد فمثلا يمكننا أن نكتب portb-- وهذا الأمر سيجعل قيمة portb تنقص بمقدار واحد عن القيمة السابقة التي كان عليها قبل الضغط على السويتش .

تحذير ..!! يجب عليك عند كتابة الأمر portb++ أو portb-- أن تكتب بعدهم أمر يجعل البك ينتظر لمدة معينة لأنه في حالة عدم فعل ذلك .. وعندما تضغط على السويتش الذي يزيد القيمة بمقدار واحد لن يزيدها بمقدار واحد بل ربما يزيدها بأكثر مع أنك ضغطت ضغطة واحدة على المفتاح والسبب العلمي الذي يفسر ذلك هو أن البك ينفذ جميع الأوامر بسرعة رهيبة جدا فأنت عندما تضغط على المفتاح وترفع يدك بسرعة قد تستغرق في هذا الأمر ربع ثانية أو نصف ثانية وربما أكثر أثناء هذه المدة يكون البك قد اختبر السويتش الذي أنت ضاغط عليه فيزيد من قيمة العدد ثم يختبر السويتش الذي لست ضاغطاً عليه فيجد أنه محرر فيرجع مرة أخرى ويختبر السويتش الذي أنت ضاغط عليه فيزيد من قيمة العداد مرة أخرى ثم يختبر السويتش الآخر ... وهكذا فربما في ربع ثانية (زمن ضغطك على المفتاح) يكون البك قد اختبر فيها هذا المفتاح أو السويتش الكثير من المرات وبالتالي سيزيد من القيمة التي عليها العداد العديد من المرات أيضا وهذا طبعا شيء غير مرغوب ... لذلك يجب عليك أن تقلل سرعته بأن تجعله ينتظر لمدة ثانية أو نصف ثانية مثلاً كما في الكود التالي :

```
void main()
{
 TRISB=0; TRISA=0XFF;
 PORTB=0;
while(1){
            if(porta.f0==0) { portb++; delay_ms(500);}
            if(porta.f1==0){portb--; delay_ms(500);}
          }
}
```

أعتقد أن الكود بسيط جداً ولا يحتاج لشرح أكثر .. والآن انظر للرسم ولاحظ MCLR



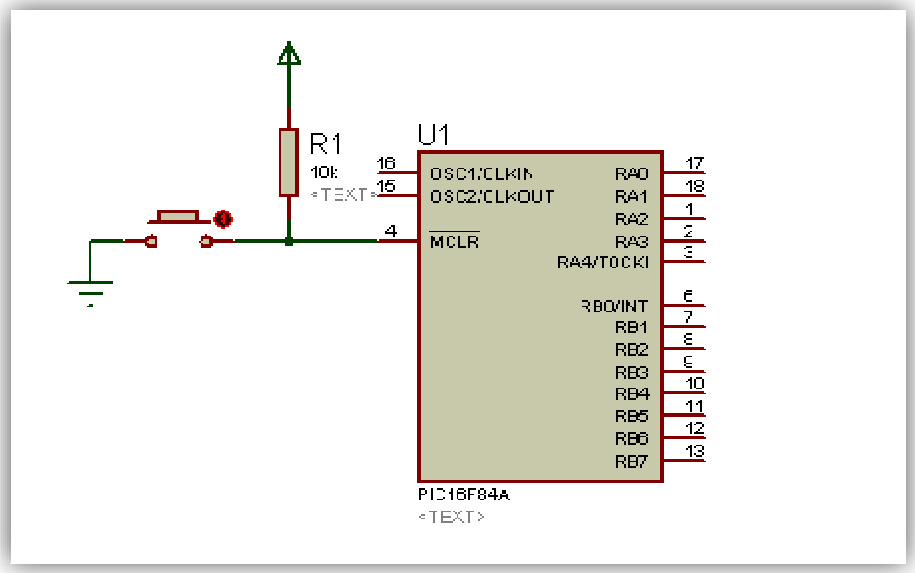
ستلاحظ وجود خط فوق كلمة MCLR وهذا يدل على أنه يعمل في حالة الجهد المنخفض active low .

أو الصفر فولت (أو بمعنى آخر سالب البطارية) . ولكن ماذا نستفيد من MCLR ؟؟

لابد أنك تلاحظ في أجهزة الحاسب الآلي الشخصية وجود زر أو سويتش عند الضغط عليه يقوم الحاسب الآلي بإعادة التشغيل وهذه العملية عادة ما تسمى بـ restart . إن MCLR له نفس الوظيفة فلو وصلنا به سالب البطارية فإنه سيقوم بإعادة تشغيل البك . بمعنى أنه سيبدأ تنفيذ أول سطر من الكود مرة أخرى .

فمثلا لو وصل البك إلى السطر العاشر من الكود وتم توصيل سالب البطارية إلى MCLR فإن البك سيعيد

التشغيل ويبدأ تنفيذ الأوامر من جديد من بداية السطر الأول ، ويمكننا أن نستفيد من هذه الخاصية في عمل سويتش يستطيع المستخدم فيه إعادة التشغيل لذلك نقوم بعمل التوصيلة التالية :



بهذا يكون MCLR واصل إليه دائما جهد موجب وعند الضغط على المفتاح الضاغط (push button) يتم إعادة تشغيل البك . طبعا هنا استخدمنا مفتاح ضاغط لأن البك يحتاج إلى جهد سالب لإعادة التشغيل ولكنه لا يبدأ بالعمل من جديد إلا بعد أن يصل إليه جهد موجب .

مشكلة مشهورة : عند تطبيق بعض دوائر البك عملياً وفي الواقع ، قد لا يعمل البك بشكل سليم أو ربما لا يعمل مطلقاً وقد يكون السبب في ذلك هو عدم توصيل MCLR بجهد موجب . إذن في دوائرك التي تستخدم فيها البك يفضل أن توصل الطرف MCLR بمقاومة متصلة بجهد ٥ فولت مثلا حتى تعمل دائرتك بشكل سليم كما يمكنك أن تضيف مفتاح ضاغط ليتمكن المستخدم من إعادة التشغيل .

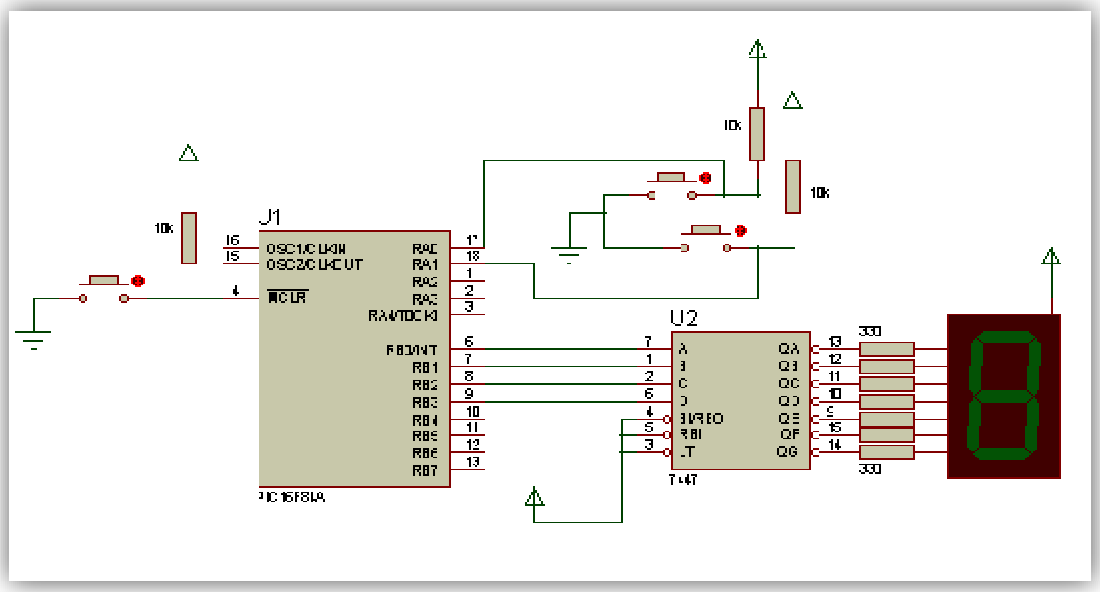
في مشروعنا هذا استخدمنا مفتاحين أحدهما يزيد من قيمة portb والآخر ينقص هذه القيمة يمكننا أيضاً إضافة مفتاح آخر نوصله بـ MCLR فعند الضغط عليه سيجعل قيمة portb بصفر لأنه بدأ تنفيذ البرنامج من جديد وهنا في بداية البرنامج يوجد الأمر portb=0; .

كما يمكننا إضافة سويتش آخر بدلا من MCLR عند الضغط عليه يقوم البك بإطفاء جميع الليدات أي يجعل الرقم صفر. فلو وصلناه بـ A2 يمكننا كتابة الأمر التالي :

```
if(porta.f2==0){portb=0; delay_ms(500);}
```

التجربة (١٦)

في هذه التجربة المطلوب عمل تعديل بسيط على الدائرة السابقة لنستخدم السفن سيجمنت بدلا من الليدات (عداد لسفن سجمنت واحدة بالسويتشات) وهنا طبعا ستكون حدود العد من ٠ إلى ٩ فقط .



ستلاحظ إضافة مقاومات قبل السفن سيجمنت . فمتى نضيف هذه المقاومات ومتى لا نضيفها ؟؟ عند تجربة الدائرة في برنامج المحاكاه بروتس فإنه لن يفرق ذلك سواءاً وضعت المقاومات أم لا لأن العناصر في هذا البرنامج غير قابلة للتلف. ولكن عند تطبيق البرنامج عمليا وفي الواقع فيجب أن تعلم نوع السفن سيجمنت المستخدمة فهناك أنواع تعمل بـ خمسة فولت (والبعض يخرج ٥ فولت) لذلك يمكن توصيلها مباشرة دون مقاومات . وهناك أنواع أخرى من السفن سيجمنت تعمل بجهد أقل من ٥ فولت لذلك يجب أن نضع قبلها مقاومات لتقليل الجهد ولكي لا تتلف السفن سيجمنت .

لذلك قبل الشروع في تنفيذ أي دائرة عملياً يجب عليك أن تعلم الجهد الذي يحتاجه كل عنصر لكي يعمل وكذلك التيار فربما تحتاج لعناصر إضافية في دائرتك مثل المقاومات أو الترانزستورات أو الريلاي أو .... على حسب الدائرة وسنذكر إن شاء الله ذلك في التجارب القادمة على حسب حاجتنا في التجربة .

والآن لنقوم بتعديل الكود ليصبح بالشكل التالي :

```
void main()
{
 TRISB=0; TRISA=0XFF;
 PORTB=0;
while(1){
            if(porta.f0==0) {
                              portb++;
                              if(portb==10) portb=0;
                              delay_ms(500);
                              }
            if(porta.f1==0)  {
                               portb--;
                              if(portb==255) portb=9;
                               delay_ms(500);
                              }
          }
}
```

من المفترض أنه عند الضغط على السويتش الأول الذي يزيد القيمة أن تزيد القيمة على السفن سيجمنت وهذا ما سيحدث فعلاً ، ولكن ماذا سيحدث إذا كان الرقم الذي يظهر على السفن سيجمنت يساوي تسعة وتم الضغط على السويتش مرة أخرى بالتأكيد سيظهر أشياء أخرى غير الأرقام على السفن سيجمنت

وللحماية من هذا الأمر تم وضع جملة if إضافية تختبر قيمة portb مباشرة بعد عملية الجمع portb++ جملة if هذه تختبر هل قيمة portb أصبحت بعشرة إذا كان كذلك لنبدأ العد من جديد أي من صفر .

```
if(portb==10) portb=0;
```

وجملة if هذه لابد أن تكون قبل أمر الإنتظار لأنه لو وضعناها بعد الأمر ;(delay_ms(500 فعندما تكون قيمة portb تساوي عشرة من خلال الأمر portb++ فلن يظهر رقم على السفن سيجمنت وسنرى ذلك لأنه يوجد الأمر ;(delay_ms(500 بعد الأمر portb++ ثم بعد ذلك يتم تنفيذ جملة if التي ستجعل السفن سيجمنت تظهر الرقم صفر ، وهذا طبعاً شيء غير مرغوب(أن لايظهر صفر مباشرةً) .

أيضاً ماذا سيحدث إذا كانت القيمة الظاهرة على السفن سيجمنت تساوي صفر وتم الضغط على السويتش A1 الذي ينقص القيمة الحالية . بالتأكيد وكما ذكرنا سابقاً لن تكون قيمة portb تساوي سالب واحد ولكن القيمة ستصبح ٢٥٥ لذلك وبما أن القيمة ٢٥٥ ليست ضمن القيم التي يمكنها الظهور على السفن سيجمنت (ليست ضمن الأرقام من صفر إلى تسعة) لذلك سيظهر شيء آخر على السفن سيجمنت غير الأرقام (وربما لا يظهر شيء مطلقا أي تنطفئ السفن سيجمنت كلها) لذلك يجب أن نضع هذا الأمر في الحسبان ونخبر البك بقولنا له إذا تم الضغط على السويتش الذي ينقص القيمة وكانت قيمة Portb تساوي صفر بالتأكيد بعد تنفيذ الأمر portb-- ستصبح القيمة هي ٢٥٥ اجعل القيمة تساوي تسعة بدلا من ذلك . أي أننا نخبره بقولنا لو كانت قيمة Portb تساوي ٢٥٥ اجعلها تساوي تسعة وذلك من خلال الأمر التالي :

```
if(portb==255) portb=9;
```

ولنفس السبب وضعنا الأمر السابق قبل أمر الإنتظار .

ولكن قد يخطر ببالك سؤال وهو لماذا لا نضع جملة if السابقة قبل الأمر portb-- . هذا طبعا لا يصح لأن قيمة portb لن تصبح بـ 255 إلا إذا تم تنفيذ الأمر portb-- أولاً وكانت portb تساوي صفر . ولكن هل يمكننا تعديل هذا الأمر ليصبح كما يلي :

```
if(portb==0) portb=9;
portb--;
```

تخيل ما سيحدث في هذه الحالة : الآن لو أن القيمة الظاهرة على السفن سيجمنت هي صفر وتم الضغط على السويتش A1 فإن البك سينفذ ما بداخل جملة if(porta.f1==0) وأول ما سينفذه هو عملية الإختبار هل قيمة portb تساوي صفر وسيتحقق الشرط وبالتالي سيجعل البك قيمة portb تساوي تسعة .. شيء جميل .. ولكن ما هذا إن البك سينتقل للأمر التالي وهو portb-- والذي سينقص القيمة ويجعلها تساوي ٨ هذا كله في منتهى السرعة لأنه لا يوجد أمر إنتظار بينهما . أي أننا لن نلاحظ ظهور الرقم تسعة على الإطلاق وسيظهر الرقم ٨ بعد الرقم صفر . لذلك يجب أن نقوم بعملية تعديل .. فكر في هذا الأمر وإذا أردت أن تعرف الإجابة التي أقترحها انظر للصفحة التالية :

```
if(porta.f1==0)  {
                    if(portb==0) portb=10;
                     portb--;
                     delay_ms(500);
                    }
```

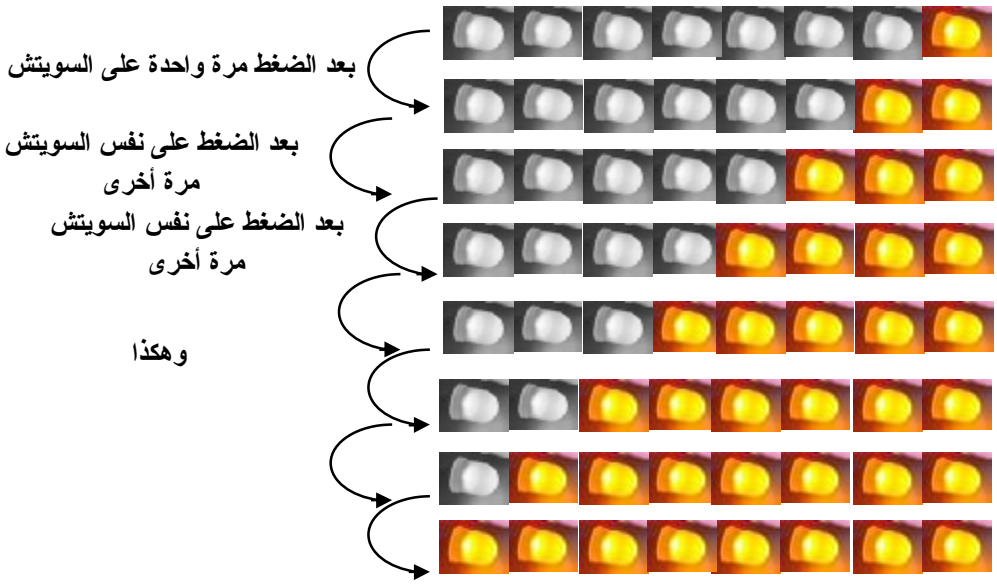
بهذه الطريقة إذا كانت قيمة portb تساوي صفر وتم الضغط على السويتش A1 فسيقوم البك بجعل قيمة portb تساوي عشرة وفي منتهى السرعة سينفذ الأمر الذي يليه وهو portb-- والذي بدوره سيجعل القيمة تساوي تسعة بعد ذلك سينتظر البك . لهذا سنلاحظ عند الضغط على السويتش ظهور الرقم تسعة مباشرة ولن نلاحظ ظهور أي تخاريف أو أرقام غريبة على السفن سيجمنت . ويمكن أيضاً تعديل ترتيب الأوامر في جملة if الخاصة بالسويتش A0 بنفس الطريقة فنجعل قيمة portb تساوي ٢٥٥ وبالتالي الأمر portb++ الذي يليها سيجعل القيمة تساوي صفر. و يمكن أن يكون الشكل النهائي لجملتي If كما يلي :

```
if(porta.f0==0) {
                    if(portb==9) portb=255;
                    portb++;
                    delay_ms(500);
                    }
if(porta.f1==0)  {
                     if(portb==0) portb=10;
                     portb--;
                     delay_ms(500);
                    }
```

عزيزي القارئ : وضحت لك ذلك من أجل التمرين فقط فأول كود كتبناه ممتاز وليس به أي مشاكل فما قمنا به فقط ترتيب الأوامر ، ولكني أعتقد أنه لابد أن تكتسب بعض المهارات البر مجية لتكمل مسيرتك في طريق الإحتراف بإذن الله . وأعتقد أنها فكرة جيدة لتنشيط العقل وتوسيع الآفاق .

التجربة (١٧)

في هذه التجربة المطلوب عند الضغط على سويتش موصل بـ A0 يضيء ليد معين (B0)ثم إذا ضغطنا مرة أخرى على نفس السويتش يضيء نفس الليد هو والذي يليه (B1) وهكذا إلى أن تضيء الثمان ليدات فإذا ضغطنا بعد ذلك على السويتش تنطفئ جميعا .



لنحاول التفكير قليلا في كيفية كتابة الكود الذي سيؤدي تلك الوظيفة ؟

مارأيك في أن نستخدم جملة شرطية (جملة if مثلا) تختبر هل تم الضغط على السويتش أم لا ؟ فإذا تم الضغط يضيء الليد B0 ولكن من المفترض أنه إذا ضغطت مرة أخرى على السويتش يظل الليد B0 مضيئا ويضيء الليد الذي بجواره B1 كيف سنفعل ذلك ؟! (فكر قليلا ..)

من المعلوم أنه إذا تحقق الشرط الخاص بجملة if (أي إذا تم الضغط على السويتش) فإن البك سوف ينتقل للأوامر التي بين القوسين الخاصين بجملة if وينفذ ما فيها وبالطبع يجب علينا أن نضع داخل هذين القوسين الأمر الذي سيضيء الليد إذن سنكتب الكود التالي :-

```
If(porta.f0==0) { portb.f0=1;
                  ..................
                  ..................
                  }
```

جميل ،، ولكن كيف سيعرف البك أنه تم الضغط على السويتش مرة أخرى ليضيء الليد b1 هل سنكتب جملة if أخرى غير الأولى ؟؟ أم ماذا ؟ .

هناك فكرة بأن نضع جملة if أخرى داخل جملة if الأولى وبهذا يختبر البك هل تم الضغط على السويتش مرة أخرى أم لا . فجملة if الأولى تختبر الضغطة الأولى وتضيء الليد b0 وجملة if الثانية تختبر الضغطة الثانية وتضيء الليد b1 وبالتالي يمكن كتابة الكود بالشكل التالي :

```
loop:
If(porta.f0==0){ portb.f0=1; delay_ms(500);
                  If(porta.f0==0) {portb.f1=1; }
                  ..........
                  }
goto loop;
```

فكرة جيدة .. أليس كذلك ؟ سأوضحها لك أكثر : إن البك سيختبر هل تم الضغط على السويتش فإذا حدث ذلك سيضيء الليد الموصل بـ b0 ثم ينتقل للأمر التالي وهو الذي يختبر هل تم الضغط على السويتش مرة أخرى فإذا تم الضغط فِعلاً سيضيء الليد الموصل بـ b1 . ولكن ماهذا ؟؟ هناك مشكلة كبيرة جداً إن البك ينفذ الأوامر بسرعة رهيبة جداً وهذا يعني أنني لو ضغت أول مرة على السويتش سيضيء الليد b0 ثم ينتقل البك وبمنتهى السرعة إلى الأمر التالي وهو الذي يجعله ينتظر نصف ثانية وبمنتهى السرعة أيضا ينتقل للأمر التالي وهو جملة if فيجد أن الشرط لم يتحقق لأنني لم أضغط بعد على السويتش مرة أخرى في هذه المدة البسيطة التي يستغرقها هو في تنفيذ الأوامر وبالتالي لن ينفذ الأوامر

التي داخل جملة if الثانية لعدم تحقق الشرط وينتقل للأمر الذي يليها ويخرج من جملة if وهذا يعني أن البك لو بدأ تنفيذ الأوامر من جديد من خلال goto loop; مثلاً فإنه عند الضغطة الثانية للسويتش سينفذ البك عندها جملة if الأولى وليس الثانية وبالتالي سيضيء الليد b0 (هو مضيء أصلا) وليس b1 . إذن الحل هو أنه إذا تم الضغط على السويتش لأول مرة أجعل البك لا يخرج من جملة if الأولى كما يلي :

```
loop:
If(porta.f0==0){ portb.f0=1;
                a1:
                If(porta.f0==0) {portb.f1=1;}
                goto a1;
                }
goto loop;
```

تخيل الآن ماذا سيحدث .!! وبهدوء تخيل معي ما يلي : عند تشغيل الدائرة لأول مرة فإن البك ينتظر إلى أن يتم الضغط على السويتش فإذا لم يحدث ذلك أي لم يتحقق الشرط يختبر الشرط مرة أخرى عن طريق goto loop; التي تجعله يعيد تنفيذ الأوامر مرة أخرى عن طريق ذهابه للعنوان loop: وبالتالي هذا ضمان لنا أنه في أي وقت نضغط فيه على السويتش سيبدأ البك بتنفيذ الأوامر التي بداخل جملة if الأولى . جميل ،، والآن إذا تحقق الشرط أي أننا ضغطنا على السويتش لأول مرة سينفذ البك ما بداخل القوسين وأول أمر ينفذه هو الذي سيضيء الليد b0 ثم يختبر البك هل تم الضغط على السويتش مرة أخرى فإذا لم يتم الضغط في تلك اللحظة سينتقل للأمر التالي وهو goto a1; والذي يجعله يكرر عملية الإختبار مرة أخرى وهكذا باستمرار يظل البك يختبر الشرط إلى أن نضغط مرة أخرى على السويتش فيضيء الليد الثاني الموصل بـ b1 . فكرة جيدة أليس كذلك !!.

والآن لنطبق نفس الفكرة على أربعة ليدات b0 , b1 , b2 , b3 ليكون الكود كما يلي :

```
loop: delay_ms(500); portb=0;
If(porta.f0==0)
        { portb.f0=1;
                a1:  If(porta.f0==0)  {portb.f1=1;
                                        a2:   If(porta.f0==0)  {portb.f2=1;
                                                                a3:     If(porta.f0==0) {portb.f3=1; goto loop;}
                                                                goto a3; }
                                        goto a2;}
                goto a1;
        }
goto loop;
```

في الكود السابق سيبدأ البك بالإنتظار لمدة نصف ثانية ثم تنطفيء جميع الليدات (ستعرف فائدة أمر الانتظار وأمر إطفاء الليدات في النهاية) ثم يختبر البك هل تم الضغط على السويتش في هذه اللحظة فإذا لم يحدث ذلك يترك البك جملة if ولا ينفذ ما بداخلها وينتقل إلى الأمر التالي وهو goto loop; والذي سيجعل البك يختبر مرة أخرى وهكذا باستمرار يظل البك يختبر إلى أن تتم الضغطة الأولى على السويتش حينها يضيء الليد b0 ثم يختبر هل تمت الضغطة الثانية على السويتش أم لا ؟ فإذا لم تتم ينتقل للأمر التالي وهو goto a1; والذي يجعله يكرر عملية الاختبار باستمرار إلى أن تتم الضغطة الثانية فيضيء الليد b1 ثم يختبر مرة أخرى وبنفس الطريقة يظل يختبر بسبب وجود الأمر goto a2; وحين يضغط المستخدم الضغطة الثالثة يضيء الليد b2 ويختبر البك مرة أخرى ويكرر عملية الإختبار بسبب وجود الأمر goto a3; وعند الضغط على السويتش مرة أخرى يضيء الليد b3 ثم ينتقل البك لبداية البرنامج من خلال الأمر goto loop; ولهذا ستجد أنني وضعت الأمر portb=0; بعد loop: حتى يبدأ البرنامج من جديد فتكون جميع الليدات مطفئة وبالتالي إذا ضغط المستخدم مرة أخرى على السويتش يضيء أول ليد ثم إذا ضغط مرة أخرى يضيء نفس الليد والليد الذي يليه وهكذا باستمرار . أما الأمر

delay_ms(500); فهو ضروري جداً فبدون وجوده لن نلاحظ أن الليد b3 أضاء لأنه فور إضاءته سينتقل البك لبداية البرنامج وينفذ الأمر portb=0; الذي يطفئ جميع الليدات لذلك لا بد من وضع وقت لكي نلاحظ فيه عملية الإضاءة .

يمكنك بنفس الطريقة كتابة كود تطبقه على جميع الليدات من b0 إلى b7 .

والآن يوجد خطأ في الكود السابق وهو أننا يجب أن نضع أمر إنتظار بعد إضاءة أي ليد لأنه كما تعلمنا جيدًا أن البك ينفذ الأمر الواحد بسرعة كبيرة وينتقل للأمر الذي يليه فينفذه أيضاً بسرعة كبيرة . والآن تخيل معي ماذا سيفعل البك عندما يتم الضغط على السويتش .!!

بالطبع سينتقل للأمر الذي يضيء الليد وبمنتهى السرعة وقبل أن يرفع المستخدم يده من على السويتش سينتقل البك للأمر التالي وهو الأمر الذي يختبر السويتش فيجد أن السويتش مضغوط عليه فيقوم بتنفيذ الأمر الذي يليه وهو إضاءة الليد الذي بجواره ، وهكذا . لذلك سنضع أمر إنتظار بعد إضاءة أي ليد ولتكن هذه المدة ١٠٠ ملي ثانية مثلا . وهناك حل آخر وهو استخدام جملة while .

كيف تعمل جملة while ؟

إن جملة while تشبه كثيراً جملة if فهي تختبر شرط معين إذا تم تحققه تنفذ الأوامر التي بين القوسين ولكن الفرق هنا أنه بعد تنفيذ الأوامر التي بين القوسين يختبر البك الشرط مرة أخرى فإذا تم تحققه ينفذ الأوامر مرة أخرى وبعد تنفيذها يختبر الشرط مرة أخرى فإن كان متحققاً يكرر تنفيذ الأوامر التي بين القوسين وهكذا إلى أن أن يأتي الوقت الذي لا يتحقق فيه الشرط حينها يكون البك قد انتهى من تنفيذ جملة while وينتقل إلى الأمر التالي .

```
While ( الشرط )
{
هنا نكتب الأوامر التي ستنفذ في حالة تحقق الشرط
وسيتكرر تنفيذها طالما ظل الشرط متحققا
}
```

إن جملة while من الأوامر البرمجية الهامة فإنها تفيد كثيراً في تبسيط الكود وحل الكثير من المشاكل ويجب علينا فهمها جيداً لذا سنضرب عليها بعض الأمثلة لنفهمها جيداً إن شاء الله :-

---

مثال : نريد ان نكتب كود وظيفته أن يختبر السويتش الموصل بـ A0 بحيث أن الليد الموصل بـ B0 يضيء ثم ينطفئ باستمرار طالما كنا ضاغطين على السويتش وفي حالة إذا رفعنا يدنا من على السويتش فإن الليد لا يضيء .

ببساطة سنكتب الكود الجميل التالي الذي سيكون في غاية البساطة بسبب استخدامنا لجملة while

```
while(porta.f0==0) {
                    portb.f0=1;
                    delay_ms(150);
                    portb.f0=0;
                    delay_ms(150);
                   }
```

طبعا لا تنسى كتابة void main() وكذلك بقية الأوامر اللازمة مثل trisa و trisb .

مثال ٢ : نريد أن نكتب كود وظيفته إضاءة الليد B0 ثم إطفاءه بعد نصف ثانية بحيث تحدث عملية الإضاءة والإطفاء تلك إذا تم الضغط على السويتش الموصل بـ A0 مع جعل إمكانية حدوث ذلك خمس مرات فقط ، بمعنى آخر : إذا تم الضغط على السويتش فإن الليد سيضيء ثم ينطفئ وإذا تم الضغط على السويتش مرة أخرى سيضيء الليد ثم ينطفئ وهكذا إلى أن تتم الضغطة الخامسة على السويتش فيضيء الليد ثم ينطفئ بعدها إذا تم الضغط على السويتش مرة أخرى فلن تحدث عملية الإضاءة والإطفاء أبداً .

(فكر قليلاً .... ثم فكر مرة أخرى لتزيد من قدراتك ومهاراتك البرمجية .... بعدها يمكنك النظر للحل المقترح في الصفحة التالية الذي قد لا يكون هو الحل الوحيد وقد يكون أيضًا ليس الحل الأمثل .. !!)

سنقوم باستخدام متغير إضافي يساعدنا على الحل ، هذا المتغير يمثل عدد مرات التكرار (مرات السماح بتنفيذ أوامر معينة)

```
mm=1;
while(mm<6) {
        if(porta.f0==0) { portb.f0=1; delay_ms(500);
                portb.f0=0; delay_ms(500);
                mm++;
               }
        }
```

اترك لنفسك بعض الوقت لفهم الكود السابق ولتخيل ما سينفذه البك .

شرح الكود السابق : إن البك سيقوم بجعل قيمة المتغير mm تساوي واحد من خلال الأمر mm=1; ثم بعد ذلك ينفذ الأمر التالي وهو جملة while فيختبر الشرط فيجد أنه متحققاً لأن قيمة mm أقل من ستة فينفذ ما بداخل جملة while أي سينفذ جملة if .

وجملة if لن تتحقق إلا إذا ضغط المستخدم على السويتش وفي حالة عدم تحقق الشرط (أي لن يتم الضغط بعد على السويتش) يكون البك قد انتهى من تنفيذ الأوامر التي بين القوسين الخاصين بجملة while فيختبر الشرط الخاص بجملة while مرة أخرى فيجده متحققاً لأن المتغير mm يساوي واحد أي أنه أقل من ستة فيكرر تنفيذ الأوامر التي بين القوسين ومعنى هذا أن البك سيكرر تنفيذ جملة while دائما وباستمرار طالما ظلت قيمة المتغير mm أقل من ستة وبالتالي ستتكرر عملية اختبار السويتش A0 ولكن ماذا سيحدث إذا تم الضغط على السيوتش A0 ؟؟ بالطبع سيضيء الليد B0 ثم بعد نصف ثانية ينطفئ الليد ثم ينتظر نصف ثانية ثم تزداد قيمة المتغير mm وبالتالي ستصبح قيمة mm تساوي واحد بعد ذلك يختبر البك الشرط الخاص بجملة while مرة أخرى فيجد أنه مازال متحققاً وبالتالي سينفذ

ما بداخل جملة while مرة أخرى وفي حالة تم الضغط على السويتش مرة أخرى ستزداد قيمة المتغير mm وتصبح بثلاثة وهكذا في كل مرة يتم الضغط على السويتش يضيء الليد وينطفئ وتزداد قيمة المتغير mm . إلى أن تصبح قيمته بخمسة فإذا تم الضغط على السويتش مرة أخرى يضيء الليد وينطفئ وتصبح قيمة المتغير mm تساوي ستة حينها عند اختبار البك للشرط الخاص بجملة while سيجد أن الشرط غير متحقق لأن المتغير mm ليس أقل من ستة وبالتالي سينتقل البك للأمر الذي بعد جملة while .

أليس الأمر رائعاً . . !! بكل سهولة ويسر وباستخدام جملة while حصلنا على قوة التحكم بعدد قليل من أسطر الكود . ألم أخبرك أن جملة while مهمة جداً ولنتعرف أكثر على مدى أهميتها وإمكانياتها الرائعة إليك هذا المثال :

مثال : نريد أن نكتب كود وظيفته جعل البك ينتظر ولا يفعل أي شيء طالما أن السويتش لم يتم الضغط عليه بعد وفي حالة تم الضغط على السويتش ينتقل للأوامر التالية .

```
While(porta.f0==1){ }

هنا نكتب الأوامر التي ستنفذ في حالة عدم تحقق الشرط الخاص

بجملة while أي في حالة تم الضغط على السويتش .
```

ما هذا .. ؟؟ إن القوسين الخاصين بجملة while ليس بينهم أوامر ، نعم صحيح لا تقلق الأمر بسيط وسنحاول تحليله جيداً . إن البك سيختبر الشرط الخاص بجملة while فيجده متحققا لأنه لم يتم الضغط بعد السويتش (طبعا A0 تصبح قيمتها بواحد في حالة عدم الضغط على المفتاح وتصبح بصفر في حالة الضغط عليه – هذا طبقا للتوصيلة الخاصة بالسويتش التي اعتدنا عليها-) . وبما أن الشرط متحققا فيسنفذ البك ما بداخل القوسين . وهنا لن يجد البك شيئا ينفذه وبالتالي يكون قد انتهى من تنفيذ ما بداخل القوسين فيقوم باختبار الشرط مرة أخرى فيجد أنه متحققاً فينفذ ما بين القوسين (أي أنه لن يفعل شيء) ويختبر الشرط مرة أخرى وهكذا باستمرار إلى أن يتم الضغط على السويتش حينها لن يتحقق الشرط لأن قيمة A0 ستصبح تساوي صفر وهنا يكون البك قد انتهى من تنفيذ جملة while وينتقل للأوامر التالية .

تكافئ

```
loop:

if(porta.f0==1) goto loop;
```

While(porta.f0==1){ }

بعد جولتنا السابقة مع جملة while لنرجع لموضوعنا الأساسي ولنحاول استخدامها في هذه التجربة .

للتذكير : في هذه التجربة المطلوب عند الضغط على سويتش موصل بـ A0 يضيء ليد معين (B0)ثم إذا ضغطنا مرة أخرى على نفس السويتش يضيء نفس الليد هو والذي يليه (B1) وهكذا إلى أن تضيء الثمان ليدات فإذا ضغطنا مرة أخرى تنطفئ جميعا .

فكر جيداً واترك لنفسك الوقت فكما أن التمارين الرياضية تزيد من قوة العضلات وتزيد من مرونة الجسم فكذلك التمارين البرمجية تزيد من قوة احترافك البرمجي وتزيد من سرعة حلك للمشكلات البرمجية.

```
void main()  {trisa.f0=1;   trisb=0; portb=0;
while(1) {  while(porta.f0==1) { }
            portb=0b00000001;   delay_ms(100);
            while(porta.f0==1) { }
            portb=0b00000011;   delay_ms(100);
            while(porta.f0==1) { }
            portb=0b00000111;   delay_ms(100);
            while(porta.f0==1) { }
            portb=0b00001111;   delay_ms(100);
            while(porta.f0==1) { }
            portb=0b00011111;   delay_ms(100);
            while(porta.f0==1) { }
            portb=0b00111111;   delay_ms(100);
           while(porta.f0==1) { }
            portb=0b01111111;   delay_ms(100);
           while(porta.f0==1) { }
            portb=0b11111111;   delay_ms(100);                  }    }
```

الآن الكود أصبح أجمل مع العلم أنه يمكن اختصاره أكثر من ذلك وهذا ما سنعرفه بعد قليل إن شاء الله .

هل هناك حلول أخرى ؟ بالتأكيد نعم ، فالأفكار البرمجية لا تتوقف وليست لها حدود وتأتي من الاجتهاد وكثرة الإطلاع والتفكير وقبل ذلك كله كرم من الله سبحانه وتعالى بأن يلهم الشخص الفكرة المناسبة ويوفقه لتنفيذها فالعلم كله بيد الله يرزقه من يشاء فاللهم ارزقنا وزدنا من كرمك وفضلك يا أكرم الأكرمين . نعود لموضوعنا وإليك الفكرة التالية وهي فكرة مهمة جداً ويمكنك استخدامها بكثرة في حل الكثير من المشاكل البرمجية وخصوصا إذا كان السويتش كل ضغطة عليه تجعل البك يقوم بوظيفة معينة ، و تعتمد هذه الفكرة على استخدام متغير إضافي . هذا المتغير تكون قيمته في البداية بصفر مثلاً فإذا تمت أول ضغطة على السويتش تصبح قيمته بواحد ويكون هناك أمر آخر يختبر قيمة المتغير فإذا كانت بواحد ينفذ أمر معين . وعند الضغط مرة أخرى على السويتش تزداد قيمة هذا المتغير وتصبح بإثنين وبالطبع سيكون هناك أمر يختبر قيمة هذا المتغير فلو كانت بإثنين سيقوم بتنفيذ أوامر معينة وإليك الحل التالي للتجربة الحالية :

```
void main()  {char x; trisa.f0=1;   trisb=0; portb=0;
while(1) { if(porta.f0==0) {x++ ; delay_ms(100);}
            if(x==0) portb=0;
           if(x==1) portb=0b00000001;
           if(x==2) portb=0b00000011;
           if(x==3) portb=0b00000111;
           if(x==4) portb=0b00001111;
           if(x==5) portb=0b00011111;
           if(x==6) portb=0b00111111;
           if(x==7) portb=0b01111111;
           if(x==8) portb=0b11111111;
           if(x==9) x=0;
            }  }
```

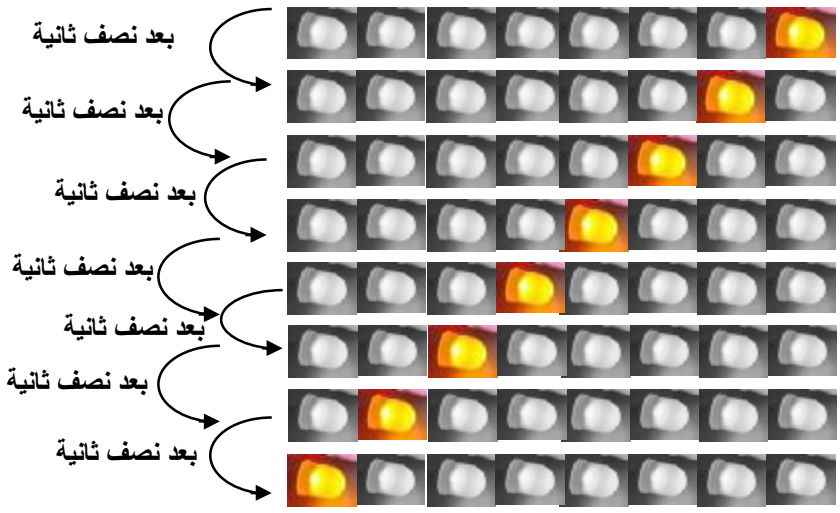
تخيل معي ما سيحدث .. عند تشغيل البرنامج قام البك بإنشاء متغير اسمه x هذا المتغير قيمته الإفتراضية ستكون بصفر وعندما ينتقل البك إلى ما بداخل جملة while(1) سيختبر هل تم الضغط على السويتش أم لا ولنفترض أنه لم يتم الضغط على السويتش بعد إذن سينتقل البك للأمر التالي ، ستلاحظ أن بقية الأوامر ما هي إلا اختبار لقيمة المتغير x إحدى هذه الجمل الاختبارية سيتحقق شرطها وفي حالتنا هذه ستتحقق الجملة الأولى لأن قيمة x تساوي صفر وبالتالي سيطفئ البك جميع الليدات من خلال الأمر portb=0; ويكرر البك عمليات الاختبار دائما لأنها ضمن جملة while(1) وفي لحظة معينة عندما يضغط المستخدم على السويتش ستزداد قيمة المتغير x من خلال الأمؤ x++; وستصبح بواحد وبالتالي ستتحقق جملة شرط معينة وهي if(x==1) وسيضيء الليد b0 ويظل البك مكرراً لعمليات الاختبار إلى أن يتم الضغط على السويتش مرة أخرى فتزيد القيمة وتصبح قيمة المتغير تساوي ٢ فيضيء الليد b0,b1 ... وهكذا . وهنا يجب الإنتباه إلى أنه إذا أصبحت قيمة المتغيرx تساوي تسعة فيجب أن نجعل قيمة المتغير تساوي صفر لنبدأ من جديد حيث أننا لدينا تسعة احتمالات كل احتمال سيؤدي إلى خرج معين على portb (هذه الاحتمالات هي x تساوي صفر أو واحد أو .... أو ثمانية) لذلك عندما يضغط المستخدم على السويتش بعد أن كانت x تساوي ثمانية ستصبح قيمتها تساوي تسعة وبالتالي لابد أن نحولها إلى صفر من خلال الأمر

```
if(x==9) x=0;
```

التجربة (١٨)

لعلنا أخذنا الكثير من التجارب التي تعتمتد على الليدات ، ولكن اعلم أخي الحبيب أن الليدات يمكن أن تستبدل ويوضع بدلا منها مواتير أو سماعات أو عناصر كهربية والكترونية أخرى أو أجهزة أخرى كل ذلك بنفس الفكرة البرمجية التي استخدمناها مع الليدات كل ما سيتغير هو ما سنوصله بالطرف الذي سنخرج منه جهد معين . كما أن الهدف من هذه التجارب هو اكتساب مهارات برمجية لابد منها وكذلك التعرف على الكثير من أوامر اللغة والتي ستشعر بأهميتها كثيراً عندما ننطلق إلى تطبيقات أكبر ومشاريع أكبر واعلم أن فهمك للأساسيات من خلال التجارب السابقة سيكون هو مصدر قوتك في برمجة المايكروكنترولر وسيكون هو السبب في جعل التجارب والمشاريع الكبيرة سهل جداً بالنسبة إليك . فلتصبر قليلا على هذه التجارب وستجد خلال التجارب القادمة إن شاء الله ما هو أكثر متعة بالنسبة لك .

في هذه التجربة الهدف هو التعرف على بعض الأوامر البرمجية الجديدة ، فالمطلوب هو إضاءة الليدات بالشكل التالي :-



بالطبع هذا البرنامج سهل ويمكن إنجازه بعدة طرق من أسهلها أن نجعل portb=0b00000001; ثم بعد ذلك نجعل البك ينتظر نصف ثانية من خلال الأمر delay_ms(500); ثم بعد ذلك نجعل portb=0b00000010; .... وهكذا . نحن الآن سنقوم بعمل هذه التجربة بطريقة أخرى وبأمر برمجي جديد فلعلك تلاحظ أن الليد في كل خطوة يتحرك مرة لليسار (تتم إزاحته لليسار) فهل هناك أمر برمجي يقوم بعمل إزاحة لليسار ؟؟ بالطبع نعم وهو أمر بسيط جداً . انظر للكود التالي فسيقوم بالوظيفة المطلوبة من خلال استخدام هذا الأمر .

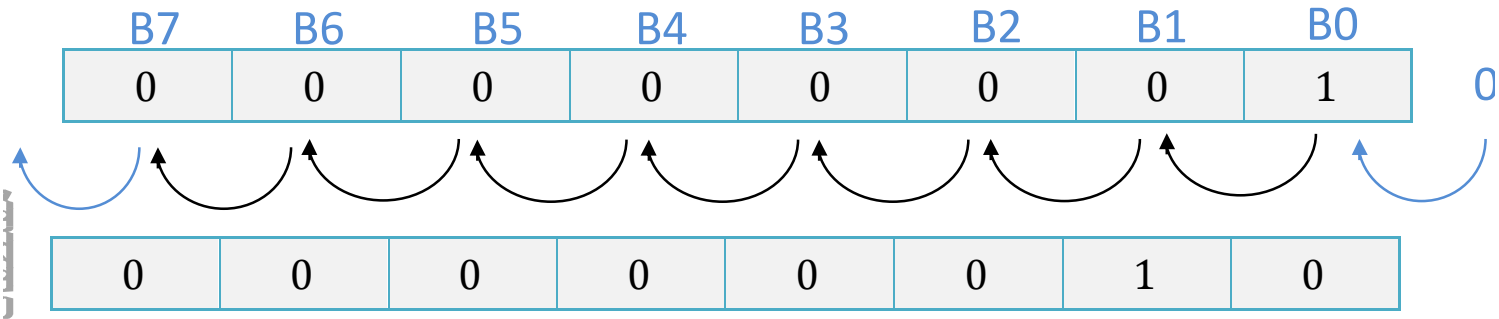
```
void main()
{ trisb=0;
ahmad:
        Portb=0b00000001;       delay_ms(500);
        for(x=0;x<8;x++){
                              portb=portb<<1;
                              delay_ms(500);
                            }
goto ahmad;
}
```

لنحلل الكود السابق : طبعا void main() تكتب في بداية أي برنامج والأمر trisb=0; سيجعل جميع الأطراف (أو الرجول) من B0 إلى B7 مهيئة على أنها خرج . وبالنسبة لكلمة ahmad: فهي مجرد عنوان والأمر الأخير goto ahmad; سيجعل البك يكرر تنفيذ ما بين هذين الأمرين باستمرار وذلك من خلال الذهاب إلى هذا العنوان وتنفيذ ما بعده ، وهذا طبعا له نفس وظيفة الأمر while(1) ووكذلك الأمر for(;;) كما ذكرنا ذلك سابقا.

بعد ذلك الأمر portb=0b00000001; سيجعل الليد الموصل بـ B0 مضيئاً أما بالنسبة لجملة for فالهدف منها تكرار مجموعة من الأوامر ثمان مرات وبدونها كنا سنضطر إلى كتابة الأمرين portb=portb<<1; delay_ms(500); ثمان مرات . إذن الفائدة من جملة for هنا تكرار تنفيذ مجموعة من الأوامر عدة مرات .

نأتي إلى النقطة الجديدة والمهمة في هذا الدرس وهو الأمر portb=portb<<1; فهذا الأمر يقوم بعمل إزاحة لمحتويات portb مرة لليسار . ولكن كيف تتم هذه الإزاحة ؟؟ ببساطة المتغير أو المسجل portb يتكون من ثمانية أجزاء (٨ بت) كل بت سينتقل مرة لليسار كما يوضح ذلك الشكل التالي :

شكل portb قبل عملية الإزاحة لليسار



شكل portb بعد عملية الإزاحة لليسار

من الرسم السابق يتضح لنا أن عملية الإزاحة (لليسار) تتم كما يلي :

البت رقم سبعة (B7) سوف يزاح لليسار خارج نطاق المتغير أو المسجل PORTB أي سيتم حذف ما به وستصبح قيمته تساوي قيمة البت رقم ستة B6 (أو بمعنى آخر B6 سوف ينتقل إلى B7) و كذلك B5 سوف ينتقل (أو يزاح) إلى B6 .. وهكذا إلى أن ينتقل B0 إلى B1 . أما B0 فستصبح قيمته بصفر وكأن الصفر يأتي له من الخارج. ومن الخطأ أن تظن أن B0 قيمته ستصبح هي قيمة B7 فليس هناك علاقة بينهما . فقط اعلم أنه إذا تمت عملية إزاحة لليسار مرة واحدة فإن B0 تصبح تساوي صفر وباقي البتات Bits سوف تزاح لليسار .

أما إذا كانت الإزاحة لليسار مرتين أي أن الأمر بالصورة التالية ; portb=portb<<2 فهذا يعني أننا قمنا بعملية الإزاحة مرة واحدة لليسار كما بالسابق ثم قمنا بعمل هذه الإزاحة مرة أخرى وبالتالي ستكون قيمة portb تساوي 00000100 .

ماذا يعني الأمر portb=portb<<1; ؟؟ يعني أن قيمة Portb الجديدة تساوي قيمة portb السابقة مع عمل إزاحة لليسار مرة واحدة لها . وهذا يشبه الأمر portb=portb+3; فهذا الأمر يعني أن قيمة portb الجديدة تساوي قيمة portb السابقة مضافاً إليها ثلاثة .

والعلامة >> تعني إزاحة لليسار ، وإذا أردنا أن نجعل الإزاحة لليمين فسنجعلها << وكأنها سهم يشير إلى إتجاه الإزاحة . أما الرقم واحد فهو يشير إلى عدد مرات الإزاحة ، ويمكن تغيير هذا الرقم بدلاً من تكرار كتابة نفس الأمر . فالأمر portb=portb<<2; يكافئ الأمر portb=portb<<1; ولكن مع تكرار كتابته مرتين .

وإليك الآن بعض الأمثلة على عمليات الإزاحة :-

مثال ١ :

إذا كان لدينا portb=0b00100111; ثم كتبنا الأمر التالي portb=portb<<1; فإن قيمة portb ستصبح بعد هذه العملية كالتالي 0b01001110; . ستلاحظ أن جميع البتات bits تم إزاحتها لليسار خطوة واحدة (خطوة واحدة بسبب الرقم واحد بعد علامة <<) .

مثال ٢ :

إذا كان لدينا متغير وليكن اسمه led وكان هذا المتغير يساوي ٦ أي أن led=0b00000110; ثم كتبنا الأمر led>>1; فإن قيمة المتغير led سيساوي 0b00000011 وحينها إذا كتبنا الأمر portb=led; فإن B0 و B1 ستصبح قيمتهما بواحد أما B2 و b3 و ... B7 سيصبحوا جميعا بصفر وذلك لأننا قمنا بمساواة portb بالمتغير led . أي كأننا كتبنا الأمر portb=0b00000011;

ملحوظة تساعدك على عدم اللخبطة والالتباس : لاحظ أن الإزاحة كانت لليمين وكأن العلامتين << هما سهم يشير لليمين . أما إذا كانت الإزاحة لليسار فإننا سنستخدم العلامتين >> وكأنهما سهم يشير لليسار .

تأمل معي الكود التالي :

```
char led;
led=0b00000110;
led=led>>1;
portb=led;
```

أول أمرين كان يمكن اختصارهما وجعلهما أمر واحد يقوم بتعريف المتغير وفي نفس الوقت يقوم بجعل قيمته تساوي ٦ (00000110 بالنظام الثنائي) وذلك بأن نكتب char led = ob00000110; وليس هناك فرق بين الطريقتين .

كذلك الأمرين led=led>>1; و portb=led; يمكن أن نكتبهم بطريقتين إما هذه الطريقة أو أن ندمجهم في أمر واحد وذلك بأن نكتب portb=led>>1; ولكن يا ترى ما الفرق بين الطريقتين ؟

بالنسبة للوظيفة فكليهما سيؤدي نفس الوظيفة في هذا البرنامج ولذلك ذكرت الطريقتين . ولكن هناك اختلاف بسيط قد يستخدم في بعض التطبيقات ، وببساطة يمكن القول أنه عندما نكتب الأمر led=led>>1; فهذا يعني أن قيمة المتغير led قد تغيرت الآن وتم عمل إزاحة لمحتويات هذا المتغيرلليمين . وهذا يعني أننا إذا قمنا بمساواة هذا المتغير بمتغير آخر وليكن اسمه GG فإن المتغير GG ستصبح قيمته تساوي 0B00000011 . ولأن PORTB يعتبر متغير لكن من نوع خاص فإنه عند مساواته بالمتغير led ستصبح قيمة portb تساوي 0b00000011 .

أما الطريقة الثانية : وهي كتابة أمر واحد فقط portb=led>>1; فهذا يعني أن قيمة portb ستساوي قيمة المتغير led مع عمل عملية إزاحة لليمين . لاحظ معي كما أخبرتك سابقا أنه في علاقات التساوي فإن الطرف الأيسر هو من سيتغير أما الطرف الأيمن سيبقى ثابتا . أي أن المتغير led في هذه الحالة سيصبح كما هو وستظل قيمته تساوي 0b00000110 ولن يتم عمل إزاحة لها ، كل ما حدث أن قيمة portb استفادت من قيمة المتغير led وأخذت هذه القيمة وعملت إزاحة لليمين لها دون أن يؤثر ذلك على قيمة المتغير led .

وبمعنى آخر فإنه إذا كان led=0b00000110; ثم كتبنا بعد ذلك الامر portb=led>>1; ثم كتبنا بعد ذلك الأمر GG=led; . فإن النتائج التي لدينا ستكون كالتالي :-

أولا : بما أن قيمة led لم تتغير (وإنما تم الإستفادة من قيمتها فقط) فإن قيمة المتغير GG ستصبح تساوي قيمة المتغير led الأصلية دون أي تغيير أي ستصبح تساوي 0b00000110 .

ثانيا: قيمة portb ستصبح تساوي 0b00000011 وذلك من خلال الامر portb=led>>1; .

ولأقرب هذا المعنى لديك أكثر وأكثر فتخيل معي ما يلي :

عندما أكتب الأمر portb=led>>1; فهذا الأمر هو نفسه لو وضعت قيمة المتغير led مكان اسمه في الأمر السابق أي كأنني كتبت الأمر ; portb=0b00000110>>1 فقط أخذنا القيمة ووضعناها هنا وتركنا المتغير led كما هو أي أنه بعد هذه العملية فإن ; led=0b00000110 . و portb=0b00000011; .

مثال ٣ :

إذا كان لدينا المتغير x=0b11000111; ثم قمنا بعمل إزاحة لليمين خطوتين من خلال الأمر x=x>>2; فإن قيمة المتغير x سوف تصبح 0b00110001 . لماذا ؟ لأننا قمنا بعمل إزاحة خطوتين أو مرتين المرة الأولى ستصبح قيمة x تساوي 0b01100011 وهذا الناتج سوف نقوم بعمل إزاحة له مرة أخرى لليمين فيصبح الناتج هو 0b00110001 .

مثال ٤ :

إذا كان لدينا المتغير ; bb=0b00000001 ثم قمنا بعمل إزاحة سبع خطوات لليسار فإن قيمة المتغير bb ستصبح تساوي 0b10000000 .

إن عمليات الإزاحة (لليمين أو لليسار) مهمة جداً في العديد من التطبيقات وليست فائدتها الأساسية هو استخدامها مع الليدات ولكن لها استخدامات أخرى أهم من ذلك بكثير لذلك كان يجب أن يتم شرحها ببعض التفصيل .

التجربة (١٩)

والآن لنقوم بعمل نفس التجربة السابقة حيث سنجعل البرنامج يؤدي نفس الوظيفة ولكن ليس بعد كل نصف ثانية بل بعد كل ضغطة على السويتش (باعتبار السويتش موصل بـ A0 مثلا) .

تأمل جيداً في الكود التالي :-

```
void main()
{
trisa.f0=1; trisb=0;
 portb=0b00000001;
while(1) {
if(porta.f0==0) { delay_ms(100); portb=portb<<1;}
            }
}
```

في بداية البرنامج سيضيء الليد الموصل بـ B0 وذلك بسبب وجود الأمر portb=0b00000001; بعد ذلك يختبر البك السويتش الموصل بـ A0 وذلك من خلال جملة if الموجودة داخل جملة while وبالتالي سيكرر البك عملية الاختبار باستمرار وفي حالة تم الضغط على السويتش سيقوم البك بعمل إزاحة لمحتويات المسجل portb من خلال الأمر portb=portb<<1; . وطبعا لابد من كتابة الأمر delay_ms(100); وذلك لكي لا ينفذ البك جملة if عدة مرات خلال ضغطة واحدة فقط فيكرر عملية الإزاحة وهذه النقطة تطرقنا إليها كثيراً .

الآن لدينا مشكلة في الكود السابق ، فتخيل معي ماذا سيحصل عندما يضغط المستخدم على السويتش أول ضغطة ؟ بالتأكيد سيتزحزح المسجل portb وبالتالي سيضيء الليد B1 بدلا من B0 ثم إذا ضغط المستخدم مرة أخرى على السويتش سيضيء الليد B2 وستنطفئ باقي الليدات .. وهكذا إلى أن يضيء الليد

الأخير وهو B7 بعد ذلك عند الضغط على السويتش مرة أخرى ستحدث عملية إزاحة لليسار مرة أخرى وبالتالي سنجد أن جميع الليدات مطفئة ، وهنا تحدث المشكلة ..!! لماذا.. ؟

لأنه إذا ضغطنا بعد ذلك على السويتش مرة أخرى فستظل جميع الليدات مطفئة لأن عملية الإزاحة ستتم للمسجل portb الذي يساوي في ذلك الوقت 0b00000000 وعملية الإزاحة له سيكون ناتجها 0b00000000 وهكذا لن تؤثر عملية الإزاحة على الليدات فستظل جميعا مطفئة .

لكي نتغلب على هذه المشكلة لابد من وضع جملة if تقوم باختبار هل قيمة portb=0b10000000; فإذا تحقق ذلك الشرط ينتظر السويتش إلى أن يتم الضغط عليه فإذا تم الضغط عليه يجعل قيمة portb تساوي 0b00000001 كما في الكود التالي :

```
void main()
{
trisa.f0=1; trisb=0;
 portb=0b00000001;
while(1) {
if(porta.f0==0) { delay_ms(100); portb=portb<<1;}
if(portb=0b10000000) { while(porta.f0==1){ } portb=0b00000001; delay_ms(100)}
            }
}
```

هل لاحظت السطر الذي تم إضافته إنه سيختبر قيمة portb هل وصلت إلى أن آخر ليد B7 هو الذي سيضيء فقط فإذا تحقق ذلك ينتظر إلى أن يتم الضغط على السويتش . السؤال هو كيف سينتظر ؟ ببساطة من خلال جملة while حيث ستختبر هل السويتش لم يتم الضغط عليه (أي تحقق الشرط) حينها يقوم البك بتنفيذ ما بداخل القوسين الخاصين بجملة while ونلاحظ أن القوسين فارغان { } ، ولا يوجد بهما أوامر لتنفيذها بعد ذلك سيعيد البك اختبار الشرط فيجد أنه مازال متحققاً فينفذ ما بداخل القوسين (أي لا ينفذ شيء) ثم يختبر الشرط مرة أخرى .. وهكذا إلى أن تأتي اللحظة التي يضغط

فيها المستخدم على السويتش حينها لن يتحقق الشرط وبهذا يكون قد انتهى البك من تنفيذ جملة while فينتقل للأمر التالي ، ومعنى هذا وأن جملة while هذه التي وضعناها هي بمثابة حاجز أو مانع يمنع البك من تنفيذ الأمر الذي يليها إلى أن يتم الضغط على السويتش (وهذه الفكرة أشرنا إليها سابقا) .

الآن بعد الضغط على السويتش سيقوم البك بجعل قيمة portb=0b00000001; ثم ينتظر البك لمدة ١٠٠ ملي ثانية من خلال أمر التأخير . ثم ينتقل البك مرة أخرى إلى مابداخل جملة while(1) ويكرر تنفيذه باستمرار .

من المؤكد أنك تعلم أهمية أمر الإنتظار الأخير فهو يجعل البك لا ينفذ أكثر من أمر في الضغطة الواحدة . وهنا يجدر الإشارة إلى أنه يمكن استبدال هذا الأمر بجملة while تجعل البك ينتظر إلى أن يتم رفع اليد من على السويتش أي ستكون مانع للبك تمنعه من الإنتقال للأمر التالي إلا بعد أن تصبح قيمة A0 تساوي صفر أي بعد رفع يد المستخدم من على السويتش حيث يتم كتابة الكود كما يلي :

```
void main()
{
trisa.f0=1; trisb=0;
 portb=0b00000001;
while(1) {
if(porta.f0==0) { while(porta.f0==0){ } portb=portb<<1;}
if(portb=0b10000000) { while(porta.f0==1){ } portb=0b00000001; while(porta.f0==0){ } }
        }
}
```

إذن عندما يضيء آخر ليد B7 سيقوم البك بالإنتظار إلى أن يتم الضغط على السويتش من خلال الأمر

while(porta.f0==1){ } فإذا تم الضغط السويتش يضيء أول ليد B0 فقط ثم ينتظر إلى أن يتم رفع اليد من على السويتش من خلال الأمر

```
while(porta.f0==0){ }
```

لعلك تعلم جيداً أنه مع استخدام السويتشات فإنه يجب علينا وضع أمر الإنتظار أو وضع جملة while

ولكن ما الفرق الذي سيلاحظه المستخدم بنيهما ؟

الفرق هو أنه في حالة أمر الإنتظار delay إذا ضغط المستخدم على السويتش لمدة طويلة فسيلاحظ أن الليد الأول يضيء ثم بعد مدة بسيطة يضيء الذي يليه ثم الذي يليه .. وهكذا . أي أن البك ينفذ الأوامر التي كتبناها له الأمر تلو الآخر كأننا ضغطنا على السويتش عدة مرات مع وجود فارق زمني بين تنفيذ هذه الأوامر .

أما في حالة استخدام جملة while فإذا ضغط المستخدم على السويتش لمدة طويلة فلن يلاحظ المستخدم أي تغيير على الليدات إلا بعد أن يرفع يده من على السويتش مهما طالت هذه المدة .

إذن قد يكون أمر الإنتظار أفضل في بعض الأحيان وقد يكون أمر while أفضل في بعض الأحيان على حسب التطبيق المراد ، وهذه نقطة مهمة يجب الانتباه إليها .

التجربة (20)

في هذه التجربة سنطور المشروع السابق أكثر فسنجعل هناك مفتاحين (سويتشين) سويتش يقوم بتحريك الليد لليسار وسويتش يقوم بتحريكه لليمين . أو بمعنى آخر سويتش يقوم بعمل إزاحة لليسار وسويتش يقوم بعمل إزاحة لليمين ، وسنقوم بتوصيل السويتش الأول بـ A0 والسويتش الثاني بـ A1 .

انظر للكود التالي :

```
void main()
{
trisa.f0=1;   trisa.f1=1; trisb=0;
 portb=0b00000001;
while(1) {
if(porta.f0==0) { delay_ms(100); portb=portb<<1;}
if(porta.f1==0) { delay_ms(100); portb=portb>>1;}
            }
}
```

نفهم من الكود أنه عند الضغط على السويتش A0 فستم عمل إزاحة لليسار و عند الضغط على السويتش A1 فستم عمل إزاحة لليمين ، وبالطبع يجب وضع أمر الإنتظار . ولكن لدي سؤال : ما الفرق بين أن نكتب أمر الإنتظار أولاً ثم نكتب بعده أمر الإزاحة ، وبين أن نكتب أمر الإزاحة أولاً ثم نكتب أمر الإنتظار ؟؟

ببساطة في الحالة الأولى سينتظر البك لمدة ١٠٠ ملي ثانية وبعد ذلك يقوم بعمل إزاحة .

أما في الحالة الثانية فيقوم البك بعمل إزاحة ثم ينتظر لمدة ١٠٠ ملي ثانية .

ولكن عند التجربة قد لا تلاحظ ذلك جيداً إلا إذا كان زمن الإنتظار كبير نسبيا كأن يكون نصف ثانية أو أكثر عندها ستلاحظ ذلك بمنتهى الوضوح .

تأمل في الكود السابق جيداً فستلاحظ أنه لدينا العديد من المشاكل فماذا سيحدث إذا كانت قيمة PORTB تساوي 0B00000001 وبعد ذلك تم الضغط على السويتش A1 أي سيتم عمل إزاحة لليمين من المؤكد أن جميع الليدات سوف تنطفئ وبالتالي إذا ضغطنا بعدها على أي سويتش A0 أو A1 فلن نلاحظ أي تغيير وستظل جميع الليدات مطفئة (لأن عملية الإزاحة للأصفار ناتجها أصفار سواءاً كانت الإزاحة لليمين أو لليسار).

إذن لا بد ومن وضع جملة if تختبر هل قيمة portb تساوي 0b00000001 فإذا تم الضغط على السويتش A1 حين ذلك يقوم البك بجعل قيمة portb تساوي 0b10000000 .

أيضاً و بنفس الفكرة عندماتكون قيمة portb تساوي 0b10000000 لابد أن يختبر البك هل تم الضغط على السويتش A0 حين ذلك يقوم البك بجعل قيمة portb تساوي 0b000000001 .

```
void main()
{
trisa.f0=1;   trisa.f1=1; trisb=0;
 portb=0b00000001;
while(1) {
if(porta.f0==0) { portb=portb<<1; delay_ms(100); }
if(porta.f1==0) { portb=portb>>1; delay_ms(100); }

if(portb==0b00000001) {if(porta.f1==0) {portb=0b10000000; delay_ms(100); }
if(portb==0b10000000) { if(porta.f0==0)  { portb=0b00000001; delay_ms(100); }
          }
}
```

هذا الحل غير كافي لأنه في بداية البرنامج ستكون قيمة portb=0b00000001 وبالتالي ستتحقق جملة if التي تختبر هذا الشرط if(portb=0b00000001) وداخل جملة if هذه يوجد جملة if أخرى تختبر هل تم الضغط على السويتش في تلك اللحظة فإذا لم يتم الضغطة في تلك اللحظة (وبالطبع هذا ما سيحدث لأن الأوامر تتنفذ في غاية السرعة) إذن سينتقل البك للأوامر التالية . وهنا يحتمل حدوث مشكلة فإذا ضغط المستخدم على السويتش A1 في اللحظة التي ينفذ فيها البك الأمر

```
if(porta.f1==0) { portb=portb>>1; delay_ms(100); }
```

أو بمعنى أدق أثناء مدة الضغط نفذ البك الأمر السابق فإن قيمة portb ستساوي 0b00000000 وتحدث المشكلة المعهودة لذلك لابد من تغيير الكود السابق ليصبح كما يلي :

```
void main() {
trisa.f0=1;   trisa.f1=1; trisb=0;    portb=0b00000001;
while(1) {
if(porta.f0==0) { portb=portb<<1; delay_ms(100); }
if(porta.f1==0) { portb=portb>>1; delay_ms(100); }
loop:
if(portb==0b00000001) {if(porta.f1==0) {portb=0b10000000; delay_ms(100); }
                       if(porta.f0==0) {portb=0b00000010; delay_ms(100); }
                      goto loop;
                      }
if(portb==0b10000000) {if(porta.f1==0) {portb=0b01000000; delay_ms(100); }
                       if(porta.f0==0)  { portb=0b00000001; delay_ms(100); }
                      goto loop;
                      }
        }
}
```

هنا في حالة كانت قيمة portb=0b00000001 فإن البك سوف يدخل في دوامة لا يخرج منها (loop) لا يخرج منها إلا إذا تغيرت قيمة portb ففي هذه الحالة يخرج من هذه الدوامة (أو الحلقة المغلقة).

ولكن ماذا يعني هذا الكلام ؟؟ إنه يعني أنه في حالة كانت portb=0b00000001 فإن البك سينفذ أوامر معينة ويعيد تنفيذها باستمرار إلى أن تتغير قيمة portb وهذه الأوامر التي ينفذها البك باستمرار هي عبارة عن جملتي if أول جملة منهم تختبر هل تم الضغط على السويتش A1 وعند تحقق هذا الشرط ستصبح قيمة PORTB تساوي 0B100000000 وجملة if الأخرى تختبر هل تم الضغط على السويتش الآخر A0 فإذا تحقق الشرط سوف تصبح قيمة portb تساوي 0b00000010 أي يتم عمل إزاحة لليسار. أما في حالة عدم الضغط على أي سويتش فإن البك ينفذ الأمر الذي يلي جملتي if السابقتين وهو الأمر goto loop; والذي سيجعل البك يكرر عملية إختبار السويتشات .

وبنفس الطريقة إذا كانت قيمة portb=0b10000000 .

يمكنك فهم ما سيحدث فقط تأمل الكود من جديد عدة مرات فهي فكرة برمجية جيدة ستساعدك كثيراً فيما بعد بإذن الله تعالى .

والآن لنشرح العلامة and و or || و | وعلامة النفي ! و ~

التجربة (١٨)

في هذه التجربة سنشرح فكرة توليد نغمات من السماعة

لنطور المشروع أكثر ونجعل هناك ثلاثة سويتشات كل سويتش يقوم بعمل نغمة معينة

لماذا لا نجعلهم سويتش واحد كل ضغطة عليه تجعله يغير النغمة

لنقوم بعمل شيء شبيه بالبيانو .

كيفية التعامل مع السفن سيجمنت منفردة .

استخدام اكثر من واحدة . (اثنين ثم ثلاثة) ثم أكثر وأكثر باستخدام طريقة multiplexing وطريقة اخرى باستخدام دائرة متكاملة مساعدة أي بدون multiplexing

فكرة مشروع تايمر يشرح كيف تشغل جهاز بعد مدة معينة وباستخدام سفن سيجمنت

شرح استخدام الشاشة LCD

تحريك الكلام في الشاشة LCD بأشكال مختلفة .

.في هذه التجربة سيكون فيها ثلاث سويتشات كل سويتش يضيء الليدات بشكل مختلف

ممكن تحط فكرة جملة FOR التي بداخلها انتظار وقت بسيط يتم تكراره وفي داخله اختبار السويتشات .

التجربة (٢١)

في هذه التجربة والتجارب التي تليها بإذن الله سنتعرف أكثر على كيفية استخدام البك مع العناصر الالكترونية والكهربية الأخرى فالمطلوب من هذه التجربة هو إصدار صوت (مثل صوت الصافرة) باستخدام Buzzer أو Bleeper .

عندما نريد أن نتحكم في أي جهاز أو قطعة الكترونية أو كهربية فإننا نحتاج لمعرفة نقطتين رئيسيتين :-

١- الجهد الذي يحتاجه هذا الجهاز (متردد أم مستمر) وما هو مقداره .

٢- التيار الذي يحتاجه لكي يؤدي أفضل النتائج .

فصديقنا العزيز (البك) له إمكانيات محدودة حيث أنه يستطيع إخراج جهد مستمر لا يزيد عن خمسة فولت وبشدة تيار قصوى 25 ملي أمبير ، ولكن مع ذلك نستطيع باستخدام البك أن نتحكم في أجهزة تعمل بـ ٢٢٠ فولت وتيار 5000 ملي أمبير (خمسة أمبير) بل وأكثر من ذلك ولكن بمساعدة قطع أخرى مثل الترانزستور و الريلاي Relay و ... .

وحتى لا نطيل في التفاصيل النظرية ، هيا بنا لنطبق العديد من التجارب والتي من خلالها سنتحكم في العديد من الأجهزة ، كل جهاز يحتاج جهد معين و تيار معين لكي يعمل بكفاءة .

في هذه التجربة المطلوب تشغيل Buzzer .

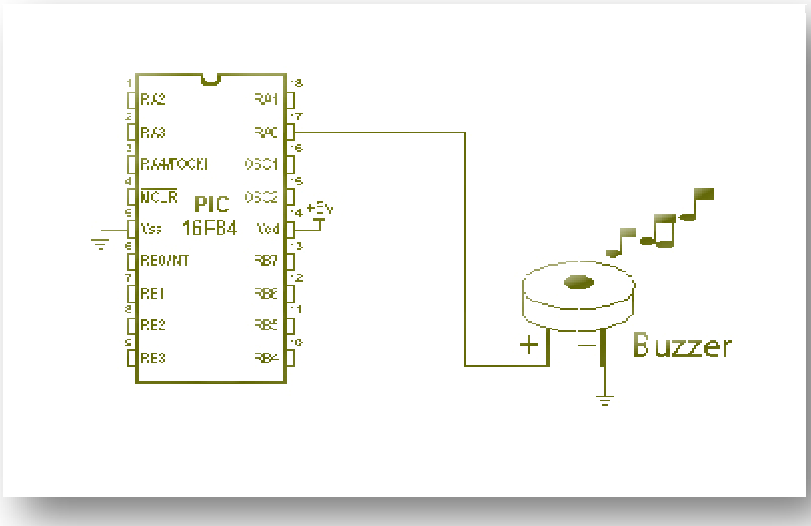
والـ Buzzer عبارة عن قطعة لها طرفان طرف موجب وطرف سالب ، هذه القطعة تصدر صوت صافرة ومن الأنواع الشهيرة ذلك النوع الذي يصدر صوت صافرة إذا طبق على طرفيه الجهد المطلوب ويسمى هذا لنوع بـ level mode Buzzer . لأنه يحتاج لمستوى معين من الجهد لكي يعمل وهذا النوع هو ما سنستخدمه في هذه التجربة . (كما يوجد قطعة أخرى تسمى Beeper تصدر صوتًا إذا طبق على طرفيها الجهد المناسب فهي تشبه الـ level Buzzer مع وجود اختلافات بسيطة) ويوجد أنواع أخرى من الـ Buzzer لا تصدر صوتاً إلا إذا تم إرسال لها نبضات (موجة مربعة) بتردد معين وهذا النوع يسمى بـ frequency mode

Buzzer كما أنه بتغيير التردد يتغير الصوت الصادر من الBuzzer و سنتطرق إلى هذا النوع فيما بعد بإذن الله تعالى (راجع ملحق اساسيات كهربية والكترونية لمزيد من المعلومات)

والـ Buzzer عمومًا يعمل بجهد معين وتيار معين ويمكننا معرفة الجهد والتيار من خلال الكتالوج أو datasheet أو أحيانا مما هو مكتوب على القطعة نفسها . وهذه بعض أشكاله :

إذا كان الـ Buzzer الذي سنستخدمه يحتاج إلى جهد ٥ فولت وتيار يساوي ٢٥ ملي أمبير أو أقل فإنه خاضع لقدرات البك العادية وفي هذه الحالة نقوم بتوصيله توصيل مباشر ولن نحتاج إلى قطع إلكترونية أخرى مثل الترانزستور ولهذا سنقوم بالتوصيل كما بالشكل :

انتبه لقطبية الـ Buzzer أثناء التوصيل



وفي هذه الحالة فإنه لكي يعمل الـ Buzzer سنقوم بتطبيق جهد موجب على الطرف الموصل به . وفي الرسم السابق تم التوصيل بالطرف A0 لذلك سنكتب الكود التالي .

```
Porta.f0=1;
```

عندها سيصدر الـ Buzzer صوتا يشبه صوت الصافرة ويظل مصدراً ذلك الصوت طالما كان الجهد مطبق عليه ، وإذا أردنا أن نوقف هذا الصوت بعد مدة معينة كل ما علينا فعله هو جعل A0 يساوي صفر

Porta.f0=0;

وإن من التطبيقات الجميلة لـ Buzzer هو أن يتم استخدامه مع السويتشات بحيث عندما يضغط المستخدم على السويتش يسمع صوت صافرة مع كل ضغطة ، فالهدف غالبًا هو لفت انتباه الأشخاص ، وإذا أردت تطبيق ذلك في مشاريعك فعليك الانتباه إلى شيء مهم وهو جعل صوت الصافرة غير مستمر ويجب إيقافه بعد مدة معينة لكي لا يكون الأمر مزعجاً(بالطبع إلا إذا كان الهدف من المشروع هو الإزعاج ...!!!) كما يفضل أن تجعل الصوت مستمراً طالما كان المستخدم ضاغطاً على السويتش فإذا رفع يده من على السويتش يتوقف الصوت . والآن لنكتب برنامج (عداد يعد من ٠ إلى ٢٥٥) وذلك عن طريق توصيل ثمان ليدات بالأطراف من B0 إلى B7 ، وسويتش عند الضغط عليه يقوم بزيادة الرقم مع إصدار صوت عند كل ضغطة (السويتش موصل بـ A1 والـ Buzzer موصل بـ A0).

```
void main()
{
trisa.f0=0; trisa.f1=1;  trisb=0;    portb=0;
while(1) {
        if(porta.f1==0) {
                porta.f0=1;          // Buzzer ON
                portb++;
                while(porta.f1==0){ }        // wait
                porta.f0=0;          // Buzzer OFF
                }
        }
}
```

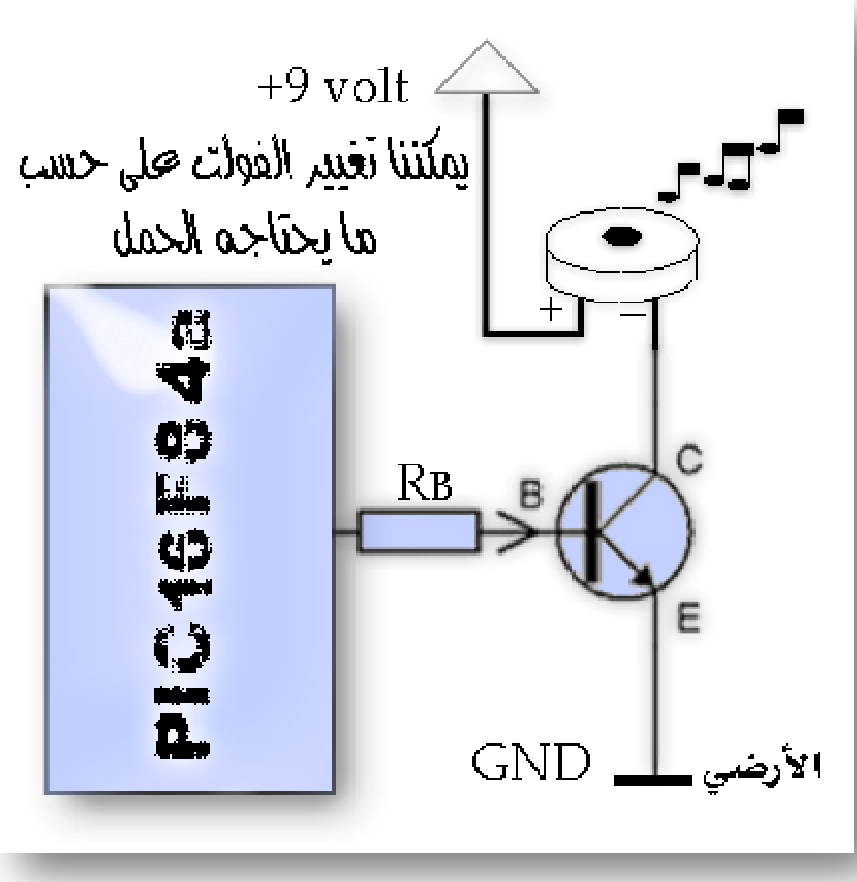
قم بتشغيل الـBuzzer وبالتالي سنسمع صوت الصافرة

انتظر إلى أن يتم رفع اليد من على السويتش فإذا حدث ذلك انتقل للأمر التالي وهو الأمر الذي سيطفئ الـ Buzzer وبالتالى سيتوقف الصوت

بنفس الطريقة تماما التي طبقناها مع الـ Buzzer يمكننا تطبيقها مع الـ Bleeper .

إذن الأمر في غاية البساطة كل ماعلينا فعله هو تطبيق جهد موجب لتشغيل الـ Buzzer وتطبيق جهد صفر لإطفاء الـ Buzzer . والآن هيا بنا لنتعرف على الطريقة التي سنستخدمها لتشغيل وإطفاء أي جهاز يعمل بجهد أكبر من ٥ فولت . فبعض أنواع الـ Buzzer يحتاج إلى ٩ فولت وبعضها إلى ١٢ فولت فما هو الحل ؟؟

بالنسبة للكود فهو نفس الكود السابق كل ما سنغيره هو الهاردوير والحل الأسهل هنا هو استخدام الترانزستور كمفتاح . transistor as a switch كما بالشكل .



(راجع ملحق أساسيات كهربية والكترونية في آخر الكتاب لمعرفة تفاصيل هذه الطريقة والقيمة المناسبة للمقاومة RB)

من الرسم يتضح أننا قمنا بتوصيل قاعدة الترانزستور Base بطرف المايكروكنترولر فإذا طبقنا جهد موجب على هذا الطرف سيكون الترانزستور كأنه مفتاح مغلق يقوم بالتوصيل بين المجمع collector والباعث emitter وبالتالي سيعمل الـ Buzzer ، وعند تطبيق جهد صفر سيكون الترانزستور كأنه مفتاح (مفتوح) open circuit وبالتالي سينطفئ الـ Buzzer .

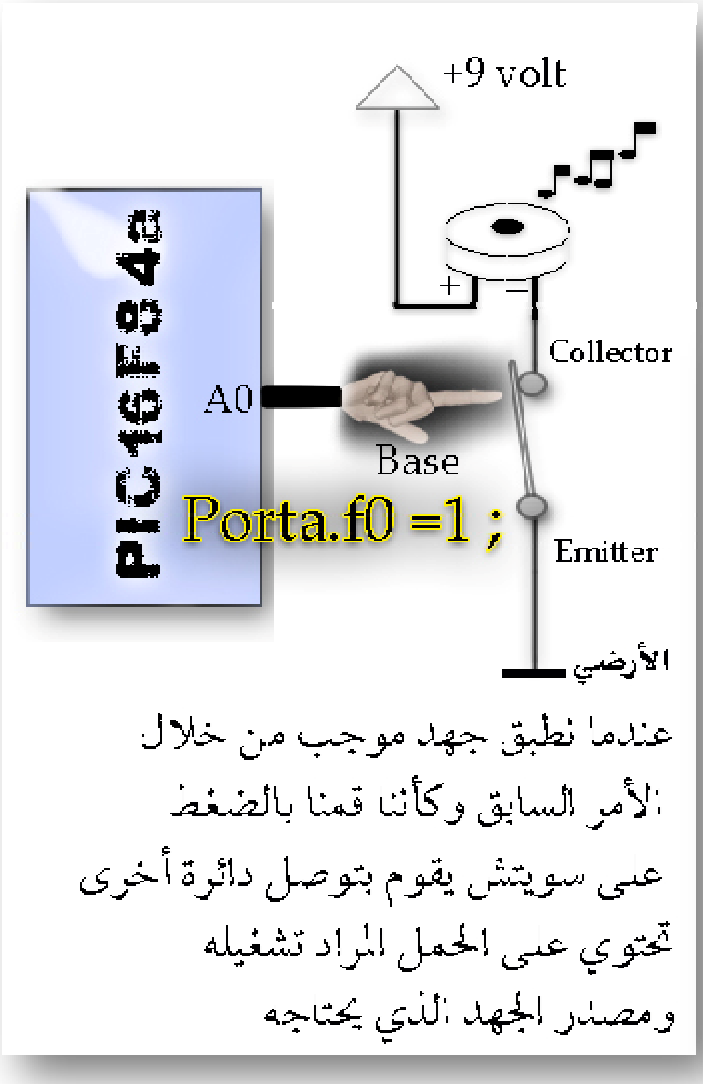
هل لك أن تتخيل معي كيف يعمل الترانستور كمفتاح ؟؟ إذن فلتتخيل كأن هناك دائرة تحتوي على مصدر جهد ٩ فولت مثلا وتحتوي أيضًا على Buzzer و على سويتش وبالطبع إذا تم الضغط على السويتش ستعمل الـ Buzzer وتصدر صوتاً . إذن نحن نحتاج أن يكون للبك يداً يستطيع أن يضغط بها على هذا السويتش إذا كتبنا له الأمر ;porta.f0=1 وهذه اليد تمكنه أيضاً من جعل السويتش أو المفتاح مفتوح فتصبح الدائرة مفتوحة فلا يعمل الـ Buzzer ولا يصدر صوتاً إذا كتبنا له الأمر ;porta.f0=0 .

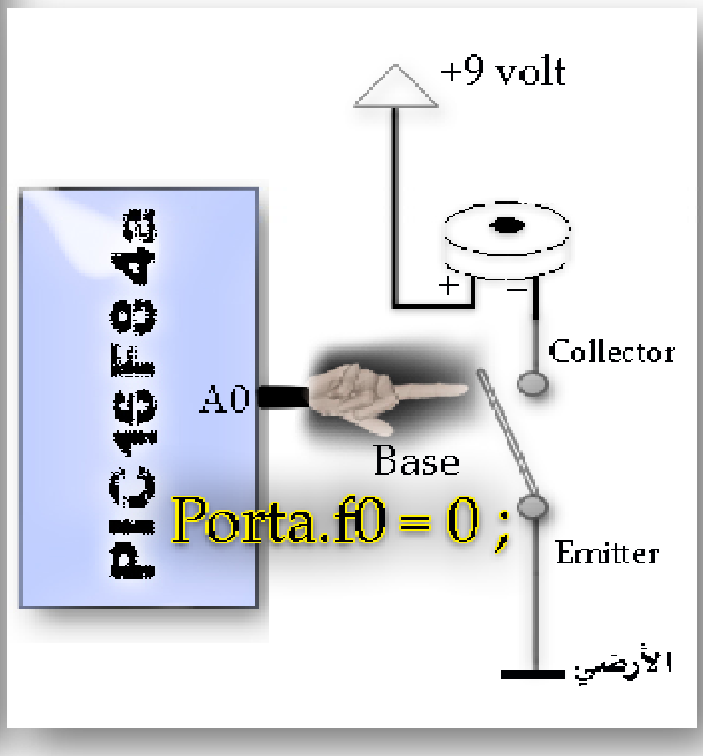
ولكن كيف نعطي هذه اليد للبك ؟؟ ببساطة قم بإضافة ترانزستور وقم بتوصيله كما بالشكل السابق فهو سيقوم بهذه المهمة وكأنه تلك اليد التي تريدها ، وبمعنى أدق فإن الترانزستور هو اليد والسويتش في نفس الوقت وهذه اليد يمكنك أن تجعلها تضغط على السويتش بتطبيق جهد موجب على القاعدة Base ويمكنك أن تقوم بجعل هذه اليد تجعل السويتش مفتوح بتطبيق جهد صفر على القاعدة كيف أطبق جهد موجب على القاعدة ؟

من خلال الأمر ;porta.f0=1

كيف أطبق جهد صفري على القاعدة ؟

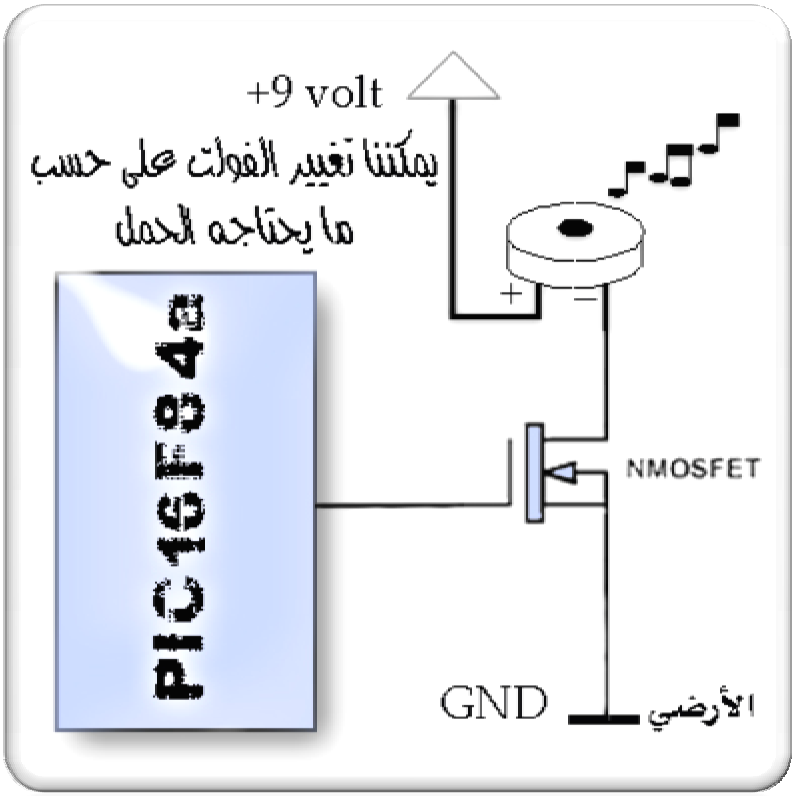
من خلال الأمر ;porta.f0=0 .





أيضا في حالة كان الحمل (هنا هو الـ Buzzer) يحتاج إلى تيار أكبر من ٢٥ ملي أمبير سنستخدم نفس الطريقة السابقة لأن الترانزستور سيعمل على تكبير التيار .

يمكننا أيضا أن نقوم باستخدام ترانزستور من نوع آخر مثل ترانزستور الـ MOSFET كما يوضح ذلك الشكل التالي :



وبنفس الطريقة إذا كان الترانزستور موصل بالبك بالطرف A0 مثلاً فإنه عند كتابتنا للأمر porta.f0=1; سيعمل الترانزستور وكأنه مفتاح مغلق وبالتالي سنسمع صوت من الـBuzzer وعندما نكتب الأمر porta.f0=0; سيصبح الترانزستور وكأنه مفتاح مفتوح (open circuit) وبالتالي لن نسمع أي صوت من الـ Buzzer .

لمعرفة الفرق بين الترانزستورات العادية BJT وترنزستور الـ MOSFET عند استخدامهم كمفتاح أو كسويتش راجع الجزء الخاص بأساسيات كهربية والكترونية .

التجربة (٢٢)

في هذه التجربة سنتعلم كيف نصدر نغمات صوتية من السماعة speaker أو من الـ Buzzerالذي يعمل بالترددات frequency mode Buzzer أو من piezo transducer فكل هذه القطع تصدر صوتًا معينًا إذا تم إرسال إليها نبضات بتردد معين . (راجع الملحق الخاص بأساسيات كهربية والكترونية للتعرف أكثر على هذه الأنواع)

في مثالنا هذا سنستخدم سماعة عادية مثل سماعة الراديو التي تسمى loudspeaker أو speaker .

بعض أشكال الـ speaker

هنا لكي نصدر صوت معين سنقوم بإرسال تردد معين وليس مجرد تطبيق جهد على طرفي السماعة . ولكن كيف سنقوم بإرسال تردد معين ؟؟ ببساطة سنقوم بتطبيق جهد موجب على أحد أطراف البك ثم نجعل البك ينتظر قليلا لمدة معينة ثم نقوم بتطبيق جهد صفر على نفس الطرف ثم نجعل البك ينتظر لمدة معينة ونكرر ذلك باستمرار ، وبهذا قمنا بعمل موجة مربعة (أو أرسلنا نبضات) لها تردد معين هذا التردد يعتمد على زمن الانتظار الذي كتبناه في الكود .

فعندما نكتب الأمر portb.f0=1; ثم الأمر ; delay_ms(5) ثم نكتب الأمر portb.f0=0; ثم الأمر delay_ms(5); ونكرر ذلك باستمرار عن طريق جعل هذه الأوامر داخل جملة while(1) { } فإننا بذلك قمنا بإرسال نبضات مستمرة بتردد معين . وبتغيير الرقم خمسة الموجود في أمر الإنتظار فإننا بذلك نقوم بتغيير تردد تلك النبضات . ولكن ما هو تردد هذه النبضات في الكود السابق وكيف يتم حسابه ؟؟

قبل أن ندخل في أي عمليات حسابية السؤال هو لماذا نريد معرفة التردد ؟؟ ببساطة لأن الـ speaker يعمل في نطاق ترددات معينة لكي ينتج النغمة الصوتية المطلوبة وكذلك الـ piezo transducer وبعض أنواع الـ Buzzer ، جميل وماذا أيضا ؟؟ فيما بعد ستقابلك العديد من الأشياء التي تحتاج لتردد معين لكي تعمل غير السماعات فبعض المواتير لكي تتحكم فيها تحتاج لتردد معين ... إذن عملية التحكم في التردد ومعرفته حسابياً أمر مهم . إذن هيا بنا نتعرف على طريقة حساب التردد :

الآن عندما نقوم بإرسال نبضة عن طريق الأمرين ; portb.f0=1; delay_ms(5) بهذين الأمرين قمنا بإخراج جهد موجب لمدة خمسة ملي ثانية أو بتعبير آخر فإننا قمنا بإرسال نبضة Hi أو نبضة واحد مقدارها خمسة ملي ثانية . جميل وبنفس الفكرة فإننا عندما نكتب الأمرين portb.f0=0; delay_ms(5); فإننا قمنا بتطبيق جهد صفري لمدة خمسة ملي ثانية أو بتعبير آخر فإننا قمنا بإرسال نبضة low أو نبضة صفر مقدارها خمسة ملي ثانية . إذن العملية السابقة يمكننا أن نقول عنها أننا نرسل وحايد وأصفار .

أرسلنا واحد لمدة ٥ ملي ثانية ثم أرسلنا صفر لمدة ٥ ملي ثانية إذن المجموع هو ١٠ ملي ثانية ، أي أن الزمن الذي استغرقته الدورة الكاملة يساوي ١٠ ملي ثانية ومعنى زمن الدورة الكاملة أي زمن نبضة الواحد + زمن نبضة الصفر اللذان يتكرران باستمرار ، و هذا الزمن هو زمن الدورة الكاملة أو ما يسمى بالزمن الدوري . جميل بهذا قد انتهينا من الشرح ...!!

لا لا انتظر لم ننتهي بعد ، فهدفنا هو حساب التردد وليس الزمن الدوري .!!

إذا كان لديك الزمن الدوري فبكل بساطة ستحصل على التردد يا صديقي فقط قم بقسمة الرقم واحد على الزمن الدوري الذي حسبته .

إذن التردد في هذا المثال يساوي : $\frac{1}{10\text{ملي ثانية}}$ وطبعاً كلمة ملي تعني أننا سنضرب القيمة في 10^{-3}

إذن التردد يساوي $\frac{1}{10 \times 10^{-3}}$ = ١٠٠ هيرتز .

والموجة هذه (واحد ثم صفر ثم واحد ثم ....) يمكننا أن نرسمها كما بالشكل التالي :

وإليك الآن عزيزي القارئ بعض الأمثلة :

مثال ١ : في البرنامج التالي احسب التردد الخاص بالموجة الناتجة من الطرف B0

```
void main()
{  trisb=0;    portb=0;
while(1) {
                portb.f0=1;  delay_ms(3);
                portb.f0=0; delay_ms(3);
            }
}
```

الحل : الطرف B0 سيكون بـ واحد لمدة ٣ ملي ثانية ثم يكون بـ صفر لمدة ٣ ملي ثانية ويتكرر ذلك باستمرار

إذن الزمن الدوري في هذه الحالة = ٣ + ٣ = ٦ ملي ثانية

إذن التردد = $\frac{1}{6 \times 10^{-3}}$ = ١٦٦,٦٦٧ هيرتز

مثال ٢ : في البرنامج التالي احسب التردد الخاص بالموجة الناتجة من الطرف B0

```
while(1) {
        portb.f0=1;  delay_ms(4);
        portb.f0=0; delay_ms(1);
    }
```

الحل : الطرف B0 سيكون بـ واحد لمدة ٤ مل ثانية ثم يكون بـ صفر لمدة ملي ثانية ويتكرر ذلك باستمرار إذن الزمن الدوري في هذه الحالة = ٤ + ١ = ٥ ملي ثانية .

إذن التردد = $\frac{1}{5 \times 10^{-3}}$ = ٢٠٠ هيرتز .

ملاحظة : ليس شرطاً أن تكون مدة النبضة واحد تساوي مدة النبضة صفر المهم هو أنهم يتكرروا باستمرار فيكون مجموعهم هو الزمن الدوري .

أظن أن الأمر في غاية البساطة والسهولة ،،

عزيزي القارئ ،، بهذا تعرفنا على الطريقة التي نحسب بها التردد من خلال قراءتنا للكود أو من خلال معرفتنا لزمن النبضات ، ولكن ليس هذا هو المطلوب ، فالمطلوب هو أن نقوم بالعملية العكسية بمعنى أننا يطلب منا أن نقوم بإصدار تردد معين من طرف معين وعلى أساس قيمة هذا التردد نحدد زمن النبضات أو الزمن الدوري ، ، وليتضح الأمر إليك الأمثلة التالية :

مثال ٣ : قم بكتابة برنامج يجعل البك ينتج تردد مقداره ١٠٠ هرتز من خلال الطرف B0 .

الحل : المعلومة التي نحتاجها الآن هي الزمن الدوري ، والزمن الدوري = $\frac{1}{\text{التردد}}$. هذه هي المعلومة الجديدة التي سنحتاجها . إذن الزمن الدوري في هذه الحالة = $\frac{1}{100}$ = 0.01 = 10×10^{-3} = ١٠ ملي ثانية

جميل جداً ،، الآن الزمن الدوري يساوي ١٠ ملي ثانية إذن يمكننا أن نجعل مدة النبضة واحد = ٥ ملي ثانية ومدة النبضة صفر تساوي ٥ ملي ثانية كما بالكود التالي :

```
while(1) {
        portb.f0=1;  delay_ms(5);
        portb.f0=0; delay_ms(5);
    }
```

كما يمكننا أن نجعل مدة النبضة واحد = ٨ ملي ثانية ومدة النبضة صفر تساوي ٢ ملي ثانية فيكون مجموعهم ١٠ ملي ثانية ، وأيضا يمكننا أن نجعل مدة النبضة واحد = ٦ ملي ثانية ومدة النبضة صفر تساوي ٤ ملي ثانية .. وهكذا (المهم أن يكون الزمن الدوري يساوي ١٠ ملي ثانية) .

في كل الحالات السابقة سيكون التردد واحد وما يختلف هو مدة كل نبضة .

إذن كان يمكننا أن نكتب الكود بالشكل التالي :

```
while(1) {
        portb.f0=1;  delay_ms(8);
        portb.f0=0; delay_ms(2);
    }
```

أو الشكل التالي :

```
while(1) {
        portb.f0=1;  delay_ms(6);
        portb.f0=0; delay_ms(4);
    }
```

أو بأشكال أخرى كثيرة ، المهم أن يكون الزمن الدوري = ١٠ ملي ثانية وبالتالي سيكون التردد في كل هذه الحالات يساوي ١٠٠ هيرتز .

معلومات مهمة لابد منها : لعلك لاحظت في الأمثلة السابقة أن التردد الناتج هو تردد صغير نسبيًا فكيف نحصل على ترددات كبيرة مثل ١٠ آلاف هيرتز أو ٥٠ ألف هيرتز ونحو ذلك من الترددات الكبيرة . إذا فكرت قليلاً فستجد أنه لا يمكننا فعل ذلك باستخدام الأمر delay_ms لأنه في هذه الحالة نحن نحتاج لوحدات أقل من الملي ثانية مثل الميكرو ثانية . وللتذكير فإن الملي ثانية هو جزء من ألف جزء من الثانية .

أما الميكروثانية فهو جزء من مليون جزء من الثانية ، أي أن الميكرو ثانية = $\frac{1}{1000000}$ = ١ x 10^{-6} ثانية

والسؤال هنا كيف أجعل البك ينتظر واحد ميكرو ثاينة ؟؟ ببساطة قم بتحويل حرف m إلى u

```
delay_us(1);
```

هذا الأمر يجعل البك ينتظر واحد ميكرو ثانية . أي أن تكرار هذا الأمر مليون مرة يجعل البك ينتظر ثانية واحدة (مجرد توضيح لمعرفة صغر هذه الكمية) .

أظن أن الأمر في غاية الروعة ،، هيا بنا الآن لنتطرق لبعض الأمثلة التي من خلالها سننتج ترددات عالية :

مثال ٤ : قم بكتابة كود يجعل البك ينتج تردد مقداره 10000 هيرتز (عشرة آلاف هيرتز أي ١٠ كيلو هيرتز).

الحل : أولا سنقوم بحساب الزمن الدوري :-

الزمن الدوري = $\frac{1}{10000}$ = ١٠٠ x 10^{-6} ١٠ = ١٠٠ ميكرو ثانية

إذن يمكننا أن نجعل مدة النبضة واحد = ٥٠ ميكرو ثانية ومدة النبضة صفر = ٥٠ ميكرو ثانية

```
while(1) {    portb.f0=1;  delay_us(50);
              portb.f0=0; delay_us(50);          }
```

مثال ٥ : قم بحساب التردد الناتج من الطرف B0 الذي ينتجه البك من خلال الكود التالي :

```
while(1) {
        portb.f0=1;  delay_us(5);
        portb.f0=0; delay_us(5);
 }
```

الحل : الزمن الدوري = زمن النبضة واحد + زمن النبضة صفر = ٥ + ٥ = ١٠ ميكرو ثانية

التردد = $\frac{1}{10\times10^{-6}}$ = ١٠٠ ألف هيرتز = ١٠٠ كيلو هرتز (100 KHz)

أعتقد أن هذا القدر كافي لشرح كيفة انتاج موجة مربعة بتردد معين و كيفية حساب التردد .

الآن هيا بنا لنرجع لموضوعنا الأساسي وهو توليد نغمة صوتية من خلال السماعة :

أولا : كيف نقوم بتوصيل السماعة للمايكروكنترولر ؟

لابد أولا من معرفة النقطتين الرئيسيتين : ١- الجهد التي تحتاجه السماعة ٢- التيار .

إذا كانت السماعة تعمل بجهد يساوي ٥ فولت فإننا نقوم بتوصيلها توصيل مباشر و إذا كانت السماعة تعمل بجهد أقل من ٥ فولت فإننا نضع مقاومة قبل السماعة كما كنا نفعل في توصيل الليد ، وإذا كانت تعمل بجهد أكبر من خمسة فولت أو أن التيار الذي تحتاجه أكبر من ٢٥ ملي أمبير فإننا سنحتاج إلى ترانزستور نستخدمه كمفتاح كما فعلنا مع الـ Buzzer.

ملحوظة : بعض أنواع السماعات تحتاج إلى أن نضع قبلها مكثف coupling capacitor مثل سماعات الأذن headphone ، وبعضها يحتاج إلى استخدام مقاومة. (راجع ملحق الأساسيات الكهربية والالكترونية لمعرفة التفاصيل) .

والآن سنقوم بإصدار صوت عن طريق جعل الطرف الموصل به السماعة يصدر تردد معين

اكتب الكود التالي وطبق الدائرة ولاحظ ما ستسمعه :

```
void main()
{ trisb=0;   portb=0;
while(1) {
                portb.f0=1;  delay_ms(1);
                portb.f0=0; delay_ms(1);
            }
}
```

في الكود السابق جعلنا مدة كل نبضة واحد ملي ثانية وهنا الزمن الدوري سيساوي ٢ ملي ثانية وبالتالي عند حساب التردد سنجد أنه يساوي ٥٠٠ هيرتز .

كما أخبرتك سابقاً فإنه عند تغيير التردد فإن الصوت الناتج سيتغير . والآن لنحاول إنتاج صوت آخر بكتابة الكود التالي :

```
void main()
{ trisb=0;   portb=0;
while(1) {
                portb.f0=1;  delay_ms(2);
                portb.f0=0; delay_ms(1);
            }
}
```

هنا سيكون الزمن الدوري يساوي ٢ + ١ = ٣ ملي ثانية وبالتالي عند حساب التردد سنجد أنه يساوي ٣٣٣ هيرتز تقريبا . جرب ذلك وحاول تطبيق الدائرة أو محاكاتها في برنامج proteus ولاحظ تغير الصوت .

أيضا يمكنك استخدام الأمر delay_us وذلك للحصول على ترددات أعلى كما في الكود التالي :

```
void main()
{ trisb=0;   portb=0;
while(1) {
            portb.f0=1;  delay_us(100);
            portb.f0=0; delay_ms(100);
          }
}
```

لعلك تأكدت من خلال الثلاث أكواد السابقة أن كل تردد ينتج صوت معين . الآن نحن نريد أن نجعل الأمر أقل إزعاجاً لأنه في الأكواد السابقة نلاحظ أن الصوت يظل مستمراً ونحن نريد أن يكون لمدة معينة ففي بعض التطبيقات تحتاج لأن تصدر صوت صافرة مثلا لمدة معينة للتنبيه أو التحذير ونحوه .

الكود الآخير الذي كتبناه يصدر صوت صافرة والنبضات الصادرة من البك تتكرر باستمرار نيتجةً لاستخدامنا الأمر while(1) { } ، والمطلوب منا هو أن نقوم بتكرار النبضات ولكن عدد معين من المرات وليس باستمرار إذن لابد أن يخطر ببالنا جملة for فهي حل مناسب جداً لما نريد .

```
void main()
{ int x;
 trisb=0;   portb=0;
for(x=0;x<2000;x++){
                    portb.f0=1;  delay_us(100);
                    portb.f0=0; delay_ms(100);
                  }
}
```

عند محاكاة الكود السابق ستسمع صوت صافرة لمدة صغيرة ثم يتوقف الصوت .

ملاحظة: لقد قمنا بتكرار الموجة ألفين مرة من خلال جملة for ولهذا من الخطأ أن نستخدم متغير من النوع char لأن مداه من صفر إلى ٢٥٥ فقط بل سنحتاج إلى متغير كبير نوعاً ما مثل النوع int .

لاحظ أن النبضة واحد مدتها ١٠٠ ميكرو ثانية وكذلك النبضة صفر أي أن الموجة الواحدة تستغرق ٢٠٠ ميكرو ثانية وبما أن هذه الموجة سوف تتكرر ألفين مرة من خلال جملة for فإن الصوت سوف نسمعه لمدة تساوي تقريبا ٢٠٠ ميكرو X ٢٠٠٠ = ٤٠٠ ملي ثانية أي ما يقارب النصف ثانية .

قم بتغيير الرقم ١٠٠ واجعله ٤٠٠ وستلاحظ أن حدة الصوت قلت فمن المعلوم أنه كلما زاد التردد كلما زادت حدة الصوت ، وهكذا يقول الفيزيائيون ويقولون أن صوت المرأة أحد من صوت الطرف(غالباً) وبالتالي فإن صوت المرأة أعلى في التردد من صوت الطرف .

إذن عند زيادة التردد فمن الطبيعي أن نلاحظ أن حدة الصوت تزداد .

انتبه..! : عندما نقلل من زمن النبضة ( )delay_us فإن التردد يزداد . فالعلاقة عكسية بين الزمن الدوري وبين التردد لأن كل منهما يساوي مقلوب الآخر ، ومن هذا نستنتج أنه لإنتاج تردد عالى فإننا نجعل زمن كل نبضة صغير . ولإنتاج تردد منخفض فإننا نجعل زمن كل النبضة كبير من خلال أمر delay .

والآن هيا بنا إلى رحلة ممتعة وهي رحلة إنتاج النغمات ، الآن عزيزي القارئ أنت مهيأ تماماً لصنع نغمات صوتية أو موسيقية مختلفة لا ينقصك إلا معلومة واحدة بسيطة جداً وهي :

لإنتاج نغمة صوتية فإننا نقوم بإصدار تردد معين لمدة بسيطة ثم نقوم بإصدار تردد أخر لمدة مثل السابقة أو مختلفة عنها ... وهكذا .

والنغمة الصوتية الناتجة تعتمد على شيئين :

١- الترددات المستخدمة .

٢- مدة تكرار الموجة (يعتمد على الرقم الذي نضعه في جملة for)

وإليك الآن هذا المثال الذي ينتج نغمة صوتية معينة والتي ستتكرر باستمرار :-

```
void main()
{ int x;
 trisb=0;   portb=0;
while(1){
for(x=0;x<300;x++){ portb.f0=1; delay_us(700);
                portb.f0=0; delay_us(700);}
for(x=0;x<300;x++){ portb.f0=1; delay_us(750);
                portb.f0=0; delay_us(750);}
for(x=0;x<250;x++){ portb.f0=1; delay_us(800);
                portb.f0=0; delay_us(800);}
for(x=0;x<220;x++){ portb.f0=1; delay_us(750);
                portb.f0=0; delay_us(750);}
for(x=0;x<400;x++){ portb.f0=1; delay_us(700);
                portb.f0=0; delay_us(700);}
for(x=0;x<420;x++){ portb.f0=1; delay_us(600);
                portb.f0=0; delay_us(600);}
for(x=0;x<350;x++){ portb.f0=1; delay_us(650);
                portb.f0=0; delay_us(650);}
for(x=0;x<320;x++){ portb.f0=1; delay_us(650);
                portb.f0=0; delay_us(650);}
for(x=0;x<300;x++){ portb.f0=1; delay_us(700);
                portb.f0=0; delay_us(700);}
        }
}
```

لا تخف من طول الكود فما هو إلا جملة for متكررة مع تغيير قيمتها وتغيير وقت الانتظار . ويمكنك عزيزي القارئ أن تغير من هذه القيم فتنتج الكثير من النغمات الرائعة والتي بالتأكيد ستكون أفضل من النغمة العشوائية السابقة وهذه نقطة مهمة ، فعللك تلاحظ أن الكثير من ألعاب الأطفال تنتج أصواتاً مختلفة شبيهة جداً بهذه الأصوات ، توجد لعبة مسدس مثلاً كلما تضغط على الزر ينتج نغمة مختلفة وكذلك بعض الدراجات يتم تركيب فيها دائرة الكترونية تنتج بعض نغمات التنبيه ، كذلك في سيارات الاسعاف والشرطة وغيرها من التطبيقات المهمة ، و الآن انت تستطيع فعلها بل والابداع فيها كما تشاء سواءًا بتغيير النغمات أو بإضاءة ليدات معها بأشكال جذابة أو بوضع عدة سويتشات كل سويتش يقوم بوظيفة معينة و الفن هنا في اختيار الترددات المناسبة والوقت المناسب لكل تردد . علماً أن هناك طرق كثيرة أخرى لتوليد الترددات وانتاج النغمات غير استخدام البك وذلك باستخدام دوائر متكاملة أخرى وربما نشير إليها فيما بعد .

يجدر التنبيه هنا أنه يمكنك انتاج نغمات جميلة بطريقة أخرى جميلة غير تغيير الأرقام تغييرًا مباشرًا فيمكنك استخدام متغيرات أو إضافة معادلات تقوم بتوليد النغمة . ولكن ماذا يعنى هذاالكلام؟؟

سأوضح لك الأمر نحن الآن نقوم بجعل البك يقوم بإنتاج تردد معين لمدة معينة ، وهذا التردد لكي نغيره فإننا نقوم بتغيير الرقم داخل أمر الإنتظار ... جميل ،، ما رأيك أن نقوم باستبدال هذا الرقم ونضع بدلاً منه متغير هذا المتغير تتغير قيمته باستمرار بأن تتزايد مثلا أو تتناقص وبالتالي سيتغير الصوت لتغير التردد.

والسؤال هنا هل يمكننا أن نكتب الأمر delay_ms(x); ؟ حيث أن x متغير تم تعريفه من قبل ؟؟؟ هل يمكننا فعل ذلك وجعل ذلك داخل جملة while(1) لكي يتكرر هذا الأمر باستمرار ونضع أيضاً الأمر x++; كما في الكود التالي :

```
void main( )
{ int x;   trisb=0;    portb=0;
while(1){ portb.f0=1;  delay_us(x);
          portb.f0=0;  delay_us(x);
          x++;
         }
}
```

أليست فكرة جميلة ؟؟ في كل مرة ستتغير قيمة المتغير x فمرة تكون بواحد وكأننا كتبنا الأمر delay_us(1) ومرة تكون باثنين وكأننا كتبنا delay_us(2) وهكذا كل مرة تزداد مدة الإنتظار . وبالتالي سيتغير الصوت الناتج .. حقاً إنها فكرة جميلة ، لكنها طريقة فاشلة للأسف ..!!

لماذا ؟ لأنه إذا كتبنا الكود السابق فلن تستطيع لغة مايكروسي ترجمته وتظهر رسالة خطأ بالأسفل . . ولكن لماذا ؟ لأن أمر الانتظار إذا كان بالميكروثانية لا يمكن أن نضع داخله متغير . هكذا صُمّمت لغة البرمجة مايكروسي ، فيمكننا أن نضع بين القوسين ثابت رقمي فقط كما تعودنا ولا يمكنا وضع متغير داخل القوسين.

ما هذا الهراء؟ لماذا تتطرح فكرة فاشلة يا مهندس أحمد ؟ .. صبراً أخي القارئ . إن الفكرة جميلة كما اتفقنا لكن الطريقة التي حاولنا كتابة الكود بها طريقة خاطئة . فالطريقة هي الفاشلة وليست الفكرة .

هل معنى ذلك أنه يمكننا تطبيق هذه الفكرة ولكن بكود مختلف ؟ بالطبع نعم .

ببساطة عندما تريد تطبيق هذه الفكرة في أي مشروع من مشاريعك قم بكتابة الأمر ;(delay_us(1 وقم بتكرار هذا الأمر عدة مرات على حسب قيمة x . بمعنى أننا سنضع هذا الأمر داخل جملة for بشكل معين كما يلي :

```
for(y=0;y<x;y++) delay_us(1);
```

وعلى حسب قيمة x سيكون زمن الانتظار .

إن لم تفهم كلامي السابق سأوضحه لك أكثر . أنت تعلم أن جملة for وظيفتها تكرار مابداخلها

```
for(y=0;y<15;y++) {
        هنا سكتب الأ امر سكرر ١٥ مرة
}
```

وهكذا إذا أردنا تكرار مجموعة من الأوامر ٥٠ مرة مثلاً فإننا سنقوم بتغيير الرقم ١٥ في الكود السابق إلى ٥٠

أما إذا أردنا تكرار أمر واحد وليس مجموعة أوامر فإنه يمكننا (اختياري) أن نستغنى عن الأقواس { } .

جميل ،، ولكن ماذا إذا أردنا أن نكرر أمر معين X من المرات ، أي على حسب قيمة X سيتكرر تنفيذ الأمر . فلو كانت قيمة X تساوي ١٢ سيتم تكرار الأمر ١٢ مرة ولو كانت قيمة X تساوي ٥٠ فإن التكرار سيكون ٥٠ مرة وهكذا .ببساطة سنستبدل الرقم ١٥ في الكود السابق ونضع بدلا منه المتغير X .

وبالتالي سيكون الكود بالشكل التالي :

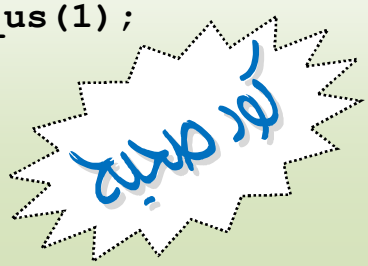
```
for(y=0;y<x;y++) delay_us(1);
```

ولتوضيح الأمر أكثر وأكثر انظر لهذا المثال :

```
void main( )
{ int x;   trisb=0;    portb=0;
while(1){ portb.f0=1;  delay_us(x);
          portb.f0=0;  delay_us(x);    x++;
         }
}
```

هذاكود خاطئ وتصحيحه سيكون بالكود التالي والذي سيؤدي نفس الفكرة (نفس الوظيفة) :

```
void main( )
{ int x,y;   trisb=0;    portb=0;
while(1){ portb.f0=1;  for(y=0;y<x;y++) delay_us(1);
          portb.f0=0;  for(y=0;y<x;y++) delay_us(1);
          x++;
         }
}
```



إذن الكود الثاني هو نفس الكود الأول من ناحية الوظيفة لكن الكود الأول غير صحيح والثاني هو الصحيح .

وتتلخص هذه الفكرة في أنه بدلاً من جعل البك ينتظر ٥٠ ميكرثانية مثلاً يمكنني أن أجعله ينتظر واحد ميكروثانية ولكن يكرر هذا الانتظار ٥٠ مرة(من خلال جملة for) فتكون المحصلة ٥٠ ميكروثانية تقريبا . وفي الكود السابق كل مرة تتغير قيمة Xمن خلال الأمر ;x++ وبالتالي سيتغير التردد باستمرار ويتغير

الصوت الناتج. في اعتقادي الشخصي أنها فكرة جميلة جداً ، و هذه الفكرة مهمة جداً فهي تحل الكثير من المشاكل ولها تطبيقات رائعة سنتعرف عليها من خلال التجارب القادمة بإذن الله تعالى .

في نهاية هذه التجربة المتعة أريد أن أنبهك لشيء مهم وهو أنه عند استخدامك للأمر ( )delay_us فإنه يجب أن يكون المذبذب الذي تستخدمه تردده عالي وكلما تعاملت مع زمن أصغر مثل الميكروثانية delay_us(1) عليك أن تختار مذبذب تردده كبير مثل ١٥ ميجا أو ٢٠ ميجاهيرتز مثلاً ولكن السؤال هو لماذا؟ والسؤال الأهم أولاً هو ما هي المذبذبات ؟

ما هي المذبذبات Oscillators ؟ المذبذبات هي دائرة تنتج موجة من تيار مستمر . هذا هو أحد التعريفات العلمية للمذبذبات . إذن الوظيفة هي إنتاج موجة أو إنتاج نبضات ، وفي تطبيقاتنا هنا يتم توصيل المذبذب أياً كان نوعه بالمايكروكنترولر فيقوم المذبذب بإرسال هذه النبضات للمايكروكنترولر وكل نبضة أو مجموعة نبضات (أو بالأصح كل دورة أو مجموعة دورات cycles) تأتي للمايكروكنترولر يقوم بتنفيذ أمر معين في الكود . لهذا المذبذب ضروري جداً للبك وبدونه لا يعمل . و إذا زادت سرعة هذه النبضات (أي زاد تردد المذبذب) زادت سرعة المايكروكنترولر في تنفيذ الأوامر .

لعلك تتذكر في التجربة رقم واحد أننا قمنا بتوصيل مكثف ٢٢ بيكو ومقاومة ١٠ كيلو بالطرف رقم ١٦ هنا يعتبر المكثف والمقاومة مذبذب من النوع RC Oscillator والتردد الناتج سيكون ١,٠٨ ميجا هيرتز تقريبا لذلك كنا نكتب التردد بهذه القيمة في لغة مايكروسي عندما كنا ننشئ مشاريع جديدة وكذلك في برنامج المحاكاة Proteus .

إن البك يعتبر نوع من أنواع المايكروكنترولر كما تعلم . و pic16f84a نوع من أنواع البك ، وهذا النوع يحتاج إلى أربع دورات لكي يقوم بتنفيذ أمر واحد في الكود وبعض الأوامر تحتاج إلى ثمان دورات . علما أن هذه الأوامر ليست أوامر لغة مايكروسي العادية مثل جملة if وجملة for ونحوه بل هذه الأوامر تخص لغة الأسمبلي (لغة التجميع Assembly languge) .

ولغة الأسملبي هي أول لغة تم تصميمها لكي يستخدمها المبرمجون لبرمجة المايكروكنترولر وهي أقل سهولة مئات المرات من لغة مايكروسي . كما أنها أطول . بمعنى أنه لكتابة كود يؤدي وظيفة معينة وكان هذا الكود مكتوب بلغة مايكروسي وكان ٢٠ سطر مثلاً فلا تستغرب أنه عند تحويل هذا الكود إلى لغة

الأسمبلي فإنه سيكون مئة سطر أو ربما مئات الأسطر . إذن لغة الأسمبلي أطول وأقل سهولة من لغة المايكروسي ولكنك إذا استخدمتها فإنك تعرف بدقة زمن تنفيذ كل أمر ومن الصعب معرفة زمن تنفيذ كل أمر بدقة في لغة مايكروسي ، فكل أمر من السي يعتبر مجموعة أوامر من الاسمبلي .

نرجع إلى حديثنا عن المذبذبات ، فكما قلنا أن سرعة البك في تنفيذ الأوامر والعمليات يعتمد على سرعة أو تردد المذبذب ، وهناك أنواع كثيرة من المذبذبات وسنذكر لمحة بسيطة عن كل نوع :

١- RC Oscillator وهو عبارة عن مقاومة ومكثف (R و C) وهذا النوع يتميز بوفرته فيمكنك الحصول عليه بسهولة كما يتميز برخص ثمنه ، ويستخدم في التطبيقات والمشاريع التي لا نحتاج إلى دقة الوقت فيها . لهذا عند استخدامه لن نتعامل مع الميكروثانية .

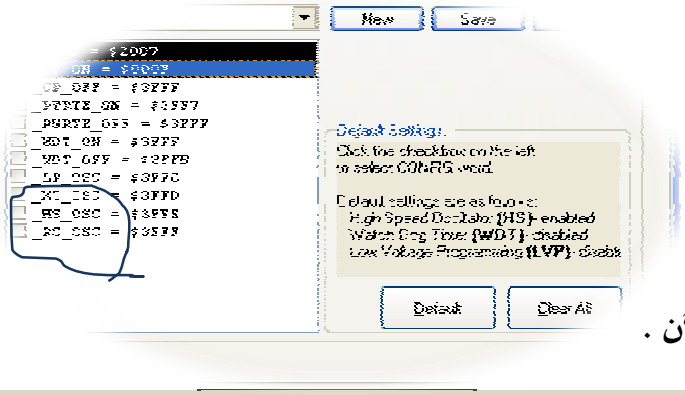
٢- Crystal oscillator (كريستال) هذا النوع هو عبارة عن قطعة الكترونية تنتج تردد معين وهذا النوع يستخدم في المشاريع التي نحتاج فيها إلى دقة عالية في الوقت فهو مناسب لهذا الغرض . وغالبا ما نجد التردد الخاص بالكريستال مكتوب على جسم القعطة .

بعض أشكال الكريستال

هناك أنواع أخرى من المذبذبات وهناك تفاصيل أخرى ولكننا لسنا بحاجة إليها الآن .

والسؤال الأهم الآن هو : ما هي الاعدادت التي ستتغير إذا استخدمنا كريستال ٢٠ ميجاهيرتز مثلا بدلاً من استخدام RC oscillator ؟

في لغة مايكروسي وعند إنشاء مشروع جديد نكتب التردد الخاص بالمذبذب الذي نستخدمه ثم سنختار واحد من الاختيارات التالية :



١- _RC_OSC
٢- _XT_OSC
٣- _HS_OSC

بالطبع هناك اختيارات أخرى لكن لن نحتاجها الآن .

إننا نختار RC_OSC_ عندما نستخدم مذبذب من النوع RC oscillator

نختار XT_OSC_ عندما نستخدم كريستال تردده له إحدى القيم المحصورة بين 200 كيلوهيرتز

إلى 4 ميجا هيرتز .

نختار HS_OSC_ عندما نستخدما كريستال تردده من 4 ميجا هيرتز إلى 25 ميجا هيرتز .

والتردد يمكننا كتابته في المكان المخصص لذلك عند إنشاء مشروع جديد

في الصورة التي تراها تم اختيار التردد 20 ميجاهيرتز

أي 20 000 000 هيرتز .

وبالطبع لابد من اختيار HS_OSC_

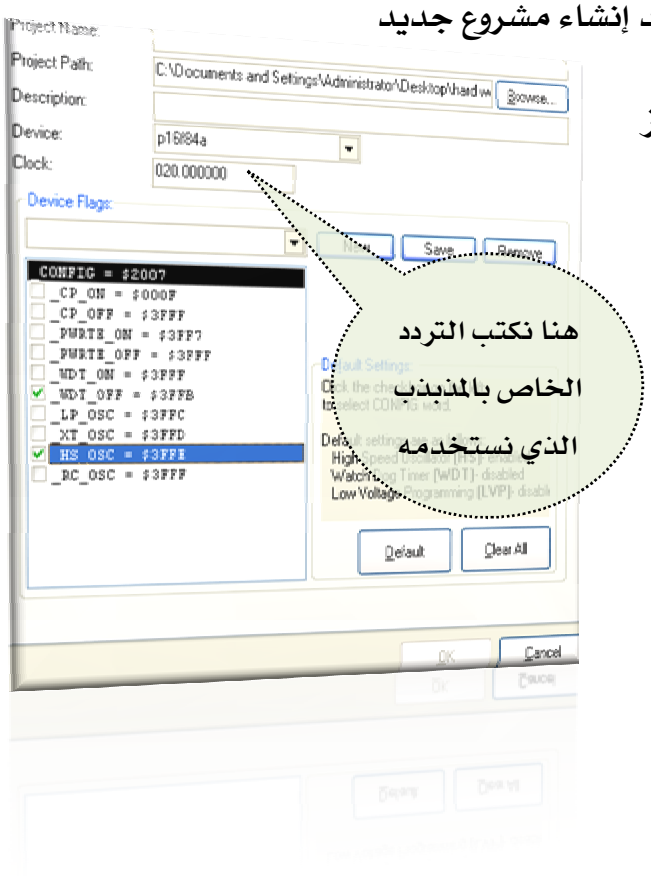
ويفضل عند استخدامك لأي نوع من أنواع البك

أن تقوم أولاً بكتابة التردد ثم تضغط على الزر

Default ثم تختار نوع المذبذب الذي تستخدمه

هل هو RC أم XT أم HS وتترك باقي الاختيارات

كما هي .



نحن الآن عرفنا المكان الذي نكتب فيه التردد ولكن كيف نعرف التردد الخاص بالمذبذب ؟

إذا كنت تستخدم كريستال فغالبا ما ستجد تردده مكتوب عليه . أما إذا كنت تستخدم RC OSC فإن تردده يعتمد على عدة عوامل أهمها قيمة المقاومة (R) وسعة المكثف (C) .

ويمكنك حساب التردد في تلك الحالة من المعادلة التالية :

$$\text{التردد} = \frac{1}{4.2 \times R \times C}$$

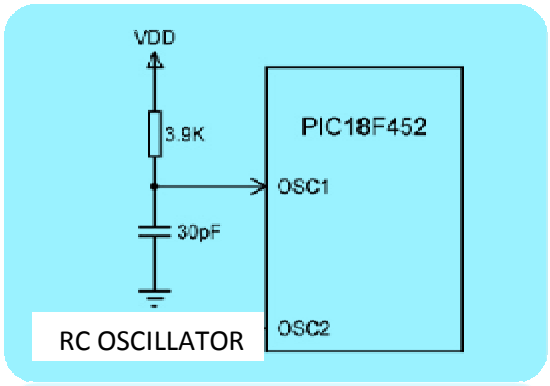
والتردد المحسوب هو تردد تقريبي لأن هناك عوامل أخرى يعتمد عليها التردد مثل درجة الحرارة .

وإليك الآن جدول ببعض القيم المستخدمة لهذا النوع من المذبذبات حيث يوضح الجدول قيمة المقاومة وسعة المكثف والتردد الناتج (التردد التقريبي) :

قيمة المقاومة	سعة المكثف	تردد المذبذب
3.3 K	22 pF	3.3 MHz
4.7 K	22 pF	2.3 MHz
10 K	22 pF	1.08 MHz
3.3 K	30 pF	2.4 MHz
4.7 K	30 pF	1.7 MHz
10 K	30 pF	0.793 MHz

لهذا عندما كنا نستخدم مكثف ٢٢ بيكو فاراد ومقاومة ١٠ كيلو أوم فإننا كنا نكتب التردد 1.08MHz .

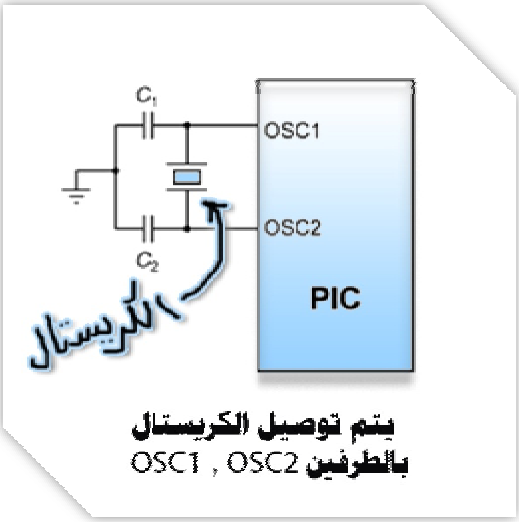
للتذكير : كنا نوصل المذبذب RC بهذه الطريقة أيًا كان نوع البك المستخدم المهم أن نقوم بالتوصيل بالطرف المكتوب بجواره OSC1 كما بالشكل الموضح :



بالطبع يمكنك تغيير قيمة المقاومة والمكثف لتغيير التردد .

كيف نقوم بتوصيل الكريستال ؟

عند توصيل الكريستال فإننا نوصله بالطرفين OSC1 ، OSC2 ومن المهم أن نقوم بتوصيل مكثفين مع الكريستال ، كما يوضح ذلك الشكل التالي :



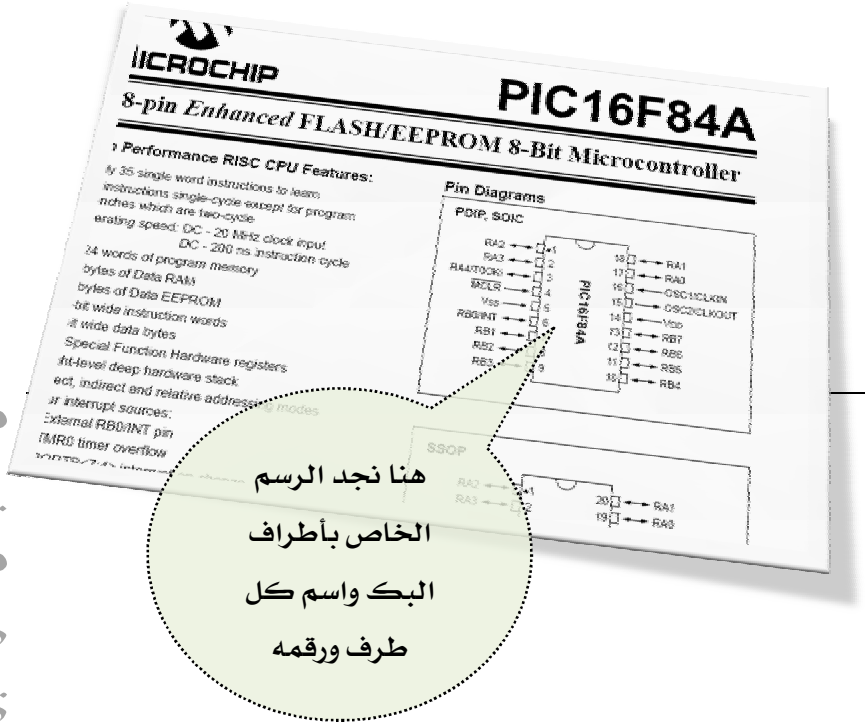
ويتم التوصيل بالأرضي كما هو موضح ولمعرفة الطرفين OSC1,OSC2 قم بفتح الداتاشيت Datasheet واذهب إلى pin diagram ، فستجد صورة توضح لك أرقام كل الأطراف واسم كل طرف ومنه ستجد OSC1,OSC2

فمثلاً في pic16f84a ستلاحظ أن

الطرفان رقم ١٦ و ١٥ لهما الاسم OSC1

و OSC2 ولهذا هذين الطرفين سنوصل

بهما الكريستال عند استخدامه



السؤال المهم الآن هو ما هي قيم تلك

المكثفات المستخدمة مع الكريستال ؟

إن قيم المكثفات تعتمد على شيئين :

١- تردد الكريستال المستخدم . ٢- الـ CRYSTAL MODE والمقصود به هل هو XT أم HS أم ..

والجدول التالي يوضح القيم التي يمكن اختيارها للمكثفات .

MODE	تردد الكريستال	قيمة كل مكثف
XT	100 KHz	100-150 pF
XT	200 KHz	22-68 pF
XT	2 MHz	15-33 pF
XT	4 MHz	15-33 pF
HS	4 MHz	15-33 pF
HS	8 MHz	15-33 pF
HS	10 MHz	15-33 pF
HS	20 MHz	15-33 pF

ستلاحظ من الجدول أن الكريستال ٨ ميجا هيرتز مثلا يمكنك أن تختار له مكثفين قيمة كل واحد منهما قد تكون ١٥ بيكو فاراد (وهي قيمة محصورة بين 15-33Pf) أو تجعل قيمة كل منهم تساوي ٢٢ بيكوفاراد لأن ٢٢ قيمة محصورة بين 15-33pF . إذن أنت لديك العديد من الاختيارات وعمومًا كلما زادت سعة المكثف زاد استقرار المذبذب ولكن في نفس الوقت يزداد الوقت اللازم لكي يبدأ ، وهذا أيضا مشار إليه في الداتاشيت الخاصة بالبك في الصفحة رقم ٢٢ ، ٢٣ (بالنسبة لـ pic16f84 datasheet)

TABLE 6-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

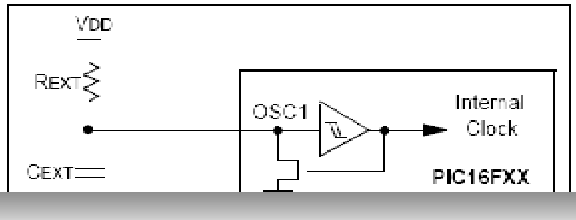
Mode	Freq	OSC1/C1	OSC2/C2
LP	32 kHz 200 kHz	68 - 100 pF 15 - 33 pF	68 - 100 pF 15 - 33 pF
XT	100 kHz 2 MHz 4 MHz	100 - 150 pF 15 - 33 pF 15 - 33 pF	100 - 150 pF 15 - 33 pF 15 - 33 pF
HS	4 MHz 20 MHz	15 - 33 pF 15 - 33 pF	15 - 33 pF 15 - 33 pF

Note: Higher capacitance increases the stability of the oscillator, but also increases the start-up time. These values are for design guidance only. Rs may be required in HS mode, as well as XT mode, to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.
For VDD > 4.5V, C1 = C2 ≈ 30 pF is recommended.

6.2.3 RC OSCILLATOR

For timing insensitive applications, the RC device option offers additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) values, capacitor (CEXT) values, and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types also affects the oscillation frequency, especially for low CEXT values. The user needs to take into account variation, due to tolerance of the external R and C components. Figure 6-3 shows how an R/C combination is connected to the PIC16F84A.

FIGURE 6-3: RC OSCILLATOR MODE



وفي نهاية هذا الدرس لا بأس بأن نشير إلى أن هناك نوع آخر من المذبذبات يسمى **ceramic resonator** هذا النوع يمكن اعتبار أنه عبارة عن كريستال ومكثفين لكن في قطعة واحدة (أو تغليف واحد) وغالباً يكون له ثلاثة أطراف طرف يوصل بالأرضي وطرفان يوصلان بـ OSC1,OSC2. ودقته جيدة وكافية لمعظم المشاريع والتطبيقات ، و لكن أقل من دقة الكريستال لذلك لا يستخدم في التطبيقات التي تحتاج إلى دقة عالية جداً في الوقت .

ويمتاز الـ resonator برخص ثمنه عن (الكريستال+مكثفين) . وعند استخدامه في مشاريعك يمكنك اختيار الوضع(mode) XT أو HS كما كنا نفعل مع الكريستال . نختار _XT_OSC عندما نستخدم resonator تردده له إحدى القيم المحصورة من ٢٠٠ كيلوهيرتز إلى ٤ ميجا هيرتز، و نختار _HS_OSC عندما نستخدما resonator تردده أكبر من 4 ميجا هيرتز .

بعض أشكال الـ Resonators

يتم توصيل الطرف الأوسط (المركزي) بالأرضي ويتم توصيل

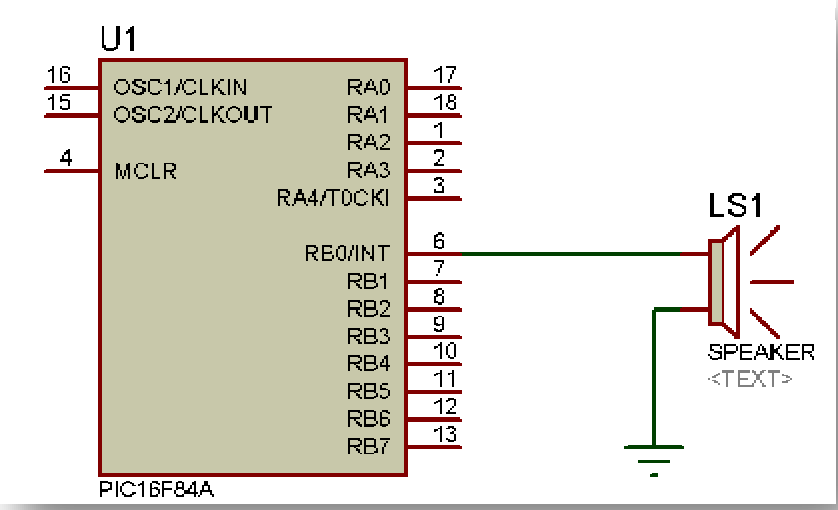
الطرفان الأول والأخير بـ OSC1 و OSC2 .

التجربة (٢٣)

هذه التجربة ما هي إلا تطبيق على التجربة السابقة ولكنها ستكون ممتعة أكثر ، فالهدف من هذه التجربة هو إنتاج بعض النغمات الصوتية والتي يمكنك استخدامها مباشرة في تصميم الألعاب أو في أجهزة الإنذار

أولاً : إليك بعض النغمات التي يمكنك استخدامها في تطبيقات أجهزة الإنذار ، فهذا الصوت الذي سنولده يشبه صوت سيارات الشرطة أو الاسعاف .

سنقوم بتوصيل السماعة (في الحقيقة أو في برنامج المحاكاة)



ثم نقوم بإنشاء مشروع جديد نستخدم فيه مذبذب من النوع crystal (كريستال) ذو تردد 4MHz ونختار XT_OSC_ أو HS_OSC_ . ثم نقوم بكتابة الكود التالي :

```
void main(){ int x; trisb=0; portb=0;
while(1){
     for(x=0;x<400;x++){ portb.f0=1;   delay_us(750);
                         portb.f0=0;   delay_us(800);}
     for(x=0;x<450;x++){ portb.f0=1;   delay_us(500);
                          portb.f0=0;   delay_us(750);}
          }    }
```

ما رأيك في هذا الصوت ؟؟ .. إنه من الجميل أن تنتج مثل هذا الصوت مع العلم أنه بالكثير من المحاولات يمكنك انتاج ما هو أفضل منه . وتتلخص فكرة الكود السابق في أننا قمنا بإنتاج ترددين تردد يستمر لمدة معينة ثم تردد آخر يعقبه ويستمر لمدة معينة ويتكرران باستمرار .

وإليك الآن نغمة أخرى تشبه السابقة لكن بتردد أعلى قليلاً .

```
void main(){ int x; trisb=0; portb=0;
while(1){
    for(x=0;x<400;x++){ portb.f0=1;  delay_us(450);
                        portb.f0=0;  delay_us(480);}
    for(x=0;x<450;x++){ portb.f0=1;  delay_us(350);
                         portb.f0=0;  delay_us(450);}
        }   }
```

هل لاحظت الفرق بينهما ..!! قمنا فقط بتغيير زمن الانتظار مما أدى لتغير التردد وبالتالي تغير الصوت .

ثانيا : بعض الأصوات التي يمكنك استخدامها في ألعاب الأطفال

هل تذكر الفكرة التي أشرنا إليها في التجربة السابقة لمحاولة تغيير التردد من خلال جعل زمن الانتظار يعتمد على متغير ، وهذا المتغير تتغير قيمته باستمرار

```
for(y=0;y<x;y++) delay_us(1);
```

سنقوم باستخدام هذه الفكرة لإنتاج بعض النغمات الجميلة التي يمكن استخدامها في الألعاب أو في التطبيقات التي تريدها . تأمل في الكود التالي والذي سينتج نغمة صوتية معينة :

```
void main ( )
{ int x,y;  trisb=0;   portb=0;
while(1){
        for(x=1;x<300;x++){
            portb.f0=1;  for(y=0;y<x;y++)delay_us(10);
            portb.f0=0;  for(y=0;y<x;y++)delay_us(10);
                              }
        }
}
```

الآن لدينا جملة for يتكرر ما فيها ٣٠٠ مرة وهي نفسها ستتكرر باستمرار لأنها داخل جملة while(1) .

وهذا تعليق بسيط على الكود : بالنسبة لجملة for الأولى ففي المرة الأولى التي ينفذ فيها ما بداخلها ستكون قيمة x=1 . أول أمر سيجده البك هو portb.f0=1; النبضة واحد هذه ستستمر لمدة معينة هذه المدة يتم تحديدها بواسطة for(y=0;y<x;y++)delay_us(10); وهنا بما أن x تساوي واحد إذن سيكرر البك ما بداخل جملة for الثانية مرة واحدة وبالتالي سينتظر لمدة ١٠ ميكروثانية وبنفس الطريقة ينتقل البك للأمر التالي portb.f0=0; حيث تستمر النبضة صفر لمدة ١٠ ميكروثانية أيضا.

ثم يقوم البك بزيادة قيمة المتغير x ويجعلها تساوي ٢ ثم ينفذ ما بداخل جملة for الأولى مرة أخرى فينفذ الأمر portb.f0=1; وتستمر النبضة واحد هذه لمدة معينة يتم تحديدها بواسطة

```
for(y=0;y<x;y++)delay_us(10);
```

وهنا قيمة x تساوي ٢ وبالتالي سيكرر البك ما بداخل جملة for الثانية مرتين وبالتالي سينتظر البك لمدة مقدارها ٢٠ ميكروثانية ، وبنفس الطريقة ينتقل البك للأمر التالي portb.f0=0; وينتظر أيضاً لمدة ٢٠ ميكروثانية ثم تزداد قيمة المتغير x وتصبح ٣ وبالتالي عند تنفيذ ما بداخل جملة for فإن زمن النبضة واحد سيصبح ٣٠ ميكروثانية وكذلك النبضة صفر ..... وهكذا .

بهذه الفكرة نحن نقوم بتغيير التردد باستمرار منتجين صوتا معينا .

وإليك الآن صوت آخر يمكنك استخدامه أو توليد ما يشبهه .

```
void main ()
{ int x,y;  trisb=0;   portb=0;
while(1){
        for(x=250;x>0;x--){
            portb.f0=1;  for(y=0;y<x;y++)delay_us(5);
            portb.f0=0;  for(y=0;y<x;y++)delay_us(5);
            portb.f0=1;  for(y=0;y<x;y++)delay_us(10);
                              }
        }
}
```

إنه يشبه كثيراً الكود السابق لكن مع جعل قيمة X تتناقص (ولا تتزايد كما في الكود السابق)

كما أننا أضفنا نبضة أخرى تتكرر لمدة معينة فأصبحت الموجة التي لدينا عبارة عن نبضة واحد ثم نبضة صفر ثم نبضة واحد بمدة زمنية مختلفة ويتكرر ذلك باستمرار .

هل تتذكر ذلك الصوت الذي نسمعه من بعض مسدسات الأطفال ؟ إليك هذه النغمة الشبيهة بما أقصد:

```
void main ()
{ int x,y;  trisb=0;   portb=0;
while(1){
        for(x=0;x<50;x++){
            portb.f0=1;  for(y=0;y<x;y++)delay_us(10);
            portb.f0=0;  for(y=0;y<x;y++)delay_us(10);
                              }
        }
}
```

حقاً إنه شيء جميل وممتع ،، الأكواد السابقة لا تحتاج إلى شرح فما هي إلا توليد لترددات مختلفة تنتج أصواتاً مختلفة ، والآن جاء دورك أنت قم بعمل التجارب بتغيير الأرقام التي في جملة for و بإضافة عدد معين من جمل for مثلاً ومن الممكن أن تقوم بكتابة أكواد عشوائياً أو بنظام معين ، المهم ، ابذل جهدك وحاول انتاج صوت جميل تستخدمه في مشاريعك ، مع تمنياتي لك بكل التوفيق والتميز .

التجربة (٢٤)

في هذه التجربة بإذن الله تعالى سوف نتعلم بعض المعلومات المهمة والتي تخص لغة البرمجة ، وسوف نتعلم كيف نجعل الكود الذي نكتبه أكثر تنظيمًا وذلك عن طريق استخدام ما يعرف بالدوال functions .

تأمل جيدا في الكود التالي (مع العلم أننا قمنا بتوصيل ثمان ليدات بالأطراف من B0 إلى B7)

```
void main()
{trisb=0;
portb=0;
while(1)
{portb=0b10000001;delay_ms(100);
 portb=~portb;    delay_ms(100);
 Portb=0b11000011;delay_ms(100);
 portb=~portb;    delay_ms(100);
 Portb=0b11100111;delay_ms(100);
 portb=~portb;    delay_ms(100);
 Portb=0b11111111;delay_ms(100);
 portb=~portb;    delay_ms(100);
    }
}
```

إذا تأملت في الكود السابق فستلاحظ أن الليدات تظهر بشكل معين وليكن 10000001 ثم بعد ذلك ينعكس هذا الشكل فيصبح 011111110 وذلك عن طريق الأمر portb=~portb; وبالطبع يوجد زمن انتظار لكي تتسنى لنا رؤية ما يحدث ، وبنفس الطريقة ستجد باقي الكود فمثلا في المرة الثانية ستجد الليدات تظهر بشكل معين وليكن 1100000011 ثم ينعكس هذا الشكل عن طريق الأمر portb=~portb; ويصبح 0011111100 . . . وهكذا إلى نهاية الكود .

وما أود أن أخبرك به الآن ، هو أن هناك طريقة يمكننا أن ننظم بها هذا الكود أكثر وأكثر وذلك عن طريق ما يسمى بالدوال . وليسهل الفهم سنعتبر أن الدالة هي بمثابة أمر جديد نكتبه فيؤدي وظيفة معينة نحن من يحددها . فمثلا نحن الآن نريد أن نقوم بإنشاء أمر جديد(بالأصح دالة جديدة) هذا الأمر نكتب فيه شكل الليدات (أصفار ووحايد) فيقوم بإظهارها على الأطراف من B0 إلى B7 ثم ينتظر لمدة ١٠٠ ملي ثانية ثم يقوم بعكس هذا الشكل ثم ينتظر لمدة ١٠٠ ملي ثانية . فما رأيك أن نقوم بتسمية هذا الأمر بـ invert مثلاً أو اي اسم آخر نختاره وليكن ahmad أو show أو .... إلخ كما نريد .

انظر للأمر التالي :

```
Invert(0b100000001);
```

هذا الأمر غير موجود في لغة مايكروسي ولكن نحن نريد أن ننشأه ، ونريد عندما نكتب هذا الأمر أن يقوم الميكروكنترولر بنفس الوظيفة التي يؤديها الكود التالي :

```
Portb=0b100000001; delay_ms(100);
Portb=~portb; delay_ms(100);
```

وبنفس الطريقة لو كتبنا الأمر :

```
Invert(0b11100111);
```

نريد أن يؤدي الميكروكنترولر نفس الوظيفة التي يؤديها الكود التالي :

```
Portb=0b111000111; delay_ms(100);
Portb=~portb; delay_ms(100);
```

وهكذا على حسب القيمة التي توضع بين القوسين الخاصين بالأمر الجديد ;( )invert سيتم جعل Portb يساوي هذه القيمة ثم بعد ١٠٠ ملي ثانية يتم عكس هذه القيمة ... أعتقد الآن أنك فهمت وظيفة الأمر الجديد الذي نريد إنشاءه وكما قلت لك فإن الأصح في التسمية أن نقول الدالة وليس الأمر ولكن لا مشكلة المهم أن نفهم الآن أياً كان المسمى .

إذن بالنسبة للدالة التي نريد انشاءها اسمها نختاره كما نشاء وليكن Invert ولكي تجعل هذه الدالة تقوم بوظيفة محددة لابد أن تقوم بعملية تسمى تعريف الدالة. هذا التعريف يمكن أن نكتبه في بداية البرنامج قبل void main وسنرى الآن الطريقة التي نكتب بها تعريف الدالة في بداية البرنامج

وإليك الكود الذي سنقوم من خلاله بتعريف الدالة :

```
void invert(char aaa)
{
portb=aaa; delay_ms(100);
portb=~portb; delay_ms(100);
}

void main()
{trisb=0;portb=0;
While(1){invert(0b11000011);
          }
}
```

هذا هو **تعريف الدالة** الذي نخبر فيه الميكروكنترولر بالوظيفة التي سيؤديها الأمر الجديد invert

هذا الأمر عندما نكتبه فإن الميكروكنترولر يبدأ بتنفيذ الأوامر التي بداخل تعريف الدالة و يجعل المتغير aaa يساوي القيمة التي كتبناها بين القوسين الخاصين بالأمر

Invert ( );

الكود السابق مكون من دالتين وكما أخبرتك في أول تجربة لنا في هذا الكتاب أن void main هي الدالة الرئيسية للبرنامج ومعنى ذلك أن الميكروكنترولر يذهب إليها أولاً لينفذ ما فيها من أوامر وإذا كان داخلها استدعاء لدالة أخرى فإنه يذهب إليها . إذن أول شيء سينفذه الميكروكنترولر هو الدالة main .

بالطبع الأمرين `trisb=0;portb=0;` لا يحتاجان إلى شرح وكذلك while(1) ، أما الأمر `invert(0b11000011);` هو استدعاء للدالة التي اسمها invert فعندما يريد الميكروكنترولر أن ينفذ هذا الأمر يذهب إلى تعريف الدالة لينفذ ما فيه من أوامر .

ستلاحظ أن أول سطر في تعريف الدالة هو void invert(char aaa)

كلمة void سنشرحها فيما بعد ، أما invert فهو اسم الدالة ، وما بين القوسين هو تعريف لمتغير اسمه aaa من النوع char . وهذا المتغير له أهمية كبيرة ستعرفها الآن .. فقط تابع بتركيز

فعندما نكتب الأمر invert(0b11000011); فإننا سنقوم بجعل قيمة المتغير aaa تساوي القيمة المكتوبة بين القوسين الخاصين بالأمر invert( ); إذن المتغير aaa الآن يساوي 0b11000011 .

ثم يقوم الميكروكنترولر بتنفيذ الأوامر التي بين القوسين الخاصين بالدالة Invert فأول أمر سيتم تنفيذه هو Portb=aaa; أي أن portb ستصبح قيمته تساوي 0b11000011 ثم يتم تنفيذ الأمر الذي يليه وهو الانتظار ثم الأمر الذي يليه وهو عكس القيم (جعل الصفر واحد والواحد صفر) عن طريق الأمر Portb=~portb; ثم امر الانتظار ثم بعد ذلك يعود الميكروكنترولر للمكان الذي في الدالة الرئيسية عند الأمر invert(0b11000011); وينفذ ما بعده من أوامر وفي حالتنا هذه سيعيد تنفيذ نفس الأمر لأنه داخل جملة while(1) .

والآن سنقوم بتحويل الكود التالي (أول كود كتبناه في هذه التجربة) إلى كود آخر أكثر تنظيماً باستخدام الدوال فتأمل جيدا في هذا الكود ثم في الكود الذي يليه الذي تم به التعديل :-

```
void main() {trisb=0;portb=0;
while(1)
{portb=0b10000001;delay_ms(100);
 portb=~portb;    delay_ms(100);
 Portb=0b11000011;delay_ms(100);
 portb=~portb;    delay_ms(100);
 Portb=0b11100111;delay_ms(100);
 portb=~portb;    delay_ms(100);
 Portb=0b11111111;delay_ms(100);
 portb=~portb;    delay_ms(100);
    }
}
```

```
void invert(char aaa)
{
portb=aaa; delay_ms(100);
portb=~portb; delay_ms(100);
}
void main(){trisb=0;portb=0;
while(1){
        invert(0b10000001);
        invert(0b11000011);
        invert(0b11100111);
        invert(0b11111111);
        }
}
```

لقد قمنا باستدعاء الدالة invert أربع مرات كل مرة نجعل فيها portb له قيمة معينة مختلفة عن الأخرى . أليس هذا أفضل من أن نكتب الكود أربع مرات !! (الكود الذي يجعل portb يساوي قيمة معينة ثم ينتظر ثم يعكس القيمة ثم ينتظر) .

إذا لم تفهم الكلام السابق اقرأه مرة أخرى وحاول فهمه جيدا ولا تيأس من كثرة المحاولات . وإليك الآن مثال آخر سيجعلنا نفهم الدوال بشكل أكثر بإذن الله تعالى :

نريد أن نقوم بعمل برنامج بحيث تضيء الليدات من B0 إلى B7 من اليمين إلى اليسار ويتكرر ذلك مرتين. ثم من اليسار إلى اليمين ، ويتكرر ذلك ثلاث مرات ثم تنطفئ جميع الليدات وتظل منطفئة إلى أن يتم الضغط على السويتش(المفتاح) الموصل بـ A0 وعندها تضيء الليدات من اليمين لليسار . وإذا تم الضغط على السويتش الموصل بـ A1 تضيء الليدات من اليسار لليمين ، وسنكتب الكود الآن بطريقة عادية بدون استخدام الدوال ثم نكتبه مرة أخرى باستخدام الدوال ولنرى الفرق .

```
void main(){char x;
trisb=0;portb=0; trisa.f0=1;trisa.f1=1;
for(x=0;x<2;x++){   portb=0b00000001;delay_ms(200);
                    portb=0b00000011;delay_ms(200);
                    portb=0b00000111;delay_ms(200);
                    portb=0b00001111;delay_ms(200);
                    portb=0b00011111;delay_ms(200);
                    portb=0b00111111;delay_ms(200);
                    portb=0b01111111;delay_ms(200);
                    portb=0b11111111;delay_ms(200);}
for(x=0;x<3;x++){portb=0b10000000;delay_ms(200);
                 portb=0b11000000;delay_ms(200);
                 portb=0b11100000;delay_ms(200);
                 portb=0b11110000;delay_ms(200);
                 portb=0b11111000;delay_ms(200);
                 portb=0b11111100;delay_ms(200);
                 portb=0b11111110;delay_ms(200);
                 portb=0b11111111;delay_ms(200);}
while(1){ if(porta.f0==0){    portb=0b00000001;delay_ms(200);
                              portb=0b00000011;delay_ms(200);
                              portb=0b00000111;delay_ms(200);
                              portb=0b00001111;delay_ms(200);
                              portb=0b00011111;delay_ms(200);
                              portb=0b00111111;delay_ms(200);
                              portb=0b01111111;delay_ms(200);
                              portb=0b11111111;delay_ms(200);}
          if(porta.f1==0) {portb=0b10000000;delay_ms(200);
                           portb=0b11000000;delay_ms(200);
                           portb=0b11100000;delay_ms(200);
                           portb=0b11110000;delay_ms(200);
                           portb=0b11111000;delay_ms(200);
                           portb=0b11111100;delay_ms(200);
                           portb=0b11111110;delay_ms(200);
                           portb=0b11111111;delay_ms(200);}
      }  }
```

حركة الليدات من اليمين لليسار وتتكرر مرتين لأنها داخل جملة for

حركة الليدات من اليسار لليمين وتتكرر ثلاث مرات

بعد أن تضيء الليدات من اليمين لليسار ثم من من اليسار لليمين ننتظر إلى أن يتم الضغط على السويتش A0 أو A1 إذن عملية اختبار السويتشات كما نعلم لابد أن تحدث باستمرار إذن سنضع جملتي If التي تختبر السويتشات داخل جملة while(1) لكي تتكرر عملية الاختبار باستمرار.

وإذا تم الضغط على السويتش A0 أي تحقق شرط جملة if التالية `if(porta.f0==0)` فإن الليدات ستضيء من اليمين لليسار وإذا تم الضغط على السويتش A1 أي تحقق شرط جملة If التالية `if(porta.f1==0)` فإن الليدات ستضيء من اليسار لليمين .

والآن هيا بنا لنقوم بإعادة كتابة نفس البرنامج ولكن باستخدام الدوال functions ولنقوم بذلك سننظر لأهم الوظائف التي في الكود كل وظيفة سنجعلها دالة . وبالنظر للكود سنرى أن أهم الوظائف هي حركة الليدات من اليمين لليسار وحركة الليدات من اليسار لليمين .

سنسمي الدالة التي تقوم بحركة الليدات من اليمين لليسار بأي اسم نختاره وليكن right_left

سنسمي الدالة التي تقوم بحركة الليدات من اليسار لليمين بأي اسم نختاره وليكن left_right

كما في الكود التالي : -

```
Void right_left(){  portb=0b00000001;delay_ms(200);
                    portb=0b00000011;delay_ms(200);
                    portb=0b00000111;delay_ms(200);
                    portb=0b00001111;delay_ms(200);
                    portb=0b00011111;delay_ms(200);
                    portb=0b00111111;delay_ms(200);
                    portb=0b01111111;delay_ms(200);
                    portb=0b11111111;delay_ms(200);}
Void left_right(){portb=0b10000000;delay_ms(200);
                  portb=0b11000000;delay_ms(200);
                  portb=0b11100000;delay_ms(200);
                  portb=0b11110000;delay_ms(200);
                  portb=0b11111000;delay_ms(200);
                  portb=0b11111100;delay_ms(200);
                  portb=0b11111110;delay_ms(200);
                  portb=0b11111111;delay_ms(200);}

void main(){char x;
trisb=0;portb=0; trisa.f0=1;trisa.f1=1;
for(x=0;x<2;x++)right_left();
for(x=0;x<3;x++)left_right();
while(1){ if(porta.f0==0) right_left();
          if(porta.f1==0) left_right();
        }
          }
```

ستلاحظ هذه المرة أننا اختصرنا مجموعة الأوامر التي تشير للحركة من اليمين لليسار بأمر واحد وهو الأمر الذي يستدعي الدالة وكتبنا right_left(); وهنا لا يوجد متغير بين القوسين ( ) لأنه لن تجرى عمليات على هذا المتغير من خلال الدالة .. هنا وظيفة الدالة فقط تنفيذ مجموعة من الأوامر بأمر واحد فقط . وكذلك الحال عندما كتبنا الأمر left_right(); فهو استدعاء للدالة التي بها الأوامر التي تقوم بالحركة من اليسار لليمين (أيضا الكود يمكن اختصاره أكثر وسأتركك تفكر بالطريقة بمفردك). والآن لنتعرف أكثر على الدوال :-

ما هي الدوال functions ؟

الدوال هي جزء من الكود يكتب لتأدية حدث أو وظيفة معينة ، وتنشأ الدوال عندما نحتاج إلى تكرار وظيفة معينة في أماكن مختلفة من الكود و كذلك لترتيب وتنظيم الكود . والمبرمج الجيد لا يتسغني أبداً عن الدوال وعن استخدامها بشكل منظم .

والدوال لها عدة أشكال سنتعرف عليها بإذن الله تعالى من خلال الأمثلة التالية :

١- الدالة التي لا تأخذ معاملات parameters ولا ترجع قيم .

ويكون الشكل العام لها كما يلي :

```
void any_name()
{
هنا نكتب الأوامر التي ستنفذ عندما نستدعي هذه الدالة
}
```

```
void main()
{.....
 .....
any name();
}
```

عندما نريد استدعاء الدالة لتنفذ مجموعة من الأوامر نكتب اسمها ثم نتبعها بقوسين فارغين ( )

مثال : نريد عمل دالة اسمها attention (أي : انتبه) هذه الدالة تقوم بإضاءة ليد معين خمس مرات بسرعة أهميتها مثلا أن ننبه المستخدم ليعلم أن ما يفعله شيء خاطئ

إذن لاستدعاء هذه الدالة سنتكتب الأمر ;()attention ويمكننا كتابته في أي وقت عندما نحتاج إلى تنبيه المستخدم لشيء ما :

```
void attention()
{ char kk;
  for(kk=0;kk<5;kk++){
                    porta.f0=1; delay_ms(100);
                    porta.f0=0; delay_ms(100);
                      }
}
```

الكود السابق هو الذي نكتبه في بداية البرنامج ، وبذلك يمكننا أن نستدعي هذه الدالة ليتم تنفيذ هذه الأوامر وذلك بكتابتنا للسطر ;()attention في أي مكان في الكود وليكن داخل ()void main أو داخل أي دالة أخرى ، وعندما أقول داخل الدالة فإنني بالطبع أقصد ما بين القوسين { } .

لنقوم الآن بكتابة كود لمشروع معين هذا المشروع يحتوي على ثلاثة سويتشات وثلاث ليدات بالإضافة إلى ليد آخر لعملية التحذير . السويتش الأول الموصل بـ b0 عند الضغط عليه سيتم اضاءة الليد الأول الموصل بـb5 والسويتش الثاني الموصل بـ B1 عند الضغط عليه سيضيء الليد الثاني الموصل b6 والسويتش الثالث الموصل بـ b2 عند الضغط عليه سيضيء الليد الثالث الموصل بـ b7 . طبعا اضاءة الليدات تكون مستمرة طالما كنا ضاغطين على السويتش فقط وعند رفع اليد من على السويتش فإن الليد ينطفئ .

الإمكانية المهمة التي نريد اضافتها هي إضاءة الليد الرابع الخاص بالتحذير خمس مرات إذا ضغط المستخدم على مفتاحين في نفس الوقت .

```
void attention()
{ char kk;
 for(kk=0;kk<5;kk++){porta.f0=1; delay_ms(100);porta.f0=0; delay_ms(100); }
}
void main()
{ trisb.f0=1; trisb.f1=1; trisb.f2=1;   اجعل الثلاث الموصل به السويتشات كدخل
 trisb.f5=0; trisb.f6=0; trisb.f7=0;   اجعل الثلاث أطرف الموصل بها الليدات كخرج
  trisa.f0=0;                  اجعل الطرف الموصل به الليد الخاص بالتحذير كخرج
  while(1){ portb.f5=0;portb.f6=0;portb.f7=0;       اطفئ الليدات في البداية وبعد رفع اليد من على أي سويتش
        if(portb.f0==0) {mm1:  portb.f5=1;     إذا تم الضغط على السويتش الأول قم بإضاءة الليد الأول
                               if(portb.f1==0||portb.f2==0) attention();قم بتحذير المستخدم إذا ضغط على أي سويتش آخر
                               if(portb.f0==0)goto mm1; إذا كنا ضاغطين على السويتش الأول كرر ما سبق مرة أخرى
                        }                                  اي واجعل الليد الأول مضيئا
        if(portb.f1==0) {mm2 portb.f6=1;          إذا تم الضغط على السويتش الثاني قم بإضاءة الليد الثاني وإذا تم الضغط
if(portb.f0==0||portb.f2==0) attention();  على السويتش الأول أو الثالث قم بعملية التحذير(استداعاء الدالة)
                         if(portb.f1==0)goto mm2; إذا كنا مستمرين في الضغط قم بتكرار ماسبق
                        }
        if(portb.f2==0) { portb.f7=1; إذا تم الضغط على السويتش الثالث قم بإضاءة الليد الثالث وإذا ضغطنا أثناء ذلك على السويتش الأول
                        if(portb.f0==0||portb.f1==0) attention();          أو الثاني قم باستدعاء دالة التنبيه
                        if(portb.f2==0)goto mm1;
                        }
        }
}
```

ما يهم أن تعلمه من الكود السابق أن هذا النوع من الدوال وظيفته فقط تنفيذ أوامر معينة ثابتة بكتابتنا لأمر واحد الذي يسمى استدعاء الدالة (كتابة اسم الدالة متبوع بقوسين فارغين ()) .

٢- الدالة التي **لها** معاملات ولا ترجع قيم .

```
void any_name(char nn)
{
هنا نكتب الأوامر التي ستنفذ عندما نستدعي هذه الدالة
}
```

```
void main()
{.....
 .....
any name(هنا نكتب أي رقم أو متغير);
}
```

عندما نريد استدعاء الدالة لتنفذ مجموعة من الأوامر نكتب اسمها ثم نتبعها بقوسين ( ) ونكتب داخل القوسين الرقم الذي سيمرر إلى المتغير nn ليصبح nn يساوي هذا الرقم المكتوب

معنى أن الدالة لها معامل أو معاملات parameters أي أن الوظيفة التي تؤديها هذه الدالة تعتمد على قيمة هذه المعاملات ومثال على ذلك الدالة invert التي كتبناها في هذه التجربة .

علما أن الدالة يمكن أن يكون لها أكثر من معامل أو متغير تعتمد عليه .

مثال : نريد عمل دالة نعطيها قيمتين بحيث تجعل portb = القيمة الأولى وينتظر البك لمدة مقدارها يساوي القيمة الثانية ثم يجعل portb بعدها يساوي صفر .

```
void MNO(char ww,int yy)
{char x;
portb=ww;
for(x=0;x<yy;x++) delay_ms(1);
portb=0;
 }
void main()
{.....
.....
MNO(0xff,1500);
}
```

الرقم 0xff هو الذي سيتم وضعه في المتغير ww وهذا الرقم لن يزيد عن ٢٥٥ لذلك سنجعل المتغير من النوع char

والقيمة ١٥٠٠ هي قيمة المتغير yy وبما أن هذا المتغير قد يأخذ قيمة أكبر من ٢٥٥ لذلك جعلناه من النوع int

لاحظ قيم المعاملات أو المتغيرات يمكننا أن نغيرها كما نشاء فالرقم ١٥٠٠ في المثال السابق هو عدد مرات التكرار لجملة for والتي تكرر أمر الانتظار لملي ثانية عدد معين من المرات . فهنا ١٥٠٠ مرة أي أن الانتظار سيكون ١٥٠٠ ملي ثانية تقريبا ، ولو كتبنا بدلا من ١٥٠٠ الرقم ٢٠٠ سيكون مدة الانتظار ٢٠٠ ملي ثانية وهكذا أليست الدوال أمراً رائعاً .. !!

وهنا يجدر الإشارة أنه فيما يخص أمر الانتظار بالملي ثانية delay_ms ذكرنا عنه فيما سبق هو والأمر delay_us أنه لايمكننا أن نضع بين القوسين متغير ويجب وضع رقم . ولكن المعلومة الجديدة أنه يوجد أمر في لغة مايكروسي يمكننا من عمل انتظار بالملي ثانية على حسب قيمة متغير هذا الأمر اسمه (vdelay_ms

وهنا في هذا الأمر يمكننا أن نقوم بوضع متغير داخل القوسين . إذن يمكننا تعديل المثال السابق ليصبح الكود كما يلي :

```
void MNO(char ww,int yy)
{portb=ww;
vdelay_ms(yy);
portb=0;
 }
void main()
{…….
…….
MNO(0xff,1500);
}
```

سؤال يحتاج إلى جواب : لماذا جعلنا المتغير ww من النوع char بينما yy من النوع int ؟؟

الجواب : جعلنا المتغير yy من النوع int لكي يسمح بالأرقام التي أكبر من ٢٥٥ حيث أن المتغير char يسمح بأرقام محدودة وهي المحصورة بين الصفر و 255 فقط ونحن نريد أن يكون زمن الانتظار كبير .

٣- الدالة التي لها معامل أو معاملات وترجع قيمة :

الآن جاء الوقت لتعرف ماذا تعني كلمة void إن هذه الكلمة تعني أن الدالة لا ترجع شيء .. كلام جميل وذكرنا ذلك سابقاً ولا بصراحة لم أفهم ؟ إذن سأوضح لك من خلال المثال التالي :

إذا أردنا أن نقوم بعمل دالة هذه الدالة وظيفتها هي جمع رقمين فمن الأفضل أن نجعل هذه الدالة ترجع لنا قيمة هذه القيمة هي نتيجة الجمع . سنقوم بتسمية هذه الدالة بالاسم add وعندما نستدعي هذه الدالة فلن تكون الفائدة مقتصرة على تنفيذ مجموعة من الأوامر بل إن الدالة سترجع إلينا قيمة معينة يمكننا الاستفادة منها .

في هذا المثال المطلوب عمل دالة لجمع رقمين وإظهار ناتج الجمع على portb (أي جعل portb يساوي قيمة ناتج الجمع) .

```
char add(char num1,int num2)
{char sum;
 sum=num1+num2;
return sum;
 }
void main()
{.....
.....
Portb=add(5,20);
}
```

انتبه : في تعريف دالة الجمع لم نكتب الأمر void لأن هذه الكلمة تعني أن الدالة لا ترجع قيمة ، وهنا الدالة ترجع قيمة وهي حاصل الجمع إذن سنستبدل كلمة void ونكتب بدلا منها نوع القيمة التي سترجع سواءا كانت char أو int أو float أو .... إلخ

أيضا بما أن الدالة ترجع قيمة فإنه بكتابتنا لاسم أي متغير وعمل علاقة تساوي بينه وبين الدالة فإن هذا المتغير سيساوي القيمة التي ترجعها الدالة . فعندما نكتب الأمر portb=add(5,20) فهذا يعني أن

portb سيساوي القيمة التي ترجعها الدالة أي سيساوي ٥+٢٠ أي خمسة وعشرون . وكذلك الحال لو كتبنا الأمر ;x=add(5,3) فإن المتغير x سيساوي ثمانية .

إن الكود السابق الذي كتبناه يمكننا أن نكتبه بشكل آخر وسيؤدي نفس الوظيفة :

```
char add(char num1,int num2)
{
 return (num1+num2);
 }
void main()
{.....
.....
Portb=add(5,20);
}
```

مثال : نريد عمل برنامج (أو كتابة كود) يقوم بجعل قيمة portb تساوي حاصل ضرب ٩ في ٢ وبعد خمس ثواني يصبح portb يساوي حاصل ضرب ١٠ في ٣ وبعد ثانيتين يساوي حاصل ضرب ٢٠٠ في ٥ .

```
int mult(char num1,int num2)
{
 return (num1*num2);
 }
void main()
{ trisb=0; portb=0;
portb=mult(9,2) ;
delay_ms(5000);
portb=mult(10,3) ;
delay_ms(2000);
portb=mult(200,5) ;
}
```

بما أن ٢٠٠ ضرب خمسة أكبر من ٢٥٥ إذن نحن نحتاج لجعل القيمة التي ترجعها الدالة من نوع آخر غير char إذن نختار النوع int . أتمنى من كل قلبي أن تكون قد استوعبت هذا الدرس المهم جيداً وتمنياتي لك بمزيد من التقدم والاحتراف والتميز ... فهذا ما تستحقه وهذا هو طريقك فلا تتخلى عنه . . !!