سلسلة

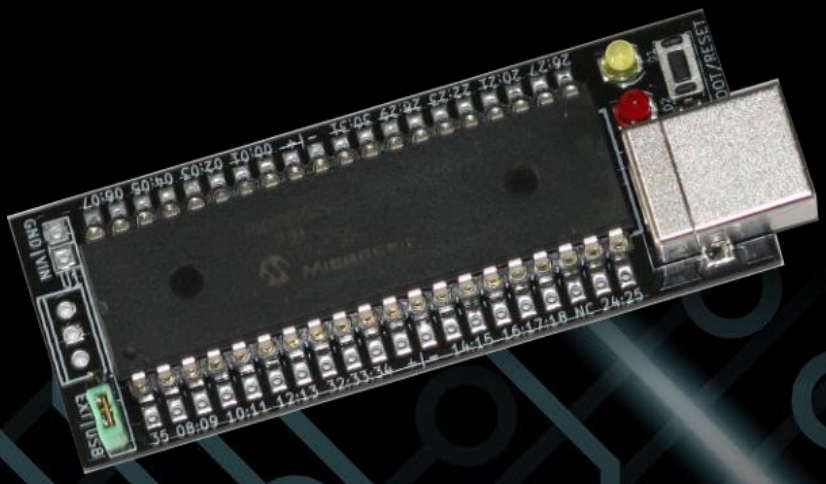
Microcontroller Encyclopedia

ميكروبيديا

موسوعة الميكروكنترولر (الكتاب الأول)

El Gammal Electronics Since 1975

Was started under dynamic leadership of Mr. Anwar El Gammal, January 1975. EL Gammal Electronics, being one of the pioneers and leading components suppliers for almost 4 decades has built an infrastructure to supply market and deliver quality products to our customers all over Egypt and Africa
Founded in 1975 EL Gammal Electronics assumes a top position among the suppliers of electronic components, sensors and tools



البك ميكروكنترولر (١)

Programming Language: C
Compiler: MikroC

Simulation
Proteus ISIS

Hardware circuits
Pictures and Designs

م. حمدي سلطان عبد الخالق

ميكروبيديا

للتواصل مع المؤلف

م. حمدي سلطان عبد الخالق

(002) 011 4645 7403

hamdy.engineer@yahoo.com

Hamdy_soltan2000@hotmail.com

www.facebook.com/enggineer

www.ask.fm/HamdySoltan

www.electrical2011.blogspot.com

www.youtube.com/user/hamdyengineer/

www.facebook.com/EmbeddedSystem2015

http://eg.linkedin.com/pub/hamdy-soultan/38/616/92b

صفحة الكتاب على موقع الفيس بوك

www.facebook.com/MicropediaBook

ميكروبيديا

موسوعة الميكروكنترولر

الكتاب الأول

البك ميكروكنترولر (١)

ميكـرويـديا

الكتاب الأول
البك ميكروكنترولر (١)

•

تأليف
م/حمدي سلطان عبد الخالق

•

إعداد وإخراج
م/ أسامه طلعت عبد الحفيظ

•

رقم الإيداع

٢٢١٤٣ / ٢٠١٢

لطان ١١ / ١٢ / ٢٠١٢

•

ميكروبيديا

Microcontroller Encyclopedia

موسوعة الميكروكنترولر

الكتاب الأول:

البك ميكروكنترولر (١)

PIC Microcontroller (1)

تأليف

م/ حمدي سلطان عبد الخالق

إعداد وإخراج: أسامه طلعت عبد الحفيظ

إهداء ...

أولا: الى عائلتي كلها والتي لا أستطيع ان أرد إليهم ما بذلوه معي من جهد مادي ومعنوي للانتهاء من هذا الكتاب، وأخص بذلك أبي الذي كان سببا لي في عدم التراجع عن كتابة هذا الكتاب وكان بعد الله خير معين لي قبل البدء واثناء وبعد الانتهاء من الكتاب.

ثانيا: الى رفيق الدرب م/ أسامة طلعت عبدالحفيظ والذي لولاه ما خرج الكتاب بهذه الصورة الرائعة والصيغ الجيدة، فهو دائما كان له البصمة واللمسة الاخيرة لكل جزء بل لكل فقرة من فقرات الكتاب، ولا أنسى طبعا دور الوالد أ/ طلعت عبدالحفيظ الذي وقف بجانبنا حتى انتهينا من وصول الكتاب إلى ايديكم.

ثالثا: الى م/ أحمد سمير فايد والذي لطالما استفدت مما يقدم من شروحات ودروس في هذا المجال سائلا الله عز وجل ان يكون ذلك في ميزان حسناته.

رابعا: الى كل من تعلمت على أيديهم وتأثرت بهم وأخص بالذكر منهم الدكتور محمد عبد القوى سليمان (رحمه الله) رئيس قسم هندسة الحاسبات الأسبق بالزقازيق، والمهندس عبد الله على صاحب سلسلة الكتب (ببساطة) وهي (أردوينو ببساطة) و (راسبيري ببساطة) و (الاختراق المادي) و (دليل محاكاة الأردينو) بالإضافة إلى أنه يقوم الان بالتجهيز لكتاب جديد أتمنى له التوفيق وأن ينفعنا بعلمه.

خامسا: الى جميع أصحابي الذين تأخرت عن السؤال عنهم لانشغالي بالكتاب وكانوا دائما يسعون لرفع المعنويات والتشجيع على إتمام هذا الكتاب واخص بالذكر منهم م/ عبد المنعم علي و م/ أشرف سمير و م/اسلام عاطف و م/خالد سعيد و م/ عمر زكريا و م/محمد عادل و م/ عبد الكريم محمد و م/ محمد الشحات و م/ محمود المعداوي و م/ عمار ياسر (رحمة الله عليه) ... وباقي الأصحاب جميعا.

بالإضافة إلى ذلك لا يفوتني أن أتقدم بإهدائيين مهمين جدا بالنسبة لي:

فالأول: إلى أمي، والتي لن يسعي هذا الكتاب بأكمله أن أذكر ما قدمته لي، وأدعو الله أن يجمعني بها في جنته.

والثاني: لحبيبة قلبي ومخطوبتي وشريكة عمري المستقبلة بإذن الله الآنسة / غفران دياب ... والتي أسأل الله أن يتمم لنا و يجمعني بها على خير في الدنيا والآخرة.

حمدي سلطان ،،،

المؤلف

الفهرس

مقدمة المؤلف

بسم اللّه الرحمن الرحيم، بها أبدأ هذا الكتاب فهي خير ما به يبتدى، ثم هذه الأبيات التي يحضر معناها في ذهني كثيرا:

اصبر على مُرِّ الجَفَا من مُعَلِّمٍ ... فإن رُسُوب العِلم في نفَراتِهِ
ومن لم يذق مُرَّ العَلَم ساعة ... تجرع ذل الجهل طول حياتِهِ
ومن فاتهُ التعلِيمُ وقت شبابهِ ... فكَبِّر عليه أربعا لِوفاتِهِ
وذات الفتى واللّه بالعِلم والتقى ... إذا لم يكونا لا اعتبارَ لِذاتِهِ

وبعد ذلك أقول أنه لاشك أن الميكروكنترولر أصبح في السنوات الأخيرة الماضية من أهم العناصر الإلكترونية، وهذا يرجع إلى أسباب عديدة منها: أنه يمكنك برمجته بحيث ينفذ أي شيء تريده، وليس هذا فقط بل يمكنك إعادة برمجته مرات عديدة اذا طرأ على ذهنك تعديل تريد أن تضيفه، بالإضافة طبعا لصغر حجمه وهذا ما يميزه عن الكمبيوتر ما جعله يحل محل الكمبيوتر في كثير من تطبيقات التحكم، والميكروكنترولر يستخدم حاليا في الكثير جدا من التطبيقات والصناعات فهو يستخدم في صناعة السيارات – خاصة السيارات الحديثة التي تحتوى على خاصية التحكم الآلي – ويستخدم في صناعة الأجهزة المنزلية ولعب الأطفال والتحكم في الإنسان الآلي فضلا عن استخداماته في عمليات التحكم المختلفة كالتحكم في درجات الحرارة والتحكم في مستوى الماء والتحكم في أنظمة الأمان التي توجد في المنازل والشركات ... ومن أجل هذا ومع قلة المادة العلمية المتاحة باللغة العربية في هذا المجال قررت أن اسطر في هذه الصفحات ما رزقني به اللّه تعالى من علم سائلا المولى عز وجل أن يجعل ذلك خالصا لوجهه الكريم وأن يجعله لنا في ميزان حسناتنا ...

هذا الكتاب بمثابة مقدمة تبدأ بك من الصفر وتصل بك إلى المستوى الذي تستطيع من خلاله التعامل مع الميكروكنترولر وعمل المشاريع التي تحتاجها فمن خلاله ستستطيع برمجة رجول الميكروكنترولر وستتمكن من توصيله بالشاشات والكيباد وستتعلم كيفية التعامل مع الإشارات التماثلية وكيفية توصيل اثنين ميكرو وكيفية التحكم في المواتير ... وغير ذلك من المواضيع الهامة وكل ذلك مزود بالصور التي تمكنك من عمل محاكاة للمشاريع على الكمبيوتر وأيضا الصور التي تمكنك من عمل الهاردوير ...

وهذا الكتاب مقدم إلى كل شخص مهتم بمجال الميكروكنترولر أو الدوائر الإلكترونية هاويا كان أو متخصصا ولقد حاولت قدر الإمكان أن أضع فيه بطريقة سهلة ومبسطة المعلومات التي قد تبدو صعبة، كما أنه لا يحتاج إلى متطلبات مسبقة سواء معرفة بالميكرو أو بلغة برمجة السي، لكن

ينغى عليك قراءة الكتاب كاملا وبالترتيب على الأقل مرة واحدة لتعرف كل ما فيه إجمالا، لأني أضع الملحوظات الهامة في مكانها الصحيح وهذه الملحوظات هي بمثابة خلاصة الخبرات والتجارب التي مررت بها في هذا المجال ...

وفي النهاية أحب أن أؤكد ترحيبي بأي نقد وأي رأى يمكنا من خلاله توصيل المعلومة صحيحة وبأفضل طريقة، وأيضا أؤكد أن الآراء التي تأتيني أخذها دائما بعين الاعتبار وانفذ الكثير منها وهذا ما حدث في هذا الكتاب ...

بعد نزول الكتاب في الأسواق بفترة بسيطة (وذلك قبل نزوله مجانا على الإنترنت) تواصل البعض مشكورين مع المؤلف رغبة منهم في تقديم الدعم لكتاب ميكروبيديا وذلك حفاظا على مسيرته واستمراره، ومؤخرا حاول البعض الآخر شراء الكتاب كنسخة إلكترونية وعندما أخبرناهم باقتراب نزوله مجانا على الإنترنت أصروا على المساهمة بجزء من ثمنه كدعم للكتب القادمة على الأقل.

وعليه واحتراما لرغبة السادة السابقين ومن يحاول أن يقدم الدعم مثلهم، فقد قام المؤلف بمحاولة بانتقاء أسهل طريقة يمكن التواصل من خلالها وهي (من مصر) خدمة فودافون كاش والتي من خلالها يمكنك تحويل أي مبلغ من رصيدك إلى رقم المؤلف ثم يقوم المؤلف باستلامه كنقود من أي فرع من فروع فودافون.

على من يرغب بالدعم يمكنه اتخاذ الخطوات البسيطة التالية:

- من أي مكان وفي أي وقت اطلب الكود #7000* أو الخدمة الصوتية 7000 من تليفونك لتحويل المبلغ لأي رقم فودافون واتبع الخطوات الاتية:
- اختار اللغة: رقم ١ للعربية أو ٢ للإنجليزية.
- اختار "تحويل الاموال" بإدخال الرقم ١.
- اضغط الرقم ١ لإدخال رقم موبايل المرسل إليه.
- أدخل رقم موبايل المرسل اليه المكون من ١١ رقم وهو: ٠١٠٢٢٦٧٩٩٥٦
- اضغط الرقم ١ لتأكيد الرقم.
- أدخل المبلغ الذي ترغب بدعمنا به.
- أكد على العملية بإدخال الرقم السري الخاص بك (٤ ارقام).
- ستصلك رسالة قصيرة لتأكيد العملية، كما ستصل المؤلف رسالة أخرى لتخبره بوجود مبلغ محول إليه.

أما بالنسبة لمن يريد الدعم من خارج مصر (كمن تواصلوا مشكورين مع المؤلف من الأردن أو من غيرها) فيمكنهم التوصل مع المؤلف على البريد الإلكتروني hamdy.engineer@yahoo.com أو من خلال أي وسيلة أخرى موجودة في بيانات المؤلف بالغلاف الخلفي للكتاب.

ولمن يكره التعامل مع الكتب الإلكترونية ويفضل الكتب المطبوعة فما زالت النسخة المطبوعة متوافرة في الأسواق، تجدونها في محل رام (باب اللوق التحرير القاهرة) ومحل إلكترا (هندسة إسكندرية) ومكتبة نور (هندسة الزقازيق) ومكتبة جرير (هندسة المنصورة).

الفصل الأول

مقدمة عامة عن الميكروكنترولر

يمكنك هذا الفصل من التعرف على الميكروكنترولر ومكوناته واستخداماته وأنواعه ومميزاته وذلك بعيدا عن السرد التاريخي له

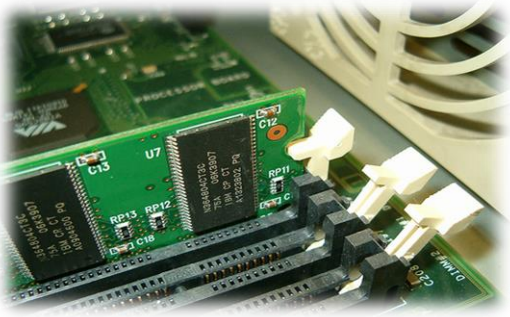
الفرق بين الكمبيوتر والميكروكنترولر

عندما أسألك مما يتكون الكمبيوتر؟ فسوف تجيبني بأسهل ما يكون ذلك لأن أغلبنا قد تعامل مع الكمبيوتر من قبل ويعرفه عن قرب، أما إذا سألتك مما يتكون الميكروكنترولر؟ عندها ستكون الإجابة صعبة إن لم تكن مستحيلة خاصة من لم يتعرض منا للميكروكنتولر من قبل، لكن المفاجأة أن إجابة السؤال الأول هي نفسها إجابة السؤال الثاني!! ... كيف ذلك؟

مكونات الكومبيوتر

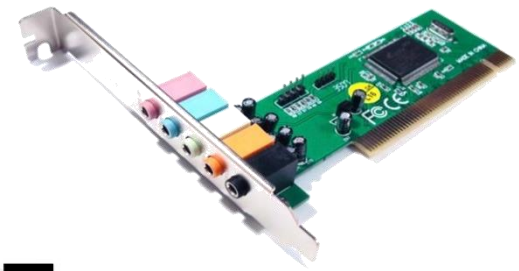
بداية دعنا نتعرف مما يتكون الكمبيوتر كقطع هاردوير من الداخل:

١- المعالج (Processor): وفائدته – كما هو معروف – هي القيام بعمليات المعالجة كالعمليات الحسابية والمنطقية.



٢- الرامات (RAM): وتستخدم لتخزين البرنامج الجاري تنفيذه حاليا وسيتم توضيح ذلك لاحقا بإذن الله.

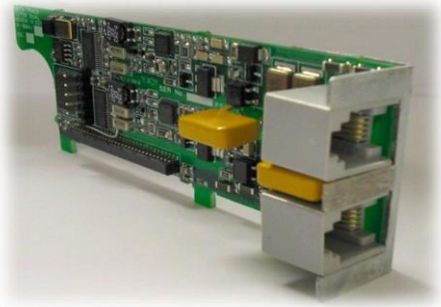
٣- الهارد ديسك (Hard Disc) ويستخدم لتخزين البيانات بأنواعها المختلفة.



٤- مجموعة كروت:

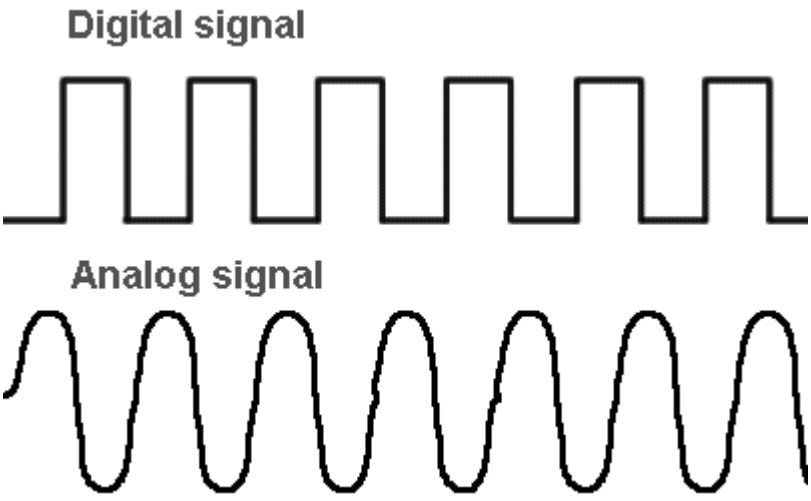
أ- كارت الصوت sound card:

ب- كارت الشاشة:



ج- كارت النت LAN card:

وتستخدم الكروت لربط المعالج بالأجهزة الخارجية التي لا يمكنه التعامل معها مباشرة مثل السماعات – الشاشات – الأنترنت ... الخ.



وترجع عدم قدرة المعالج على التعامل مباشرة مع هذه الأجهزة الخارجية إلى طبيعة البيانات حيث أن شكل البيانات التي يتعامل معها المعالج يختلف عن شكل البيانات التي تتعامل معها تلك الأجهزة، فالمعالج يتعامل مع هذه البيانات في صورة رقمية digital (صفر وواحد) أما الأجهزة الخارجية – كالسماعات مثلا – فهي تتعامل مع إشارات (تسمى إشارات تماثلية أو Analogue signals) وبالتالي يستخدم كارت الصوت للتحويل بين هذين الشكلين من البيانات

٥- وأخيرا الكيسة (Case): والتي تستخدم لاحتواء جميع مكونات الكمبيوتر السابقة.

مكونات الميكروكنترولر

وبعد أن انتهينا من السؤال الأول جاء دور السؤال الثاني وهو: ما هي مكونات الميكروكنترولر وكما ذكر من قبل فإنها نفس مكونات الكمبيوتر لكن مع بعض الاختلاف في المسميات والأحجام والإمكانيات ... لنرى كيف ذلك دعنا نتعرف عليهم:

١- معالج أيضا ولكنه يسمى هنا (microprocessor)والكلمة البادئة (micro) تطلق غالبا في الإنجليزية على الصورة المصغرة من الشيء فمثلا عندما نقارن بين الثانية والميكرو ثانية نجد أن الثانية تتكون من مليون ميكرو ثانية، وهذا إن دل على شيء فإنما يدل على أن إمكانيات المعالج في الميكروكنترولر أقل منها في معالج الكمبيوتر (غالبا إن لم يكن دائما) ويرجع هذا إلى أن معالج الكمبيوتر يصنع ليستخدم في جميع الأغراض فمثلا يستخدم لتشغيل البرامج وكذلك الألعاب وكذلك التعامل مع الإنترنت وغير ذلك من هذه الاستخدامات.

٢- رامات أيضا (RAM) وغالبا ما تؤدى نفس الوظيفة التي تؤديها في حالة الكمبيوتر والتي سيتم شرحها لاحقا.

٣- Flash memory: وهي التي تقوم بنفس وظيفة الهارد ديسك حيث تقوم بتخزين البرنامج الذى يراد للميكرو أن يقوم بتنفيذه.

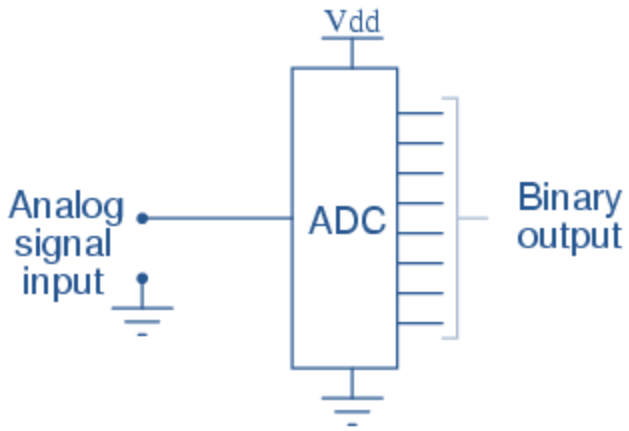
٤- مجموعة كروت أو ما يسمى هنا interfaces وهي تشبه في عملها الكروت الموجودة في الكمبيوتر فمثلا يوجد:

أ- Ethernet interface والذى يستخدم في توصيل الميكرو بالأنترنت.

ب- LCD interface والذى يستخدم لتوصيل الميكرو بشاشات العرض LCD الصغيرة.

ج- Serial and USB interfaces: واللذان يستخدمان لتوصيل الميكرو بالكمبيوتر أو توصيله بميكرو آخر.

د- ADC interface: والذى يستخدم لقراءة الإشارات الأنالوج التناظرية فكما أوضحنا من قبل فإن المعالج يتعامل فقط مع الإشارات الرقمية الد يجيتال ولكى نجعله يتعامل مع الإشارات الأنالوج يستخدم هذا الـ interface والذى يقوم بتحويل الإشارة الأنالوج إلى نظيرتها

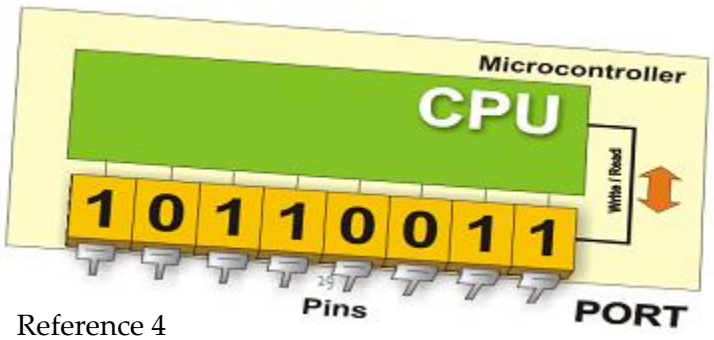


بالشكل الد يجيتال لكى يستطيع يفهمها المعالج وبالتالي يتعامل معها كما هو مبين بالشكل.

هـ التايمر Timer: ويستخدم لتنفيذ مجموعة أوامر بعد فترة زمنية محددة.

و- Power Supply module: من المعروف أن أي IC مثل الميكروكنترولر أو غيره تحتاج جهد مناسب لكى يعمل وفى حالة الميكروكنترولر الجهد اللازم لتشغيله هو خمسة فولت ... ولكن إذا ما حدث أي عطل أو فقد في جهد البطارية لأي سبب من الأسباب وأصبحت تعطى ٤,٥ فولت مثلا بدلا من ٥ هل سيعمل الميكروكنترولر أم لا ؟؟ الجواب هنا أنه سيعمل وهنا تظهر أهمية الـ Power supply module الذي يستخدم لكي يجعل الميكرو يعمل عند على مدي محدود من القيم (range) وليس عند قيمة محددة فمثلا عندما يكون الجهد من ٣ إلى ٥ فولت فان الميكرو يعمل.

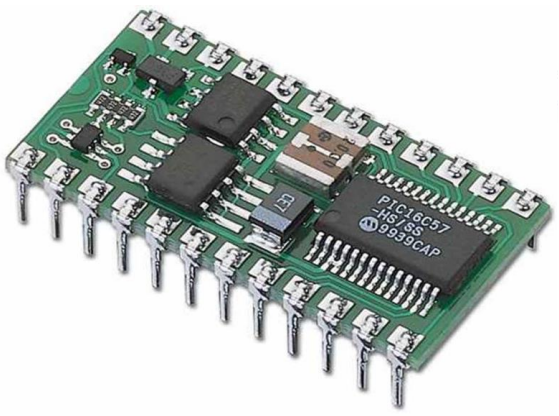
ز- Input and output ports: وهى عبارة عن مجموعة من المخارج (ports) كل منها (غالبا) ما يكون عبارة عن ٨ من الـ buffers التي يستخدمها المعالج لإخراج قيم جهد على رجول الميكروكنترولر أو لاستقبال القيم منها كما هو موضح بالشكل المقابل.



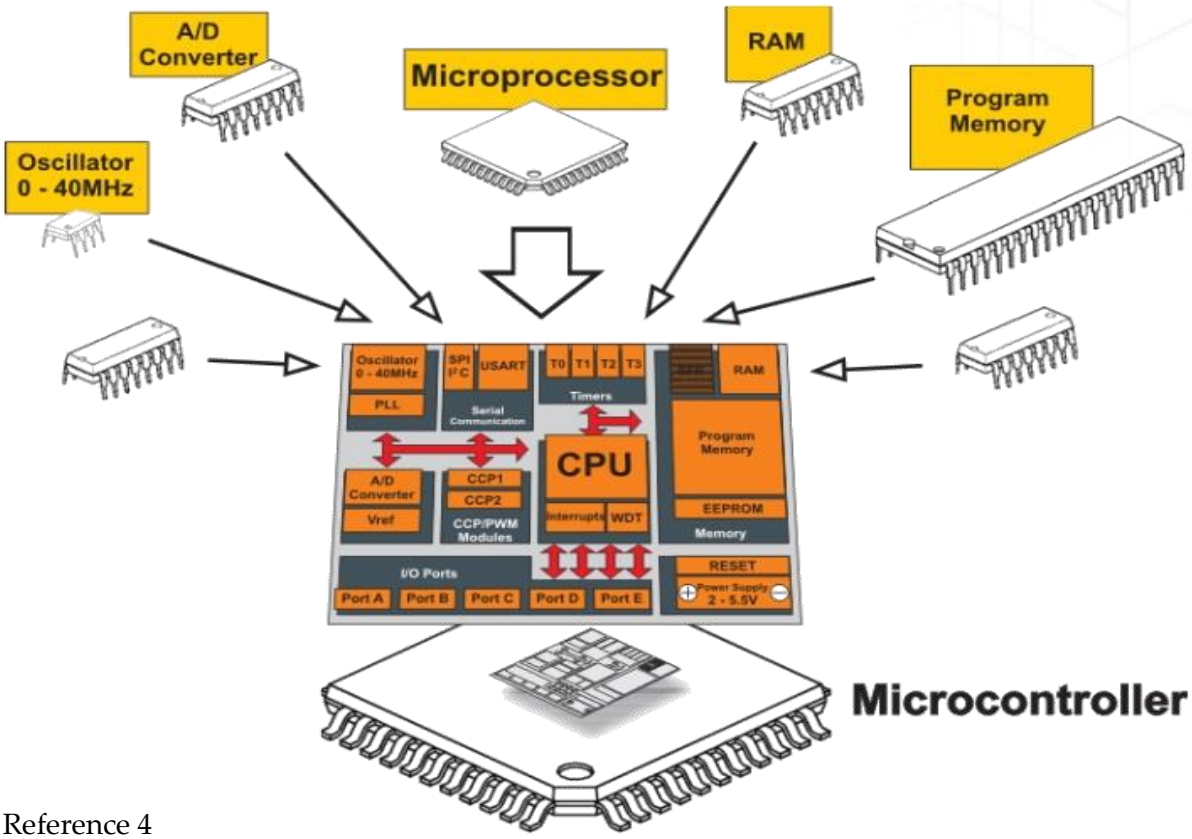
Reference 4

ح- يوجد أيضا interfaces أخرى سنتعرض لها في حينها إن شاء الله.

٥- أيضا Case ولكنها هنا عبارة عن طبقة سميكة تغطى المكونات الداخلية للميكروكنتولر، وتظهر الصورة التالية الميكروكنترولر قبل وبعد إذالة هذه الطبقة الخارجية:



وبالنظر إلى مكونات الميكروكنترولر والكومبيوتر سابقة الذكر نجد أنها تقريبا نفس المكونات ولذلك يمكن اعتبار أن الميكروكنترولر هو عبارة عن كمبيوتر صغير لذلك سمى المقطع الأول منه Micro ونظرا لأن غالب استخدامه في عمليات التحكم سمى الجزء الثاني منه controller.



Reference 4

هذه معناه أنه في النهاية فإن الميكرو عبارة عن تجميع بين ميكروبروسيوسور وذاكرة بأنواعها المختلفة ومجموعة كروت أو interfaces كما بالشكل السابق.

مميزات الميكروكنترولر

وللميكرو على الكمبيوتر مميزات عدة أهمها:

١) صغر الحجم وهي الميزة التي تمكننا من استخدامه في أي مكان.

٢) صغر القدرة المستهلكة less power consumption إذ أنه لكي يعمل يحتاج لجهد ٥ فولت وتيار صغير جدا مقارنة بما يحتاجه الكمبيوتر.

٣) تكلفة الميكرو أصغر بكثير من الكمبيوتر.

لكن هذا ليس معناه أننا يمكننا الاستغناء عن الكمبيوتر إذ أن الميكرو إمكانياته محدودة مقارنة بالكومبيوتر فمثلا الميكرو لا يستطيع تشغيل برنامج الوورد لكن لكل منهم التطبيقات التي يستخدم فيها.

استخدامات الميكروكنترولر

- التحكم في عمل الإنسان الألى: فمثلا التحكم في سرعته ... التحكم في مساره ... التحكم في حركة الأذرع ... قراءة المعلومات (صوت أو فيديو ...) ... إلخ.
- التحكم في درجة الحرارة.
- التحكم في الزمن اللازم لتشغيل الأجهزة.
- التحكم في مستوى الماء في خزان ما.
- التحكم في رطوبة التربة.
- التحكم في الإضاءة.
- الأنظمة السرية أو أنظمة الأمان مثل Home security system.
- يستخدم أيضا في السيارات للتحكم في حركة الفرامل.
- ويوجد في الكثير من الأجهزة المنزلية وغير ذلك من الاستخدامات الكثيرة المتعددة ...

أنواع الذاكرة في الميكروكنترولر

- Flash memory: وتسمى أيضا program memory وتستخدم لتخزين البرنامج ... بمعنى أننا بعد أن نكتب البرنامج على الكمبيوتر ونريد أن (نحرقه) أو ننزله على الميكرو عن طريق البروجرامر فإنه يتم تخزينه على الفلاش.

- RAM: وتسمى أيضا data memory وتستخدم لتخزين المتغيرات التي يتم تعريفها في البرنامج وتستخدم أيضا في تخزين الأوامر التي يتم تنفيذها حاليا.

- EEPROM: وتستخدم أيضا لتخزين البيانات والمتغيرات لكن الفرق بينها وبين الـ RAM أنها تحتفظ بمحتوياتها من بيانات عند فصل الجهد أو الكهرباء عن الميكرو وهذا على عكس الـ RAM التي تفقد محتوياتها بمجرد فصل الجهد عن الميكرو.

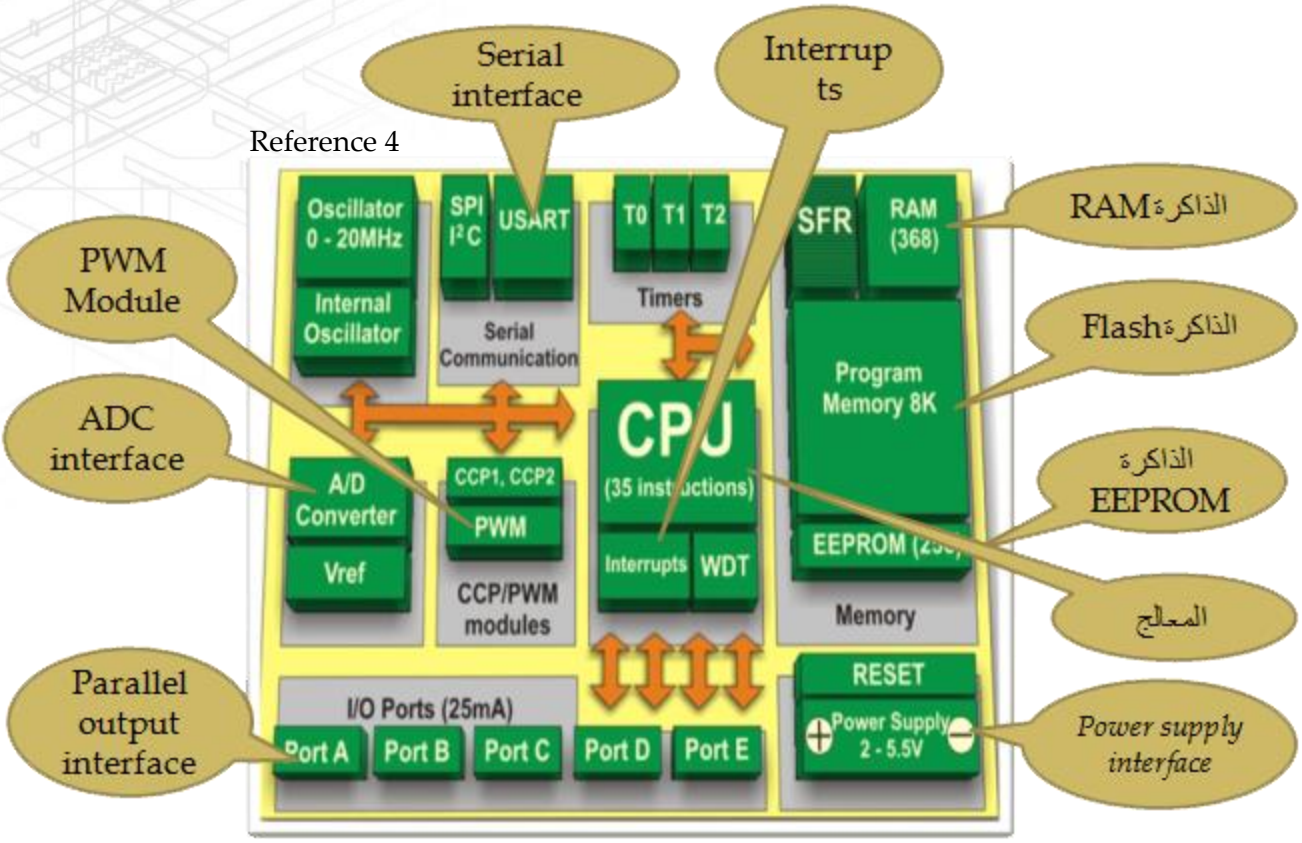
معنى Interface

أوضحنا أن الميكروكنترولر يستطيع إخراج من القيم فقط إما صفر وإما خمسة فولت، ولكن أغلب الأجهزة الأخرى تحتاج إلى قيم مختلفة للجهد فمثلا المروحة الكهربائية تحتاج ٢٢٠ فولت متغير لكي تعمل ... فكيف إذن نستخدم الخمسة فولت الخاصة بالميكروكنترولر في تشغيل والتحكم في المروحة التي تعمل على ٢٢٠ فولت.

يتم هذا عن طريق دائرة هاردوير نقوم نحن بتصميمها بحيث عندما يدخل لها خمسة فولت توصل جهد خارجي قيمته ٢٢٠ فولت متغير إلى المروحة لكي تعمل، مثل هذه الدوائر الهاردوير تسمى interface وبالتالي فإن الإنترفيس بصفة عامة هو عبارة عن دائرة نستخدمها بغرض التوصيل بين الأجهزة الغير متلائمة مع بعضها incompatible ولأن الميكرو غير متلائم مع المروحة فإنه لا يمكن التوصيل بينهم مباشرة وإنما من خلال إنترفيس، وكذلك أيضا فإن المعالج في الكومبيوتر لا يمكننا توصيله بالسماعات مباشرة وإنما يتم ذلك من خلال إنترفيس وهو كارت الصوت المذكور سابقا ...

الشكل التالي يوضح بنوع من التفصيل مكونات الميكروكنترولر من الداخل بما في ذلك الأنواع المختلفة للذاكرة وبعض الـ interfaces الشائعة ...

Reference 4



أنواع الميكروكنترولر

تختلف أنواع الميكروكنترولر نتيجة اختلاف الشركات المصنعة له حيث يوجد العديد من الشركات التي تقوم بتصنيعه فعلى سبيل المثال:

اسم الشركة	لوجو (شعار الشركة)	اسم الميكرو	عائلات الميكرو
Microchip	MICROCHIP	PIC Microcontroller	PIC10, PIC12, PIC16, PIC18, PIC24
Intel	intel	MCS-51	8051
ATMEL	ATMEL	AVR	ATmega , AT90 ,AVR32
Toshiba	TOSHIBA		TX19A
Zilog	zilog		Z180 , Z80

ولكل شركة طريقتها الخاصة في تصميم الميكروكنترولر الخاص بها (organization) لكن هذا لا يهمنا كثيرا – كغير متخصصين – لأننا في النهاية سننظر لكل ميكروكنترولر على أنه عبارة عن معالج وذاكرة بأنواعها المختلفة ومجموعة interfaces.

النوع الذي سنركز عليه في هذا الكتاب هو الـ PIC Microcontroller وهو من صناعة شركة Microchip، وهذا النوع عبارة عن مجموعة من العائلات (Families) وتختلف كل عائلة عن الأخرى – غالبا – فيما تحتويه من interfaces مثل ADC، Ethernet، USB، UART، PWM، ... إلخ.

PIC Microcontroller families:

هذه الصورة توضح الاختلاف في عائلات الـ PIC Microcontroller.

Family	ROM [Kbytes]	RAM [bytes]	Pins	Clock Freq. [MHz]	A/D Inputs	Resolution of A/D Converter	Compar-ators	8/16 – bit Timers	Serial Comm.	PWM Outputs	Others
Base-Line 8 - bit architecture, 12-bit Instruction Word Length											
PIC10FXXX	0.375 - 0.75	16 - 24	6 - 8	4 - 8	0 - 2	8	0 - 1	1 x 8	-	-	-
PIC12FXXX	0.75 - 1.5	25 - 38	8	4 - 8	0 - 3	8	0 - 1	1 x 8	-	-	EEPROM
PIC16FXXX	0.75 - 3	25 - 134	14 - 44	20	0 - 3	8	0 - 2	1 x 8	-	-	EEPROM
PIC16HVXXX	1.5	25	18 - 20	20	-	-	-	1 x 8	-	-	Vdd = 15V
Mid-Range 8 - bit architecture, 14-bit Instruction World Length											
PIC12FXXX	1.75 - 3.5	64 - 128	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	EEPROM
PIC12HVXXX	1.75	64	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	-
PIC16FXXX	1.75 - 14	64 - 368	14 - 64	20	0 - 13	8 or 10	0 - 2	1 - 2 x 8 1 x 16	USART I2C SPI	0 - 3	-
PIC16HVXXX	1.75 - 3.5	64 - 128	14 - 20	20	0 - 12	10	2	2 x 8 1 x 16	USART I2C SPI	-	-

وكل عائلة عبارة عن أكثر من ميكروكنترولر يختلف كل واحد عن الآخر اختلافات بسيطة فلو فرضنا مثلا العائلة 16F والتي تحتوي على الـ ADC interface فسنجد أن الميكرو 16F877A يحتوي على ٨ رجول لقراءة الإشارات الأنالوج بينما الميكرو 16F876A والذي هو من نفس العائلة

يحتوي على ٥ رجول فقط لقراءة نفس الإشارات، وعلى غرار هذا المثال تتضح بقية الاختلافات البسيطة، ويمكنك التعرف أكثر على الاختلافات عن طريق قراءة الداتا شيت والتي سنوضح لاحقا مصادر الحصول عليها وكيفية قراءتها.

وهذا الجدول يوضح الاختلاف بين مجموعة من الميكروكنترولر تنتمي لنفس العائلة:

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages Reference 5	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

اختيار الميكروكنترولر المناسب

يتم اختيار الميكروكنترولر على أساس مجموعة من العوامل:

- فمثلا يجب تحديد عدد رجول الميكروكنترولر التي يحتاجها المشروع وما هي نوعيتها ما إذا كانت د يجيتال أو أنالوج وبناء على التفاوت في إمكانيات عائلات الميكرو وأفرادها يتم اختيار الميكرو الذي يتناسب مع تلك الاحتياجات، فعلى سبيل المثال إذا كان المشروع يحتاج إلى ٢٠ سويتش فلا يمكن استخدام ميكرو يحتوي ٨ رجول فقط وهكذا.

- أيضا يجب تحديد الـ interfaces التي يحتاجها المشروع وتختار نوع الميكرو الذي يحتوي هذه الـ interfaces جميعا، فمثلا إذا كنت تريد توصيل الميكروكنترولر بالإنترنت لنقل معلومات ما، فعندئذ ستلجأ إلى العائلة 18F وليس العائلة 16F لأن العائلة 18F هي التي تحتوي على Ethernet interface وهكذا.

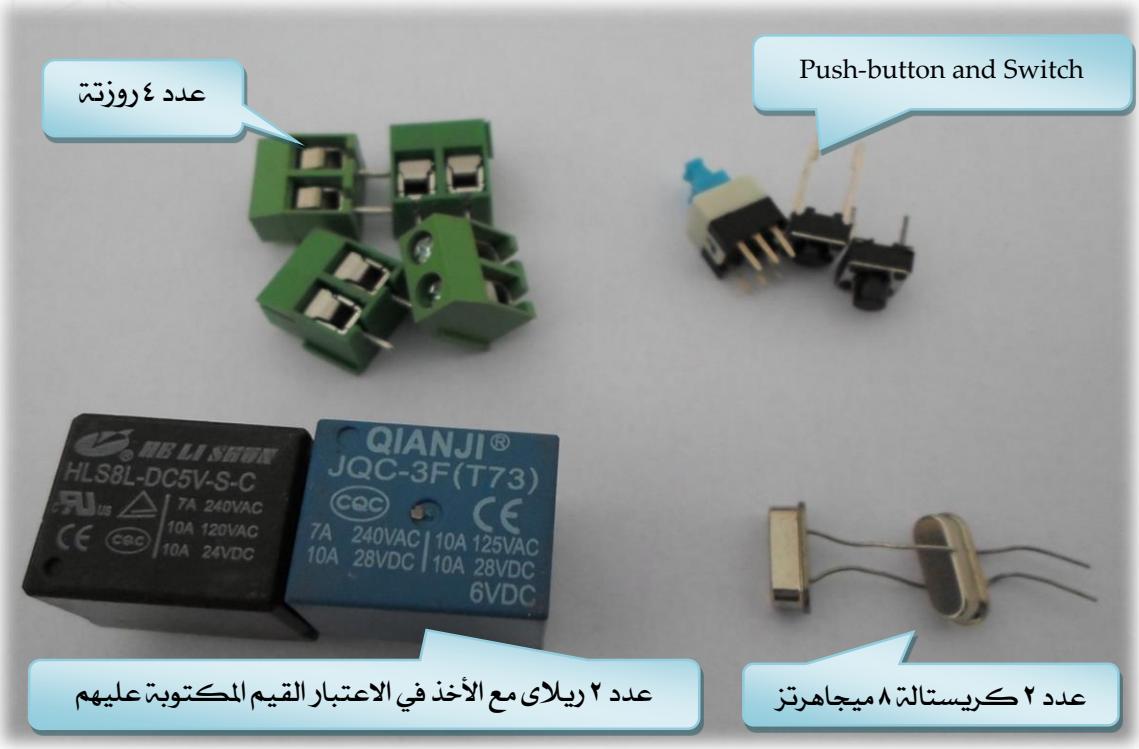
- من الممكن أن يؤثر أيضا حجم الذاكرة في اختيار نوع الميكرو، فاذا كان البرنامج الذي تريد تنفيذه برنامج كبير في حجمه عند التخزين فسوف يحتاج بالتأكيد ذاكرة كبيرة ولذا سنحتاج ميكرو له ذاكرة تتناسب مع البرنامج وسنعرف لاحقا كيف يمكن معرفة حجم الذاكرة التي يشغلها البرنامج.

هذه الصورة توضح مجموعة من الميكروكنترولر من عائلات مختلفة موضحا عليها مميزات كل واحد

PIC #	# of pins	I/O pins	Program ROM words	File RAM bytes	EEPROM bytes	Analogue inputs	Timers 8/16 bits	Max clock (MHz)	Internal osc. (MHz)	In-circuit debug	Serial comms
12F675	8	6	1k	64	128	4x10-bit	1/1	20	4	YES	NO
16F628A	18	16	2k	224	128	NO	2/1	20	4	NO	UART
16F630	14	12	1k	64	128	NO	1/1	20	4	YES	NO
16F648A	18	16	4k	256	256	NO	2/1	20	4	NO	UART
16F676	14	12	1k	64	128	8x10-bit	1/1	20	4	YES	UART
16F73	28	22	4k	192	NO	5x8-bit	2/1	20	NO	NO	ALL
16F77	40	33	8k	368	NO	8x8-bit	2/1	20	NO	NO	ALL
16F818	18	16	1k	128	128	5x10-bit	2/1	20	8	YES	I2C,SPI
16F84	18	13	1k	64	64	NO	1/0	10	NO	NO	NO
16F84A	18	13	1k	64	64	NO	1/0	20	NO	NO	NO
16F88	18	16	4k	368	256	7x10-bit	2/1	20	8	YES	ALL
16F874A	40	33	4k	192	128	8x10-bit	2/1	20	NO	YES	ALL
16F876A	28	22	8k	256	368	5x10-bit	2/1	20	NO	YES	ALL
16F877A	40	33	8k	256	368	8x10-bit	2/1	20	NO	YES	ALL
18F2320	28	25	4k	512	256	10x10-bit	1/3	40	8	YES	ALL
18F6520	64	52	16k	2048	1024	12x10-bit	1/3	40	NO	YES	ALL
18F8621	80	68	32k	3840	1024	16x8-bit	1/3	40	10	YES	I2C,SPI
18F8720	80	68	64k	3840	1024	16x10-bit	1/3	40	NO	YES	ALL

المكونات المطلوبة لمشاريع الكتاب

وفيما يلي إجمال لكل المكونات الإلكترونية المطلوبة لتنفيذ أي مشروع هاردوير عملي موجود في أي فصل في هذا الكتاب:

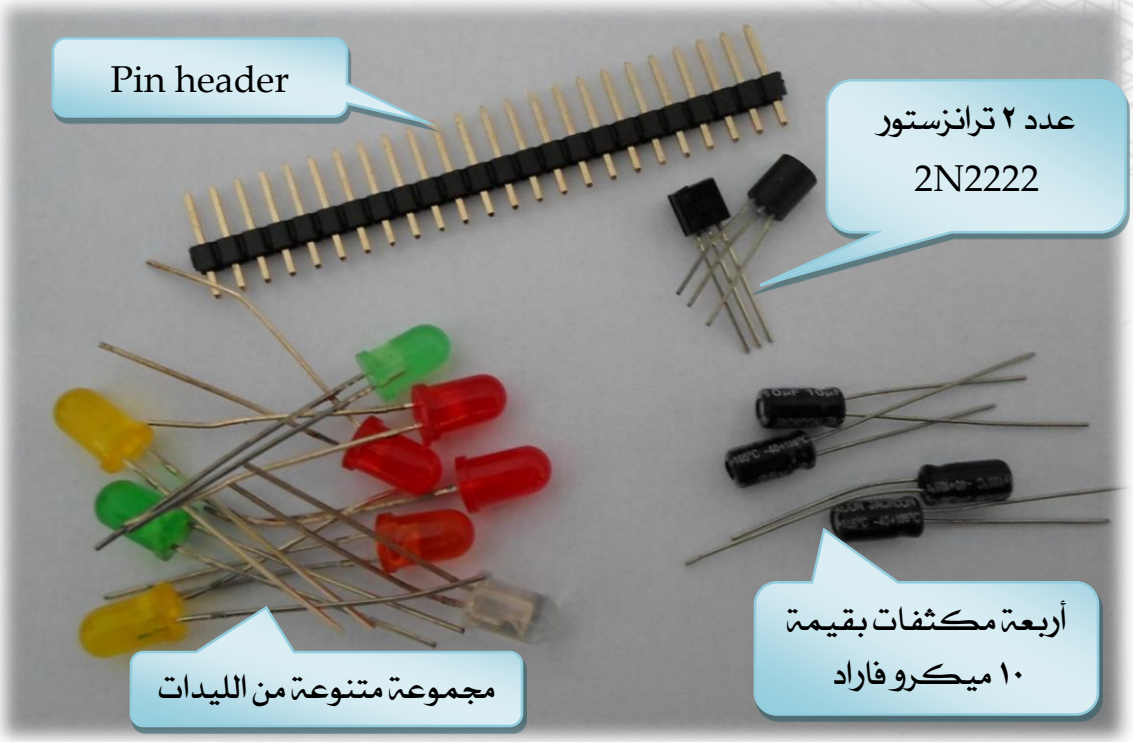


الروزتة: تستخدم عادة لتوصيل الأسلاك التي يصعب توصيلها في الـ test board إذ أن توصيل سلك يستلزم كونه رفيع وعلى درجة كافية من الصلابة، فمثلا إذا أردنا توصيل سلوك الكهرباء العادية والتي سنستخدمها عند التحكم في الأحمال الـ ٢٢٠ فولت والتي تكون سميكة في القلب المعدني لها فلن يمكننا توصيلها إلا من خلال الروزتة ...

الكريستالة: تستخدم للحصول على الـ Clock التي يحتاجها الميكرو لكي يعمل ...

الريلاي: يستخدم لتوصيل الجهود العالية بالميكروكنترولر.

متر سلك نت: والذي سنستخدمه للتوصيل بين العناصر الإلكترونية



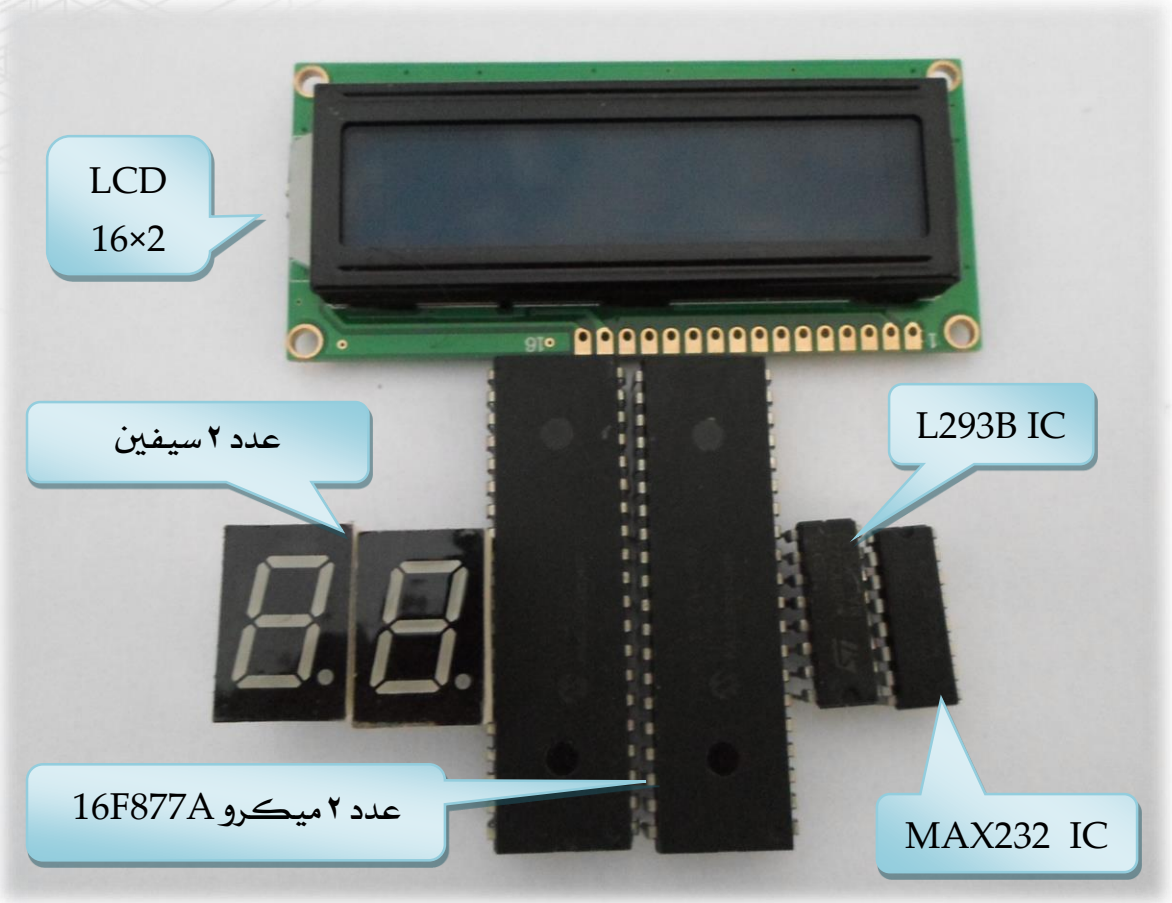
المكثفات: لها استخدامات عدة ولكن أهمها هو أننا سنحتاجها عند توصيل الميكرو بالكمبيوتر

PIN header: نحتاجها لتوصيل الشاشة بالـ test board كما سيتبين فيما بعد.

الترانزستورات: ولها استخدامات كثيرة لكننا هنا سنركز على كيفية استخدام الترانزستور كسويتش ...

DC Motor

بطارية



الشاشة: تستخدم لعرض القيم والنصوص، فعلى سبيل المثال يمكن استخدامها لعرض قيمة درجة الحرارة المقاسة ...

السيفين سيجمنت: تستخدم أيضا في عرض القيم وسنتعرف على تفاصيلها لاحقا ...

L293b: يستخدم للتحكم في الموتور...

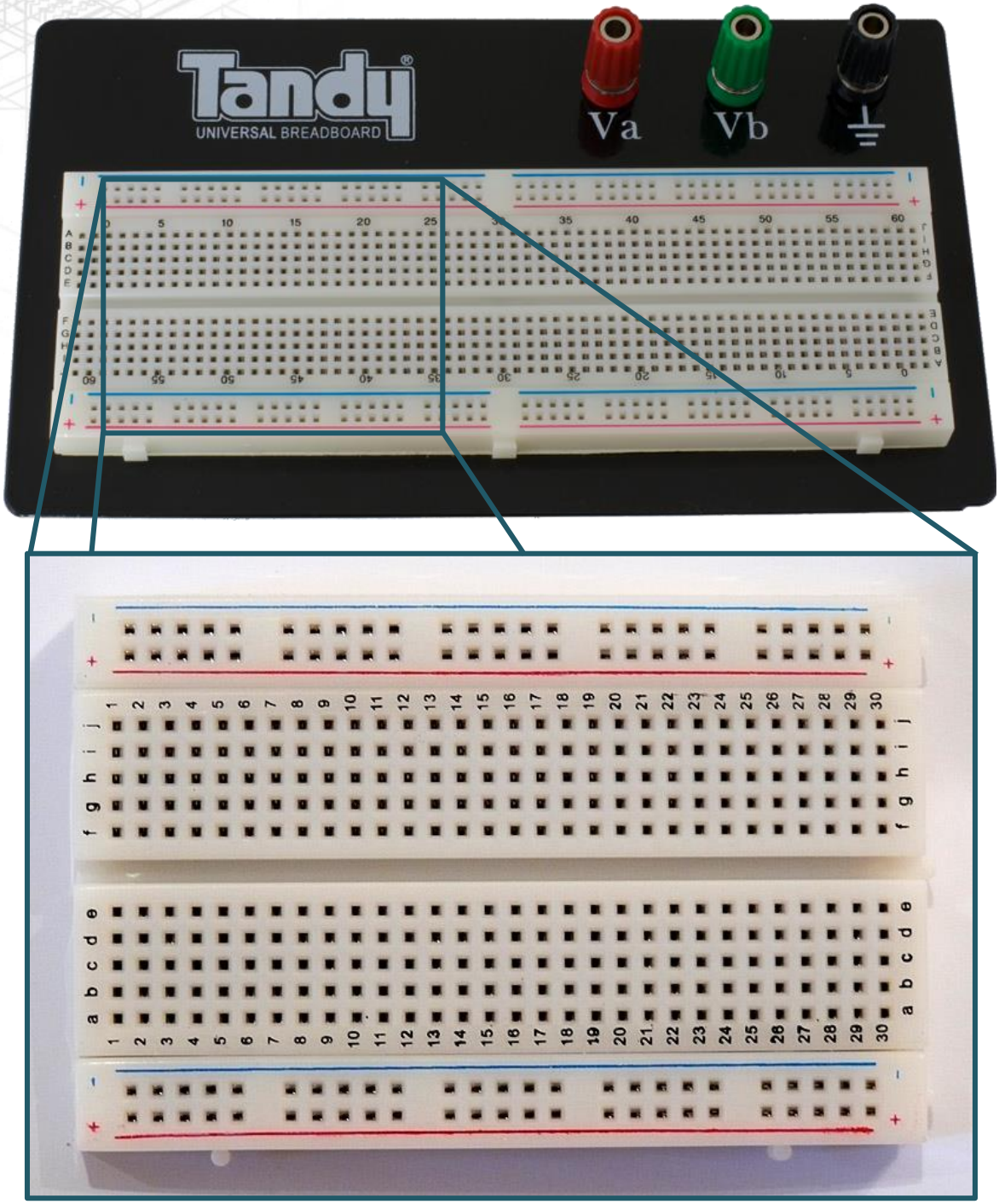
Max232: يستخدم لتوصيل الميكرو بالكمبيوتر...

7805: من المعروف أن الميكرو يعمل على جهد خمسة فولت ومعظم مصادر الجهد الموجودة قد لا تخرج خمسة فولت، فمنها مثلا ٩ أو ١٢ فولت وبالتالي نستخدم هذا الآي سي للتحويل من هذه الجهود إلى الخمسة فولت.

البروجرامر: وتستخدم لنقل الكود من الكمبيوتر إلى الميكروكنترولر، وهذه العملية تسمى حرق الميكروكنترولر.

Serial Cable: يستخدم هذا الكابل لتوصيل الميكرو بالكمبيوتر ويستخدم أيضا مع بعض أنواع البروجرامر لتوصيلها بالكمبيوتر.

Test Board:



وتستخدم لتوصيل العناصر الإلكترونية ببعضها حيث يتم تجربة الدائرة عليها أولا للتأكد من انه لا يوجد بها مشكلة ومن ثم بعد ذلك نقوم بعمل الدائرة على PCB.

وأخيرا: الآفوميتر:

والذي يستخدم لقياس الجهود والتيارات واختبار الدائرة والبحث عن بعض الأخطاء الصغيرة بها.

يمكنك شراء هذه المكونات من مصر من شركة أنور الجمال (شارع باب اللوق – التحرير) وموقعها الإلكتروني:

www.elgammalelectronics.com

أو من المملكة العربية السعودية من شركة الأساليب الذكية (مكة المكرمة – بطحاء قريش) وموقعها الإلكتروني:

www.s-m.com.sa

كما يمكنك التوجه إلى أي من الشركتين وطلب مجموعة (ميكروبيديا) وسوف تحصل على المجموعة الكاملة من المكونات المذكورة سابقا.

أو يمكنك الحصول على هذه المكونات منفردة في أي شركة مكونات إلكترونية أخرى ...

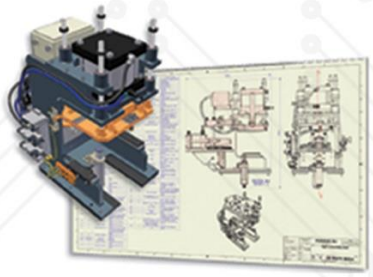
www. s - m .com.sa

SmartMethodS
الأساليب الذكية

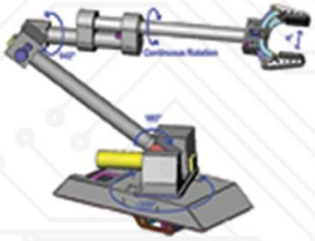
كل مايحتاجه المبتكر من أنظمة إلكترونية وميكانيكية

النظم المضمنة
Embedded System

تصميم الدوائر الإلكترونية
Electronic Circuit Design



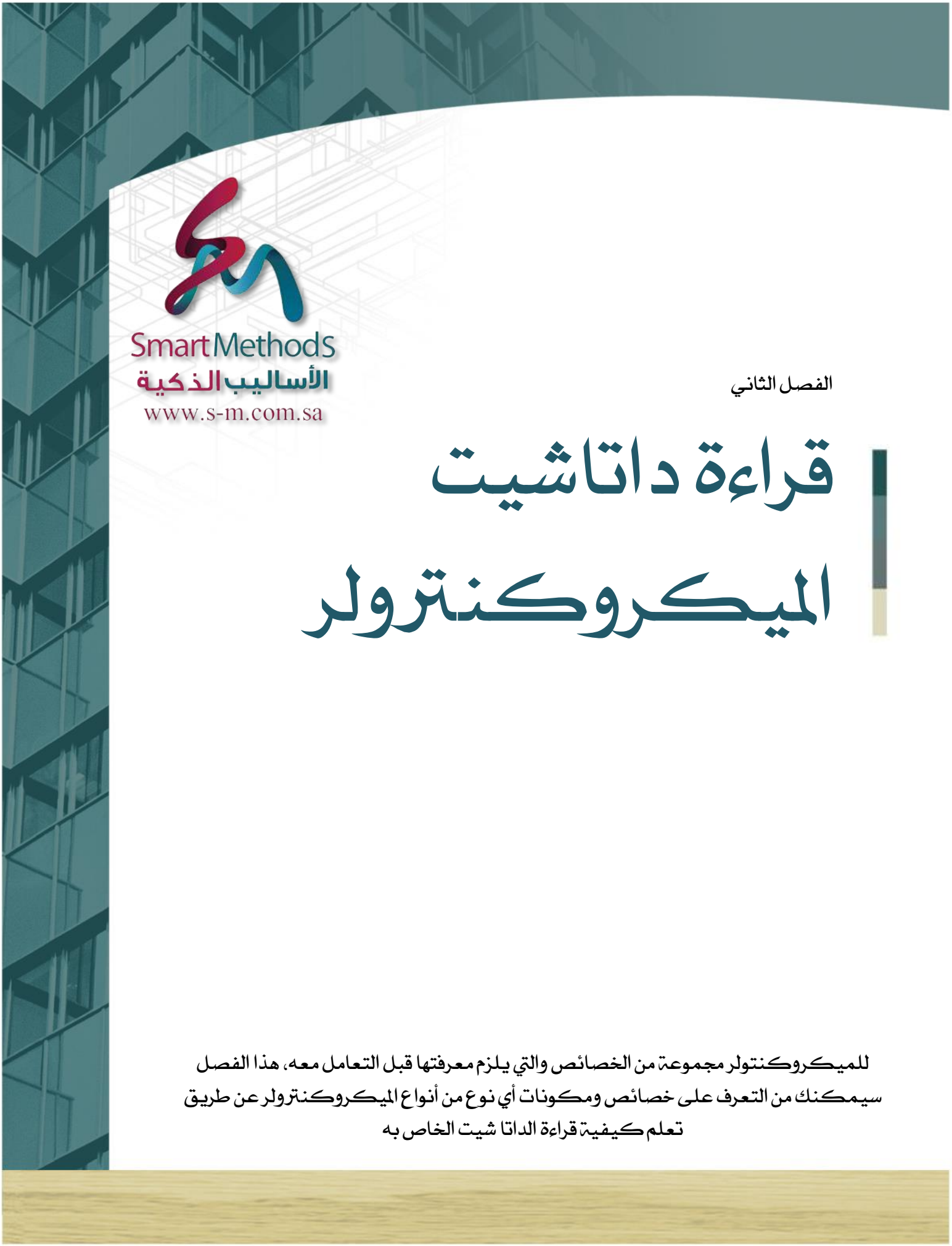
الطباعة ثلاثية الأبعاد
3D Printing



التحكم الآلي والروبوت
Robot & Control System

برمجة تطبيقات
App Programming

تطبيقات الذكاء الصناعي
Artificial intelligence &
Fuzzy Logic Application



الفصل الثاني

قراءة داتاشيت الميكروكنترولر

للميكروكنتولر مجموعة من الخصائص والتي يلزم معرفتها قبل التعامل معه، هذا الفصل سيمكنك من التعرف على خصائص ومكونات أي نوع من أنواع الميكروكنترولر عن طريق تعلم كيفية قراءة الداتا شيت الخاص به

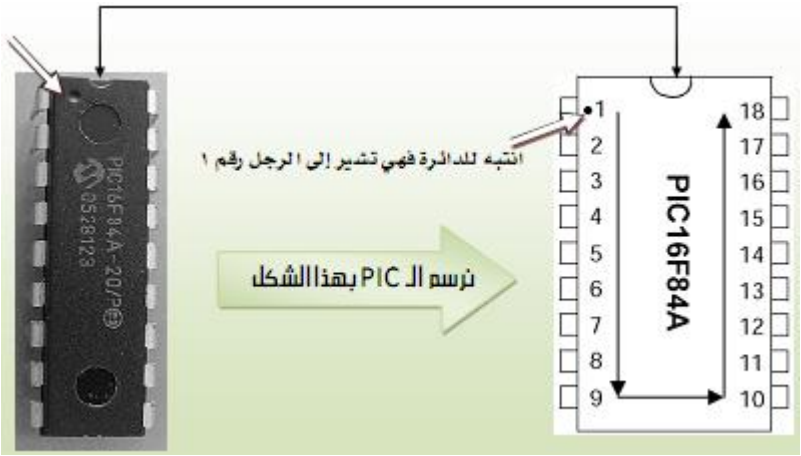
الحصول على الداتا شيت

يمكنك تنزيل ملفات الداتا شيت للمكونات الإلكترونية المختلفة من مواقع الداتا شيت المختلفة ومنها على سبيل المثال:

www.alldatasheet.com
www.datasheetcatalog.com

حيث يمكنك البحث عن أي آي سي تريده عن طريق البحث عن رقمه في أحد هذه المواقع، وطبعا يمكنك البحث في جوجل.

ترقيم رجول الميكروكنترولر



طريقة الترقيم الرجول لأي IC ثابتة ومعروفة وهي أننا نقوم بتحديد مكان النقطة المحفورة الموجودة على الـ IC فتكون الرجل التي بجوارها هي الرجل رقم واحد ثم زيادة العد مع الدوران في اتجاه الأسهم الموضحة في الشكل.

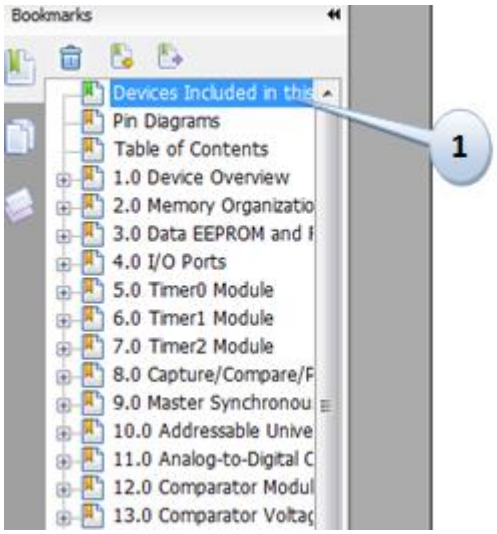
الـ Clock

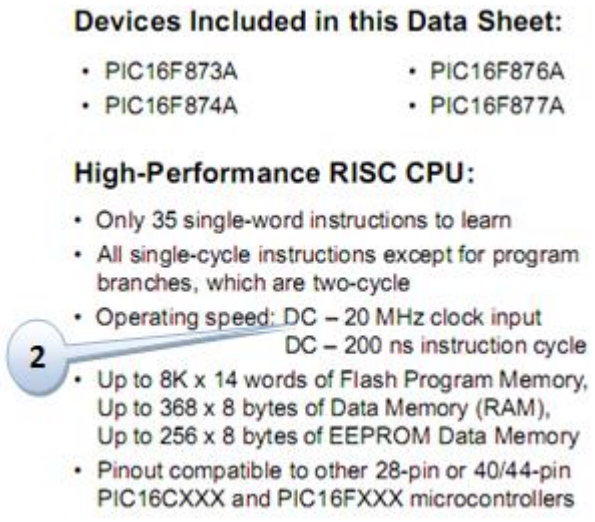
من المعلوم لدينا انه لا يمكن للسيارة أن تسير بدون وقود وكذلك لا يمكن للميكرو أن يعمل بدون الـ clock والجهد الخمسة فولت.

حيث أنه من وظائف الـ clock أنها تحدد سرعة تنفيذ الميكروكنترولر للأوامر، كما يحتاجها الميكروكنترولر في تنفيذ بعض الوظائف الخاصة التي تطلب منها مثلا تنفيذ أمر معين بعد مدة زمنية محددة.

والـclock عبارة عن إشارة كما بالشكل التالي:

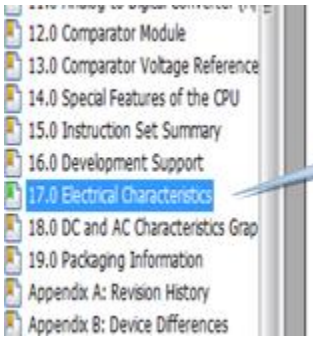
ومن أهم خواصها هو التردد الذي تصدر به هذا الإشارة أو الذبذبات التي في الشكل، كما أن لكل ميكروكنترولر قيمة قصوى للتردد يمكن له أن يتعامل معها وهي مثلا ٢٠ ميجا هرتز في حالة الميكرو 16F877A وهذه القيمة نحصل عليها من الداتا شيت الخاصة به.





بعض الخواص الكهربائية الهامة للميكرو PIC16F877A

سوف نتناول الداتا شيت الخاص بهذا الميكرو الشهير كمثال لكيفية استخراج المعلومات الهامة من الداتا شيت.



أقصى تيار يمكن لهذا الميكرو أن يقوم بإخراجه (في حالة تشغيل رجوله كخرج) أو استقباله (في حالة تشغيل رجوله كدخل) هي ٢٥ ميللي أمبير لكل رجل من رجوله، وبالتالي فان الأحمال التي تعمل على تيار أكبر من هذه القيمة لا يمكن توصيلها مباشرة على الميكروكنترولر ...

مثال: هل يمكننا توصيل الليد مباشرة على رجل من رجول الميكروكنترولر لكي يتحكم في أضاءته ؟؟؟

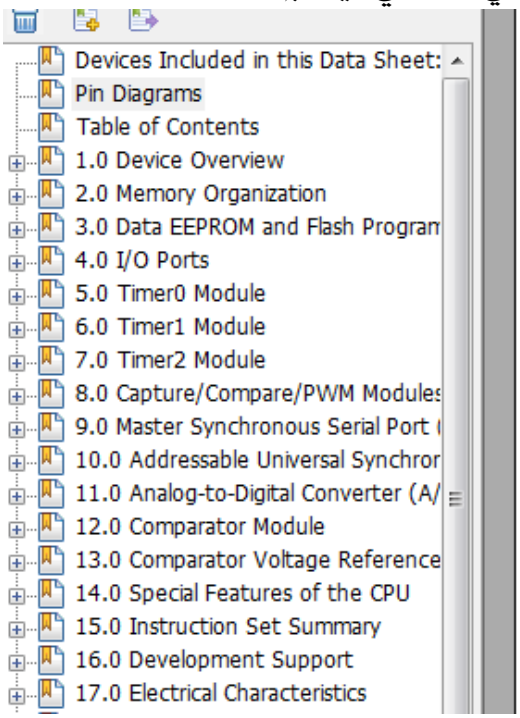
الإجابة: نعم، لأن الليد يعمل على جهد من ١,٥ إلى ٢ فولت وتيار قد لا يزيد عن ١٥ ميللي أمبير وهي قيم أقل من القيم التي يقوم الميكرو بإخراجها للفولت وللتيار.

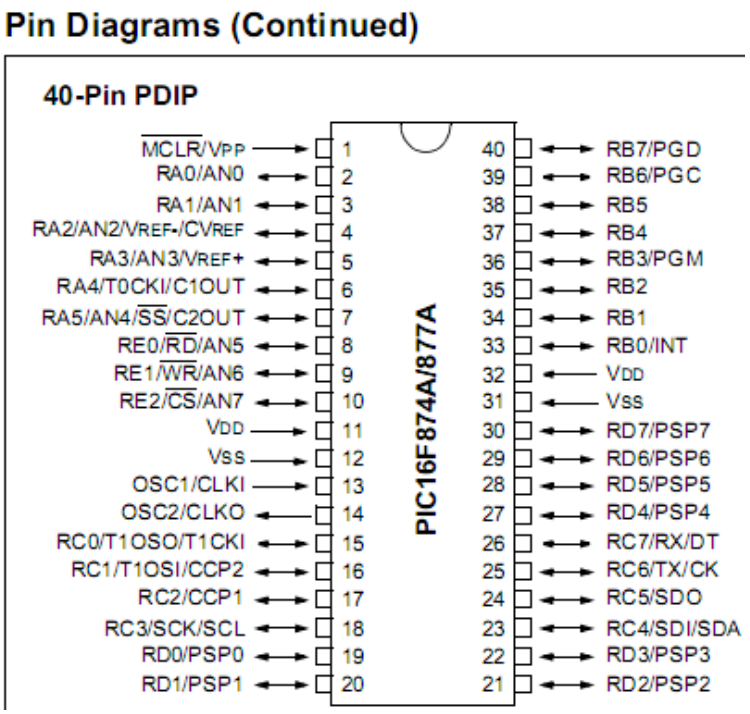
مثال: هل يمكننا توصيل موتور يعمل على ٥ فولت و١٠٠ ميللي أمبير على رجل من رجول الميكروكنترولر ؟؟؟

الإجابة لا، فعلى الرغم من أن جهد الموتور خمسة فولت مساوي لجهد الميكرو إلا أن التيار الذي يحتاجه الموتور أكبر من الذي يقوم الميكرو بإخراجه.

التعرف على بعض وظائف رجول الميكروكنترولر

هذا الجزء يوضح كيفية معرفة وظيفة كل رجل من رجول الميكروكنترولر وحيث أنه قد يكون لرجل واحدة أكثر من وظيفة فسوف نتعرض الآن لبعضها وسنذكر الباقي كل في حينه بإذن الله.

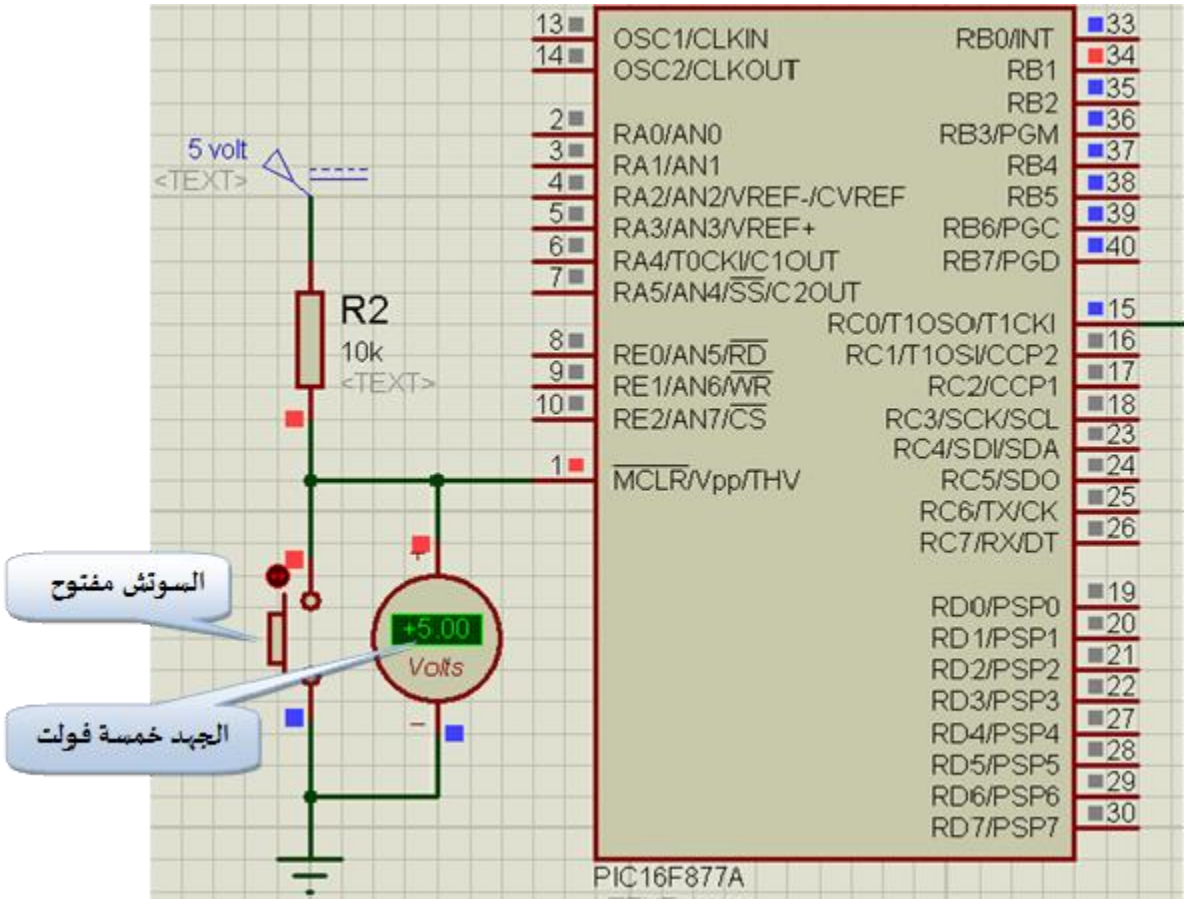




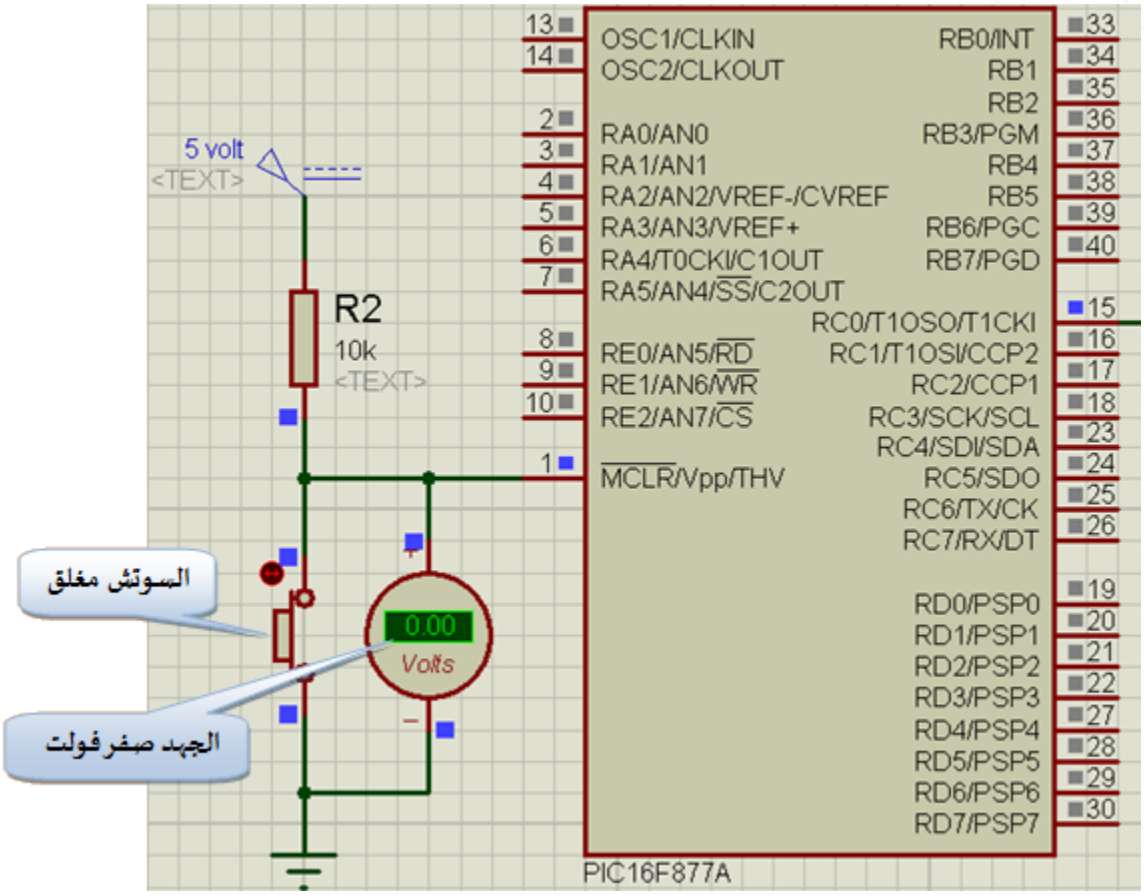
الميكرو 16F877A الذي اتخذناه كمثال في شرح هذا الباب يتكون عن ٤٠ رجل كما بالشكل السابق.

الرجل رقم (١)

من الشكل السابق المأخوذ من الداتا شيت فإن هذه الرجل مكتوب عليها MCLR وهو لفظ يرمز إلى master clear وهذا يعني أنها تستخدم لعمل إعادة تشغيل للميكروكنترولر أي إعادة بدأ تنفيذ البرنامج من بدايته وهذا يتضح جليا في مسابقات الروبوكون (Robocon competition) وفيها مثلا اذا فقد الروبوت مساره أو حدثت له مشكلة ما فان المشرف عليه يقوم بوضعه في مكانه الأصلي ثم يقوم بإعادة التشغيل من جديد عن طريق مفتاح يتم تركيبه على هذه الرجل، ومن الملاحظ وجود شرطة أعلى كلمة MCLR وهذه الشرطة دائما ما تعني أن هذه الرجل تعمل Active low أو بمعنى آخر فإن هذه الرجل ستقوم بوظيفتها أي ستقوم بعمل إعادة للتشغيل اذا وضع عليها جهد Low أي اذا وضع عليها صفر فولت وبالتالي فان الميكرو لكى يعمل بشكل طبيعي ولا يعيد التشغيل باستمرار فلابد أن توصل هذه الرجل بخمسة فولت دائما وعندما تحدث مشكلة ما ونريد عمل إعادة تشغيل نوصل عليها صفر فولت ... كيف ذلك؟؟؟ يتم ذلك من خلال الدائرة الموضحة بالشكل التالي:



كما نرى فإن دائرة هذه الرجل تحتوي على مفتاح يعرف بالـ push button ومقاومة ١٠ كيلو أوم ومصدر جهد ٥ فولت موصل كما هو بالشكل، نلاحظ انه في حالة التشغيل العادية أي عندما يكون السويتش غير مضغوط عليه يكون الخرج ٥ فولت وبالتالي يعمل الميكروكنترولر. وعندما يتم الضغط على السويتش فان الجهد على الرجل MCLR يصبح صفر وبالتالي يحدث إعادة تشغيل:



وأيضا تستخدم هذه الرجل عند برمجة الميكرو (أي عند تنزيل البرنامج عليه) لكن هذا لا يهمنا الأن.

الرجول (١١) و (١٢)

ذكرنا من قبل أن الميكرو لن يعمل بدون جهد موصل عليه وقيمته تساوى خمسة فولت وهذا الجهد يوصل على هذين الطرفين بحيث يوصل الطرف الموجب للخمسة فولت على الرجل ١١ والطرف السالب على الرجل ١٢، وبالنظر في الداتا شيت نستطيع استنتاج التالي: الرجل ١١ مكتوب عليها VDD وبالتالي نستطيع استنتاج أن الرجل التي يكتب عليها VDD في أي نوع آخر من أنواع الميكرو هي

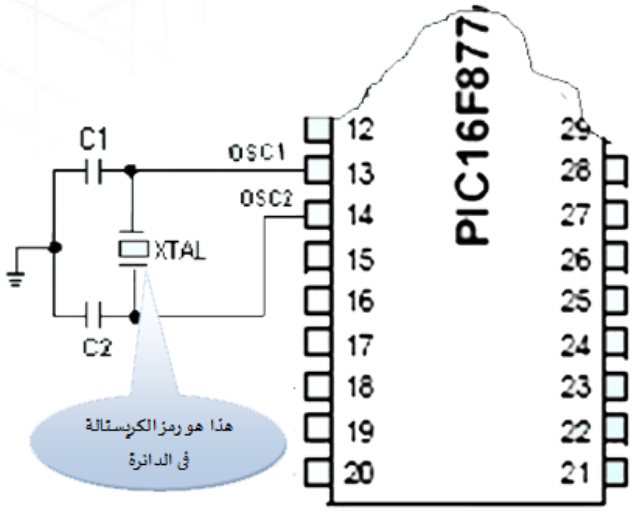
التي يوصل عليها الطرف الموجب للجهد بصرف النظر عن رقمها، والرجل ١٢ مكتوب عليها VSS وهى الرجل التي دائما توصل بسالب البطارية الخمسة فولت أو توصل بالأرضي Ground.

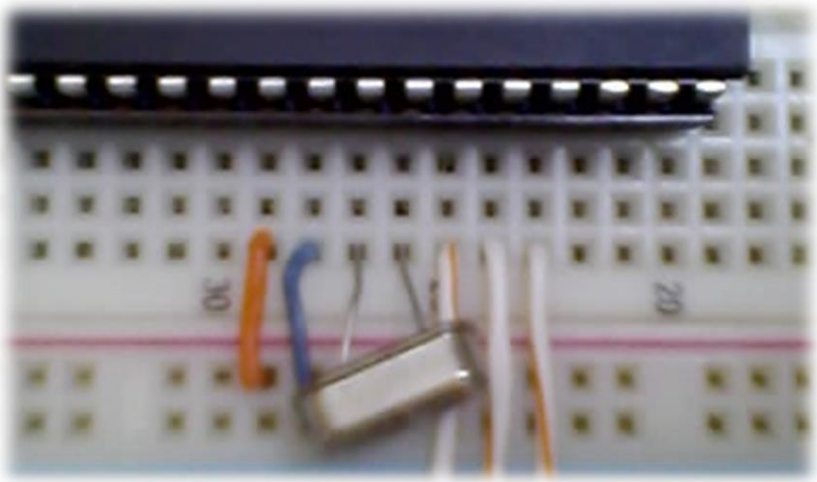
الرجول (٣١) و (٣٢)

نفس الاستخدام للرجلين ١١ و١٢ (الرجل ٣١ توصل على الموجب و٣٢ توصل على السالب) ويمكن الاكتفاء بتوصيل ١١ و١٢ فقط.

الرجول (١٣) و (١٤)

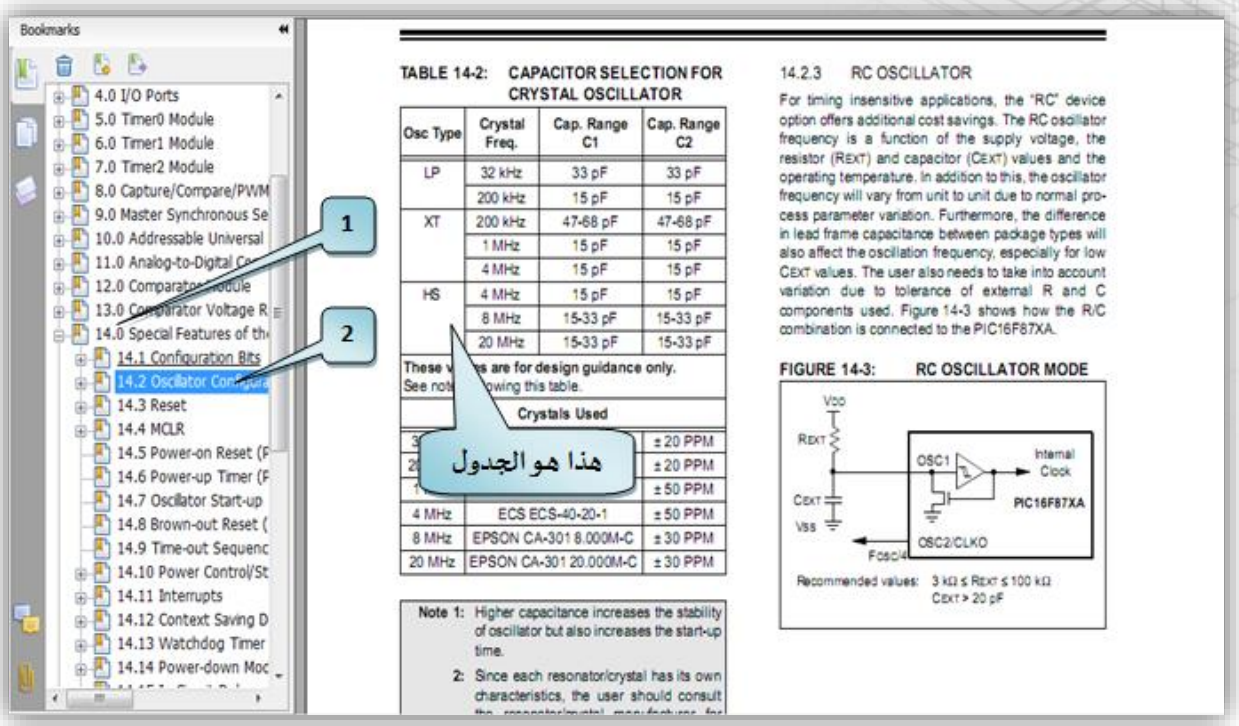
يوصل عليهما الكريستالة (Oscillator) والتي تستخدم لتوليد إشارة الـ clock التي بدونها لن يعمل الميكرو، والكريستالات أنواع تختلف في قيم ترددات الإشارة التي تخرجها منها ١ أو ٤ أو ٨ أو ٢٠ ميجا هرتز و غير ذلك، و فى هذا الكتاب سنعمل على الكريستالة ٨ ميجا هرتز.





دائرة الكريستالة

يظهر في الأشكال السابقة التوصيل الصحيح للكريستالة، حيث يتم توصيل مكثفين مع الكريستالة كما هو موضح، وتتوقف قيم هذه المكثفات هذه على قيمة الكريستالة ويمكن الحصول على هذه من الداتا شيت الخاصة بالميكرو كما بالشكل الآتي:



ويتضح من هذا الجدول أن هذا الميكرو يعمل على 3 modes وهم LP، XT ، HS وكل منهم له قيمة مختلفة للكريستالة، ولو لاحظت المود المستخدم عند التردد ٨ ميجاهرتز لوجدته HS وهذا ما سيجعلنا نختاره عندما نقوم بعمل مشروع جديد في الفصول التالية ... وبالتالي عندما تتعامل مع أي نوع آخر من أنواع الميكروكنترولر يجب معرفة الـ Modes التي يعمل عليها وقيم الكريستالات الخاصة بكل Mode ثم نقوم بالاختيار الصحيح عند عمل مشروع جديد، إذ أن اختيار مود خاطئ سيؤدي إلى عدم أن المشروع لن يعمل على مستوى الهاردوير وان عمل في السوفتوير ...

ملحوظة: في شغلنا العملي في الفصول التالية لم نقم بتوصيل المكثفات ويمكنك أيضا فعل ذلك وسيعمل المشروع بدون مشاكل ...

ملحوظة: بعض أنواع الميكروكنترولر قد تحتوي على كريستالة داخلية وفي هذه الحالة يمكننا الاستغناء عن دائرة الكريستالة الخارجية ولكن يلزم تحديد الإعدادات في البرنامج كما سنرى لاحقا

باقي ال ٣٣ رجل

أما باقي الرجول فهم الذين يستخدموا لربط الميكروكنترولر باي جهاز خارجي مثل: مفتاح – ليد – موتور - توصيل الميكرو بالكمبيوتر – سنسور – قراءة الإشارات الأنالوج – ...) ويتم تقسيمهم إلى خمسة مخارج ports، يتضمن الجدول التالي أسماءهم وتقسيماتهم:

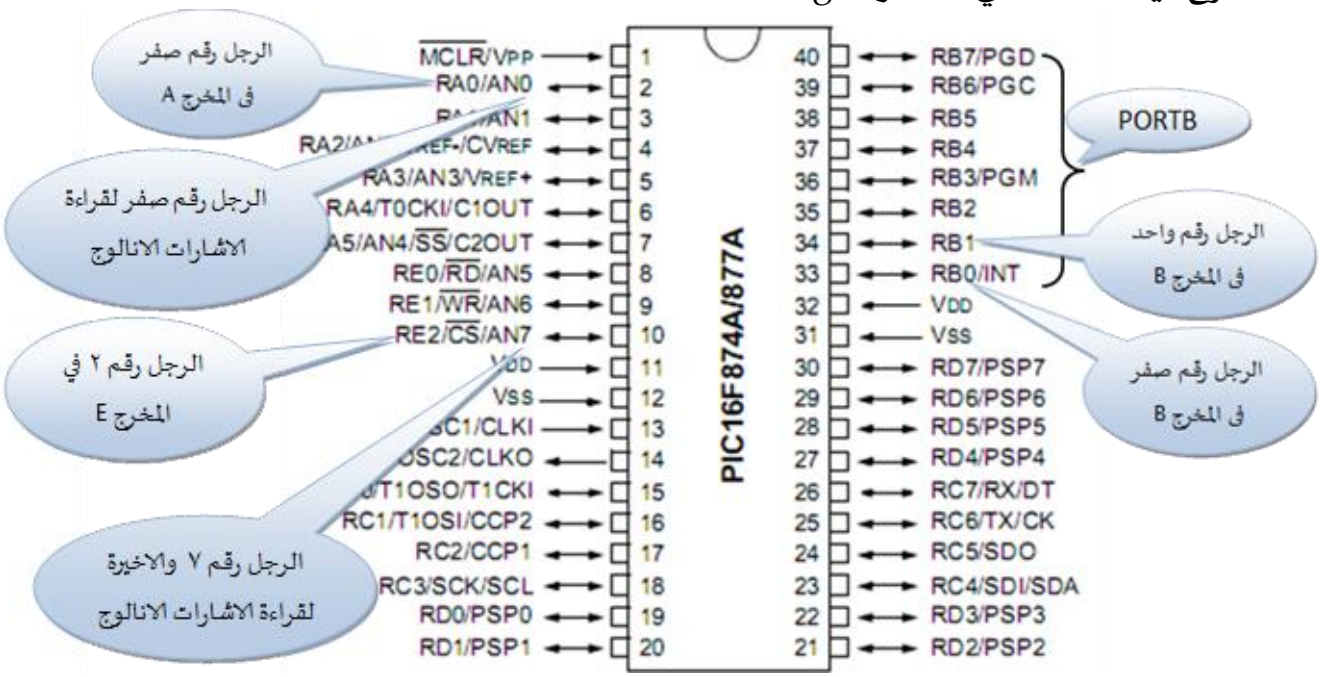
اسم المخرج	عدد رجول المخرج	رجول المخرج
PORTA	6 PINS	RA0,RA1,RA2,RA3,RA4,RA5
PORTB	8 PINS	RB0,RB1,RB2,RB3,RB4,RB5,RB6,RB7
PORTC	8 PINS	RC0,RC1,RC2,RC3,RC4,RC5,RC6,RC7
PORTD	8 PINS	RD0,RD1,RD2,RD3,RD4,RD5,RD6,RD7
PORTE	3 PINS	RE0,RE1,RE2

الإشارات الد يجيتال

يستخدم الـ ٣٣ رجل لقراءة الإشارات الد يجيتال كتلك التي يمكن أن نحصل عليها من السويتش فمثلا عند الضغط على السويتش يكون على رجل الميكرو خمسة فولت وعند تركه يكون على رجل الميكرو صفر فولت أو العكس كما في دائرة الـ MCLR المذكورة من قبل، وتستخدم أيضا هذه الرجول لإخراج قيم د يجيتال أي لإخراج خمسة فولت أو صفر.

الإشارات الأنالوج

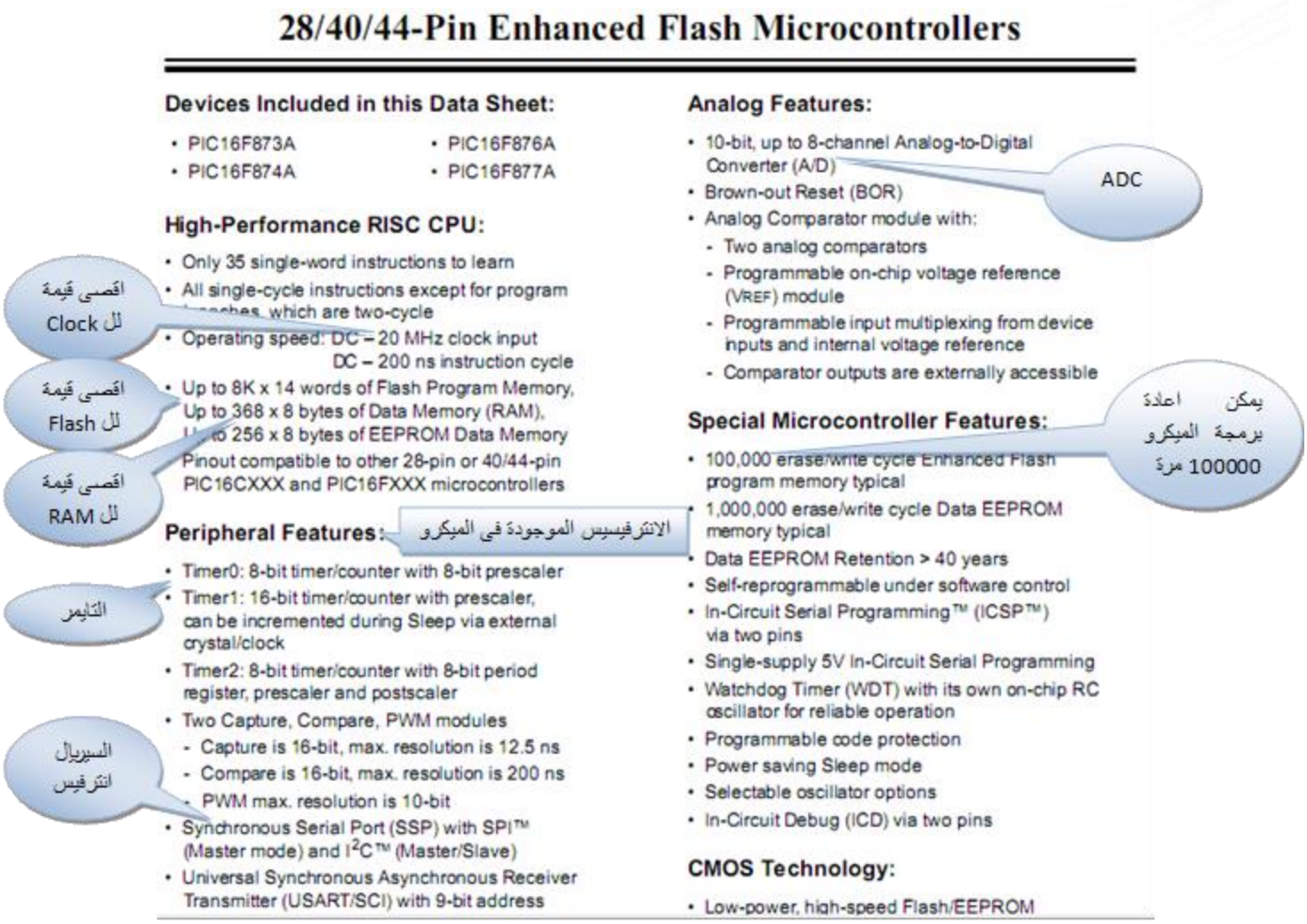
وبالإضافة لاستخدامهما مع الد يجيتال فإن كلا من المخرج E وخمسة رجول من المخرج A يستخدموا لقراءة الإشارات الأنالوج، وبالتالي لابد من تحديد هل يتم استخدامهم كأنالوج أم كد يجيتال وهو ما يتم بالبرمجة، وعلى مستوى الداتا شيت نلاحظ أن الرجل رقم ٢ مكتوب بجوارها RA0 أي انه الرجل رقم صفر في المخرج A، ومكتوب أيضا AN0 أي انه عند الرجل رقم صفر التي تستخدم لقراءة الإشارات الأنالوج حيث أن AN هي اختصار Analogue.



ملحوظة:
يوجد وظائف أخرى لرجول الميكروكنترولر سيتم توضيحها في حينها بإذن الله ...

معلومات أخرى

الصور التالية توضح بعض المعلومات الإضافية الأخرى التي يمكن الحصول عليها من الداتا شيت الخاص بهذا الميكرو:



28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM),
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (VREF) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low-power, high-speed Flash/EEPROM

سلطان

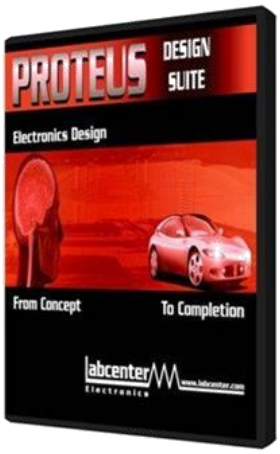
الفصل الثالث

البرامج المستخدمة مع الميكروكنترولر

من بين كل البرامج التي تتعامل مع الميكروكنترولر سنحدد في هذا الفصل البرامج التي سنستخدمها في هذا الكتاب مع النوع PIC وكيفية الحصول عليها واستخدامها وتنصيب ما يصعب تنصيبه منها الكومبيوتر

في هذا الباب سنتعرف سويا على البرامج التي سنحتاج إليها وكيفية تنزيلها وتسطيبها على الجهاز، تختلف هذه الطريقة باختلاف أرقام الإصدارات من البرامج وكذلك وصلات التنزيل من الإنترنت وأشكال المواقع ولذا فإن الطريقة المذكورة في هذا الكتاب هي تبعا آخر إصدارات البرامج وأشكال المواقع في وقت إصداره وغالبا ما تكون الاختلافات طفيفة بالاختلاف في رقم الإصدار ...

أهم هذه البرامج هو الـ MikroC والذي يستخدم في كتابة كود المشروع المراد من الميكرو أن يقوم بتنفيذه وتصحيح الأخطاء فيه وتوليد الملف الـ Hexadecimal أيضا والذي يتم حرقه على الميكروكنترولر (أي الملف الذي يمثل البرنامج والوظيفة التي نريد للميكرو تنفيذها لكن بصيغة الـ Hexadecimal).



بالإضافة إلى برنامج المحاكاة الشهير Proteus وهو برنامج يستخدم لمحاكاة الدوائر الإلكترونية، وهذا طبعا مفيد في اكتشاف الأخطاء (كما سيتبين لاحقا إن شاء الله) ويوفر علينا الكثير من الوقت حيث نحاكي الدائرة على الكومبيوتر ونتوقع الخرج منها ونغير أخطاءها على التصميم بدل من التجربة عن طريق تنفيذ الدائرة الهاردوير التي قد ينتج عنها خسائر.

برنامج الميكرو سي MikroC

برنامج MikroC PRO for PIC من تصميم وإنتاج شركة (ميكرو إلكترونيكا MikroElecktronika وموقعها على الإنترنت:

www.mikroe.com

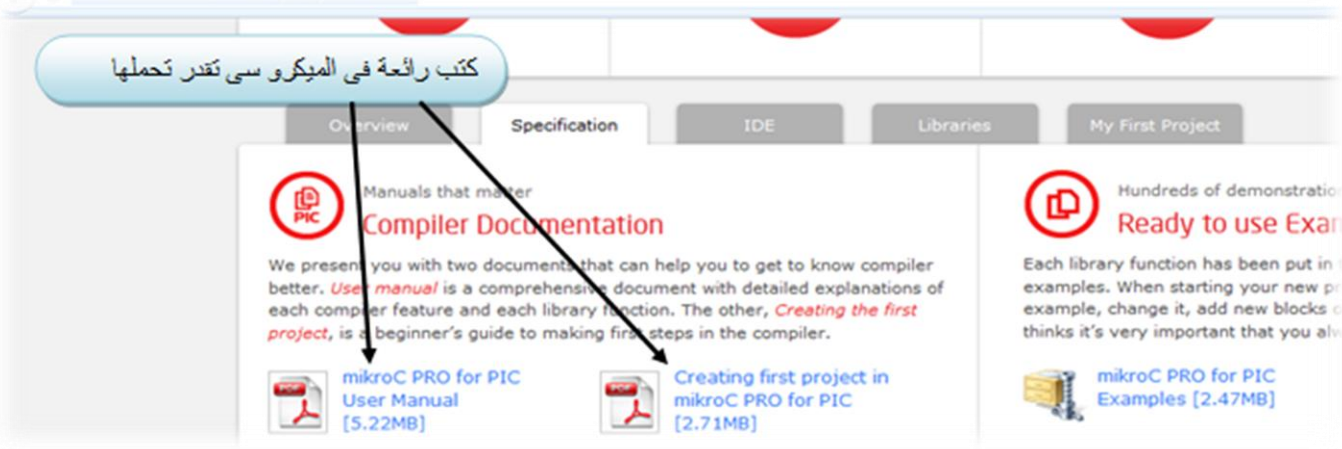
يمكن الدخول على هذا الموقع والوصول إلى ملف تحميل أحدث نسخة من البرنامج أو مباشرة من هذا اللينك:

www.mikroe.com/mikroc/pic/

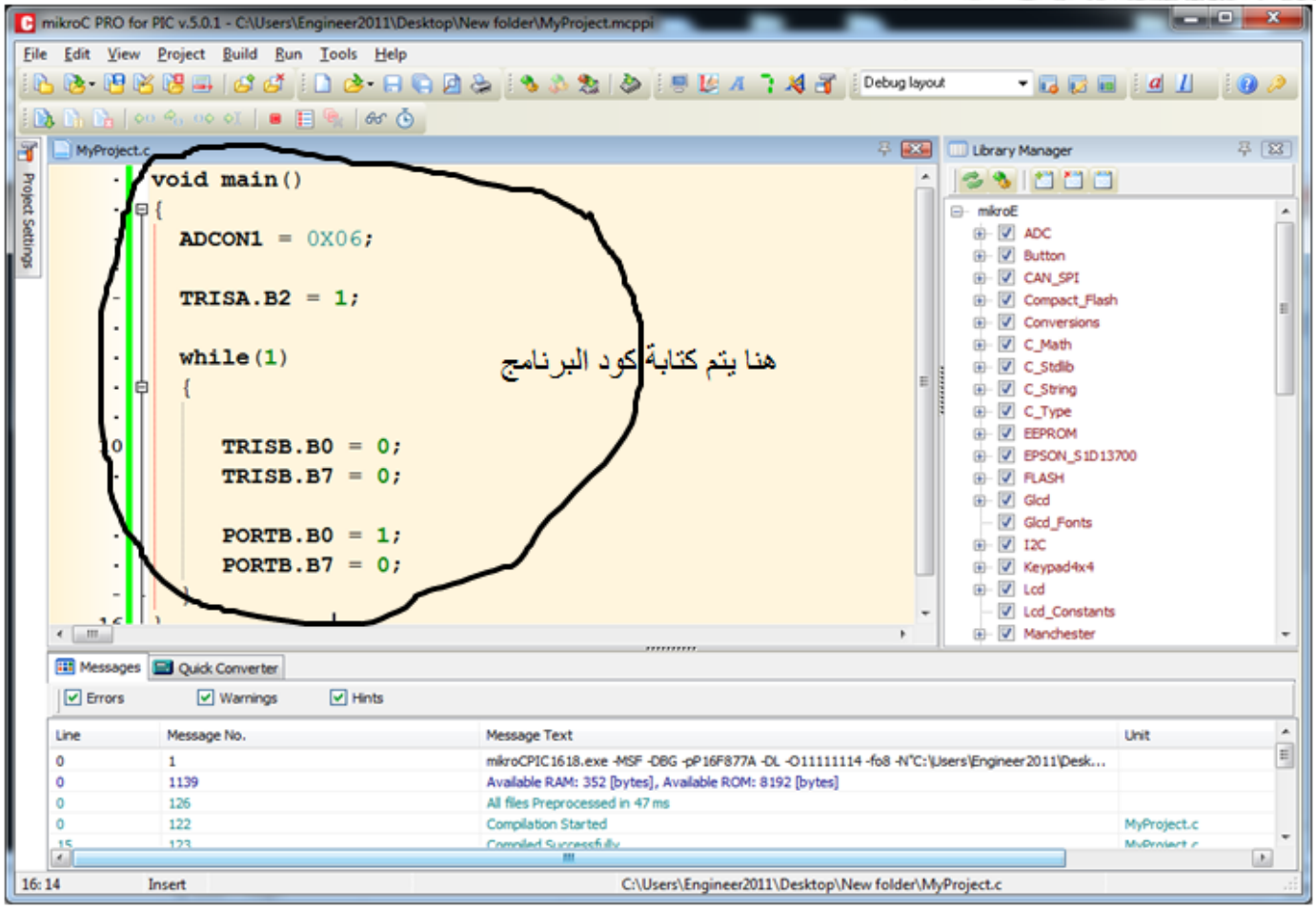
رقم الإصدار المستخدم في هذا البرنامج هو 5.61 كما هوا مكتوب تحت رابط التحميل.

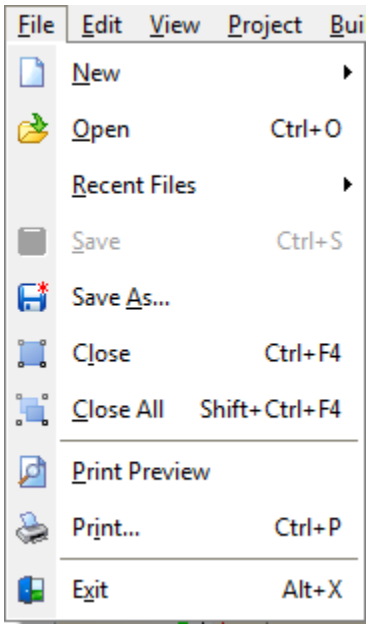
ومن نفس الصفحة يمكن الضغط على Specification أو الدخول إلى اللينك التالي وتحميل بعض الكتب والـ Manuals والأمثلة المفيدة التي تصدرها الشركة:

http://www.mikroe.com/mikroc/pic/specification/



واجهة البرنامج



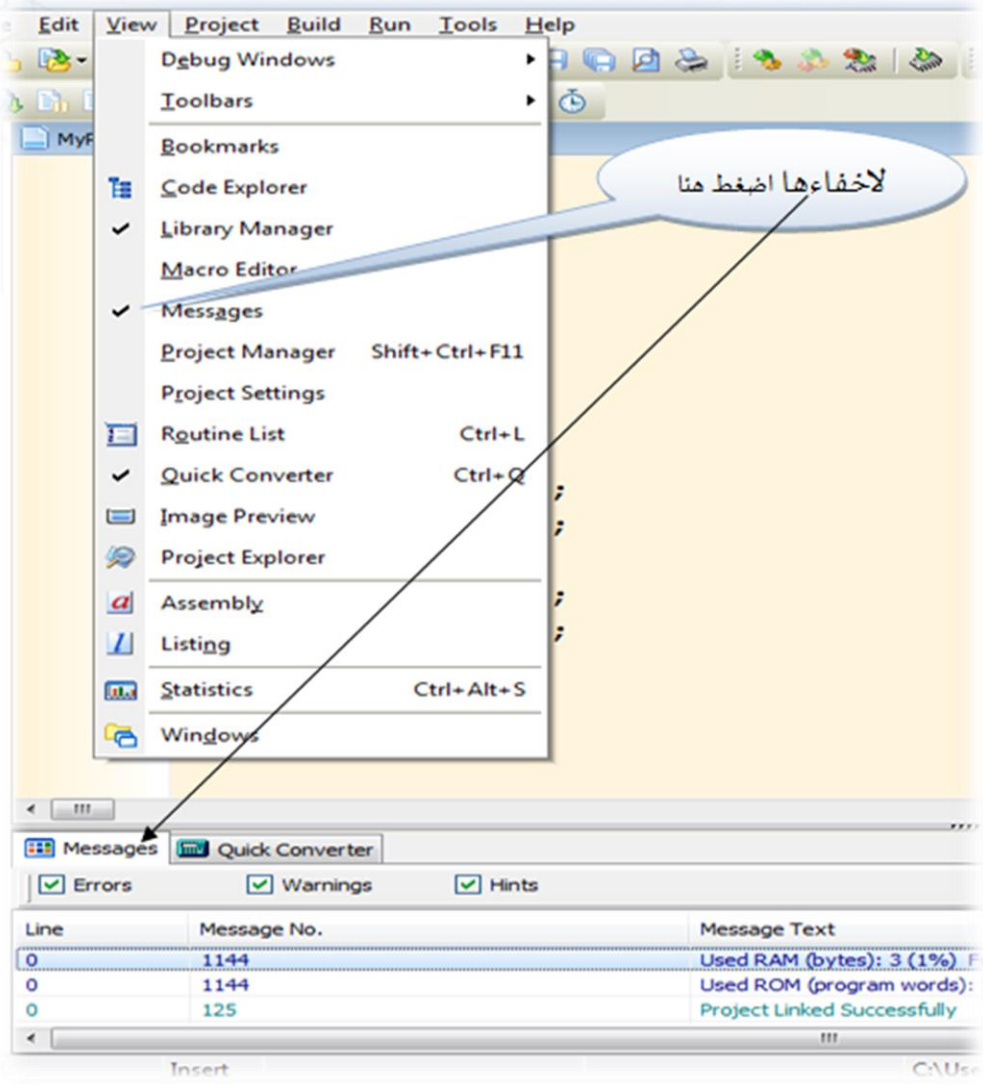


شأنه كشأن أي برنامج آخر فهو يحتوي على مجموعة من القوائم File, Edit, View, Project, Build, Tools, Help ولكل قائمة أوامرها الخاصة التي تندرج أسفلها.

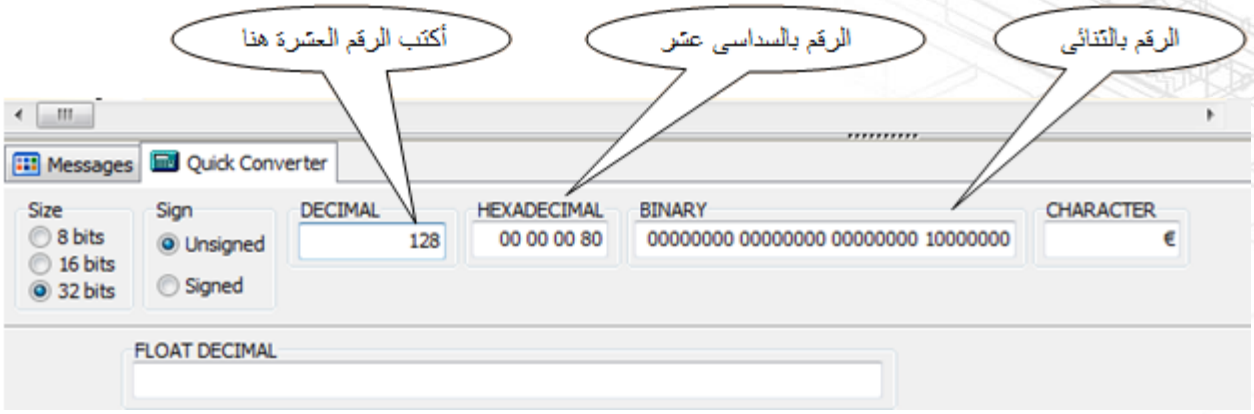
القائمة File: تستخدم لفتح ملف أو إغلاقه أو إنشاء ملف جديد أو حفظ التغييرات في الملف الحالي أو طباعة الملف أو فتح ملف من الملفات التي استخدمت مؤخرا أو إنهاء البرنامج كلية.

القائمة View: وتستخدم في إظهار وإخفاء المربعات في الواجهة ومن أهم هذه المربعات:

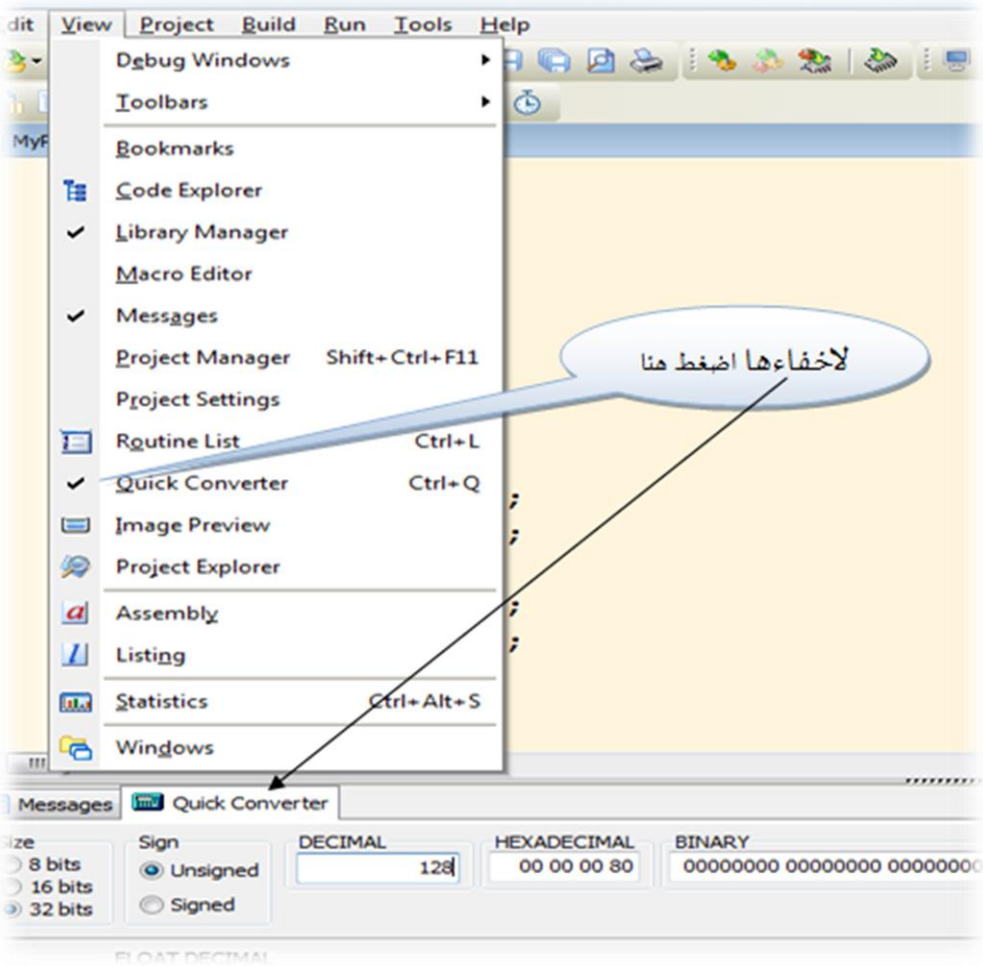
مربع الرسائل: تظهر فيه الأخطاء والتنبيهات الخاصة بالكود المكتوب ومكانها فيه ولذلك هو مهم جدا.



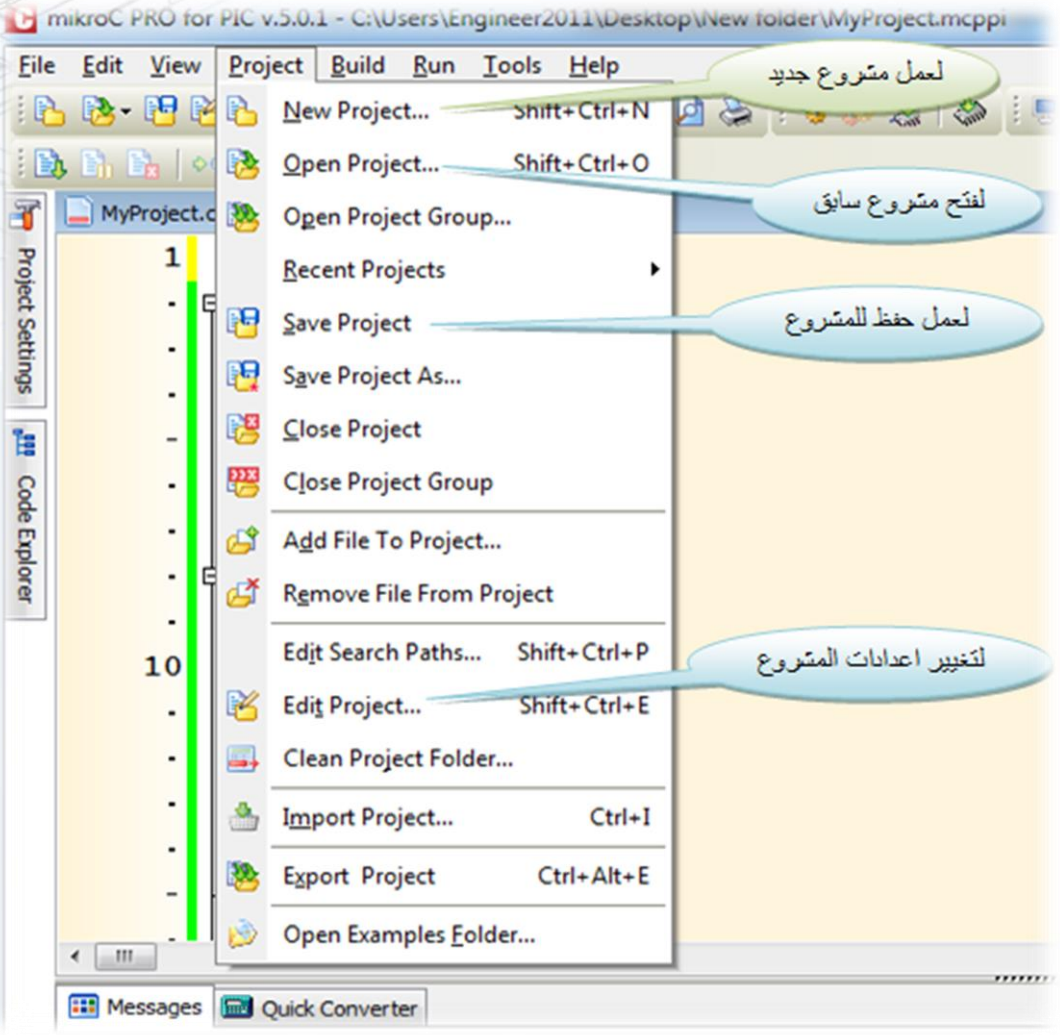
مربع التحويلات بين الأنظمة العددية: وهو مربع نستخدمه لتحويل رقم بين الأنظمة العددية المختلفة، فمثلا يمكنك تحويل أي رقم عشري إلى ثنائي أو العكس أو إلى hexadecimal أو العكس، ما عليك سوى كتابة الرقم في خانة الرقم العشري وستجد انه تم تحويله إلى ما يناظره بالثنائي وظهرت نتيجة التحويل في خانة الثنائي كما بالشكل التالي:

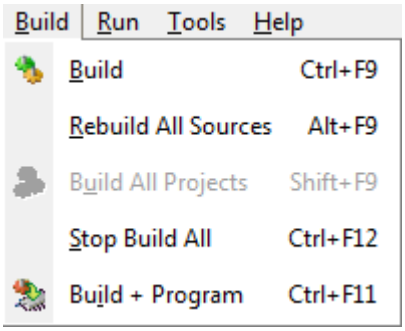


ولإخفائه أيضا:

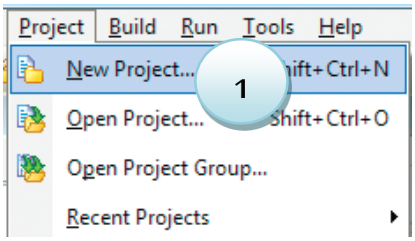


قائمة Project: وتستخدم لعمل مشروع جديد أو إغلاق مشروع أو حفظ مشروع أو تغيير إعدادات مشروع وغير ذلك فيما يخص المشروع ...



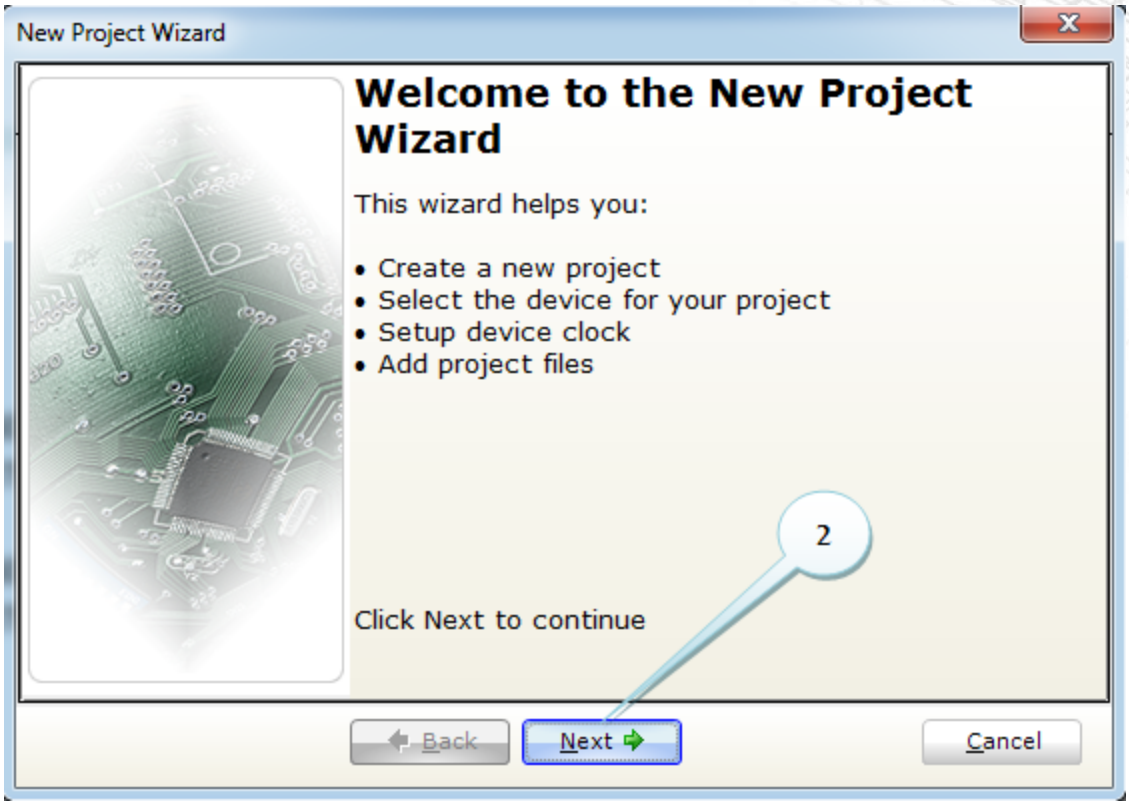


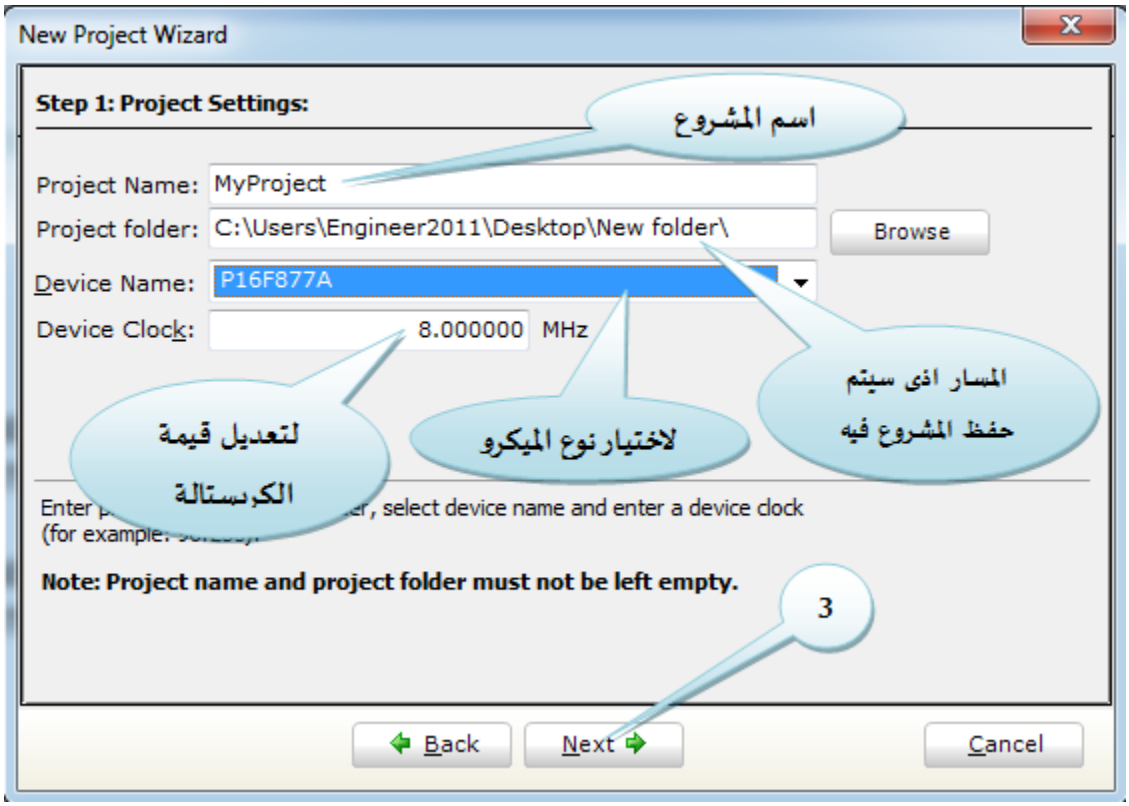
قائمة Build: وتستخدم لعمل Build أو Compile لكود البرنامج، والتي تعنى إيجاد الأخطاء في الكود وإظهارها في مربع الرسائل، وإذا لم يكن هناك أخطاء يتم توليد ملف الـ hexadecimal في المجلد المحفوظ فيه المشروع والذي يستخدم لحرق البرنامج على الميكرو كما ذكرنا سابقا.

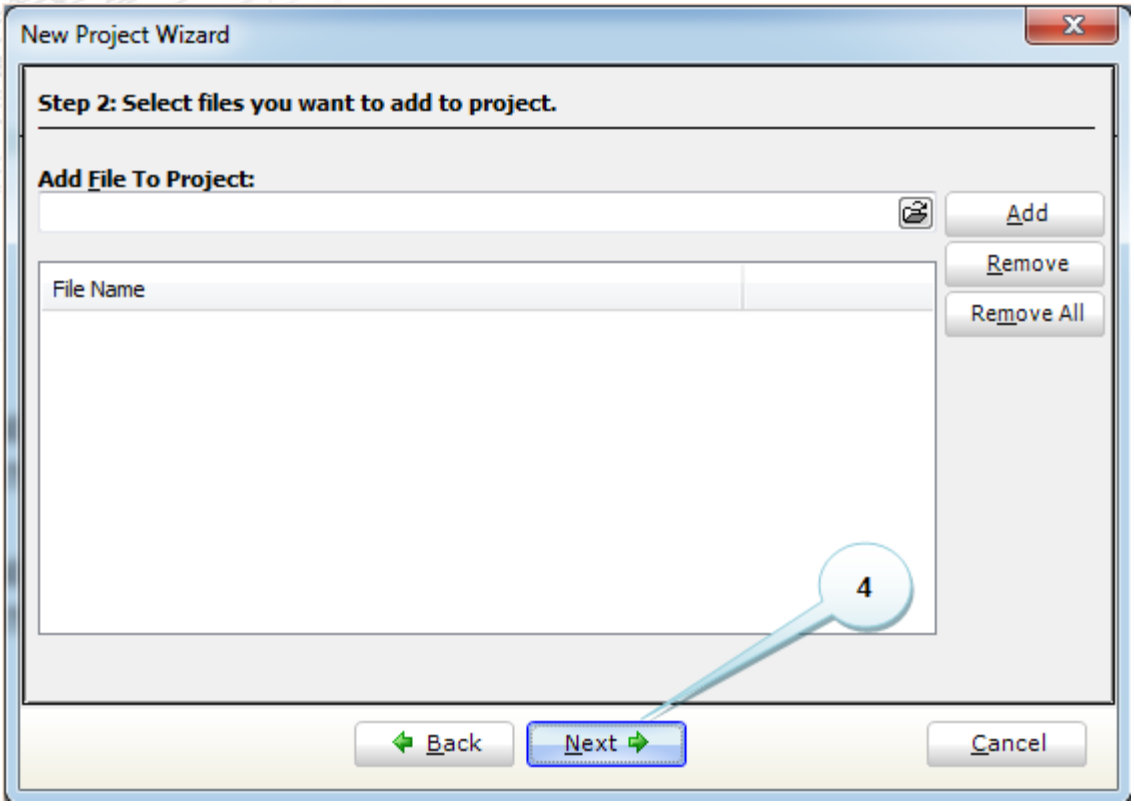


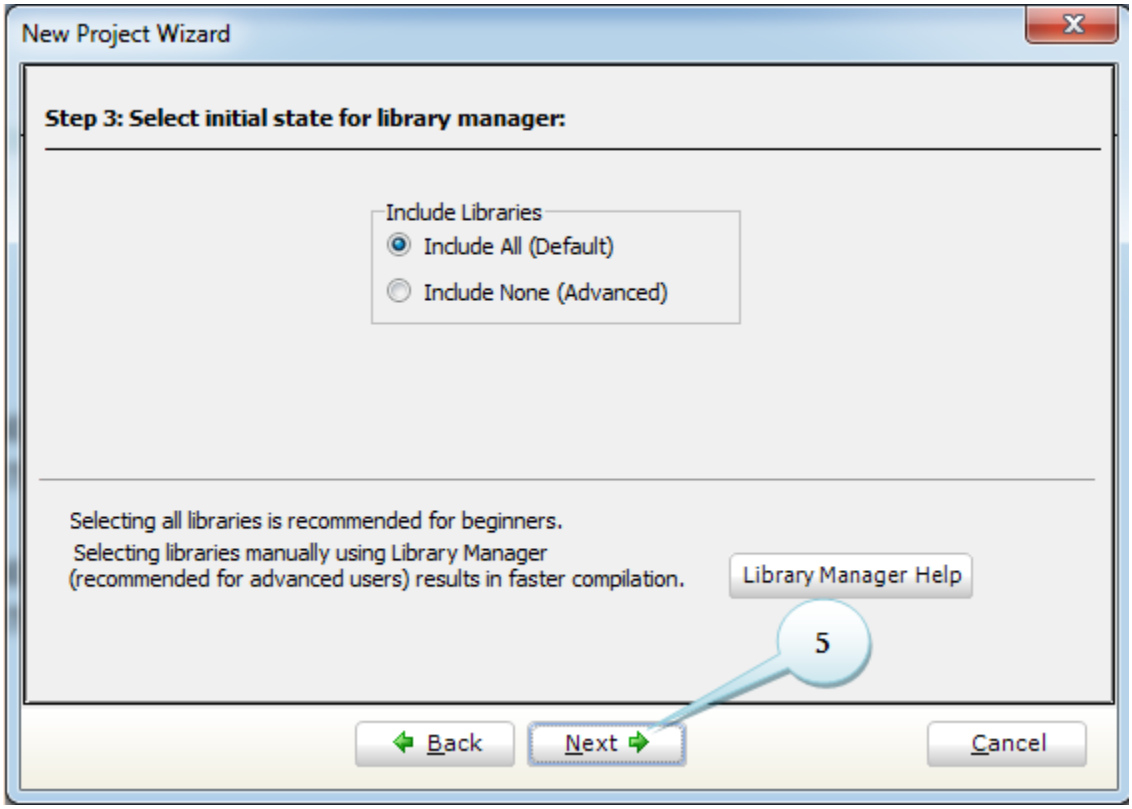
إنشاء مشروع جديد

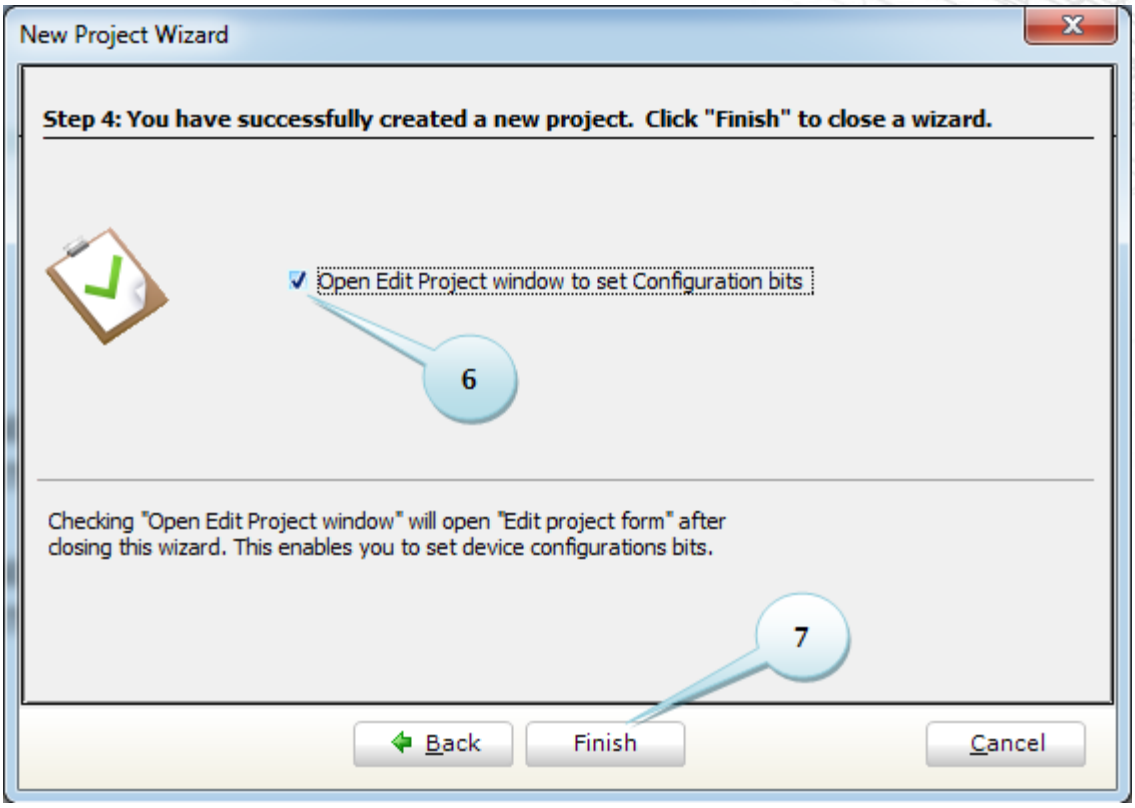
من قائمة Project نختار New Project، فيظهر مساعد المشروع الجديد New Project Wizard، نتبع الخطوات معه كما في الصور التالية:

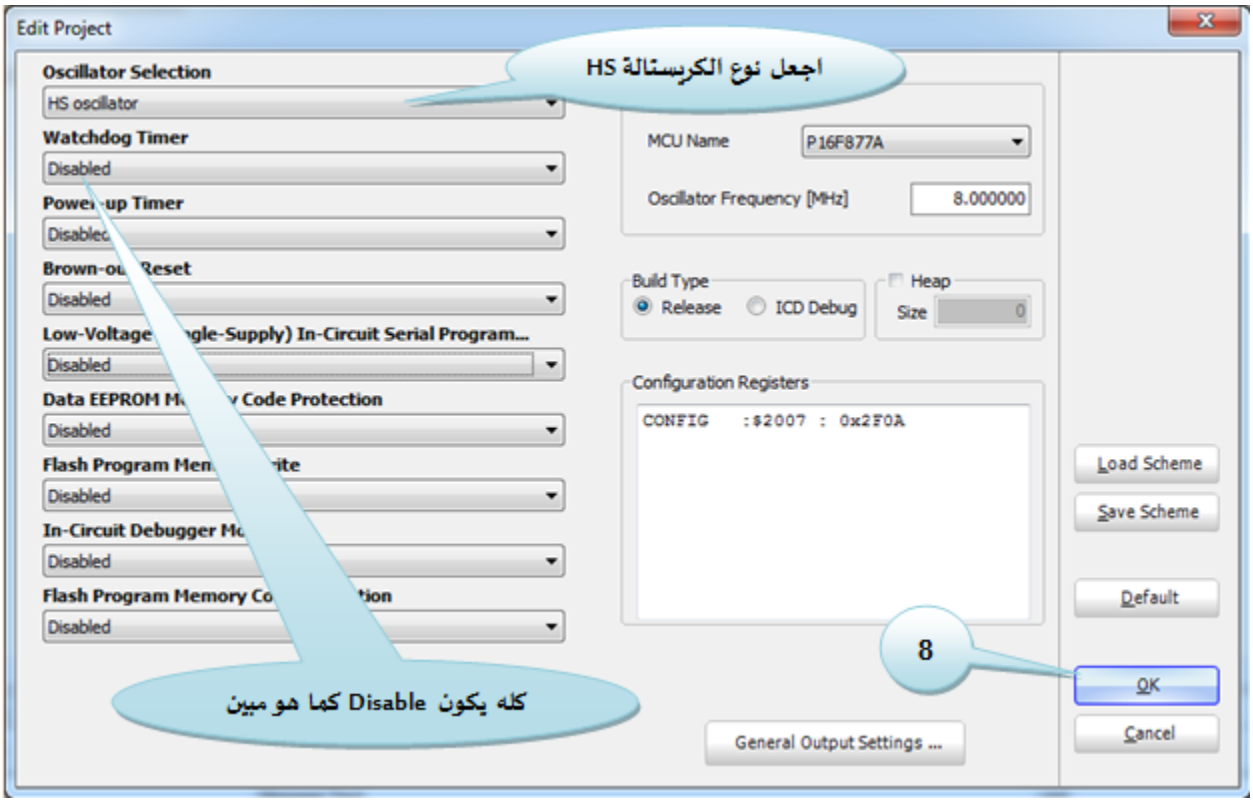

New Project Wizard
Welcome to the New Project Wizard
This wizard helps you:
• Create a new project
• Select the device for your project
• Setup device clock
• Add project files
Click Next to continue
2
Back
Next
Cancel

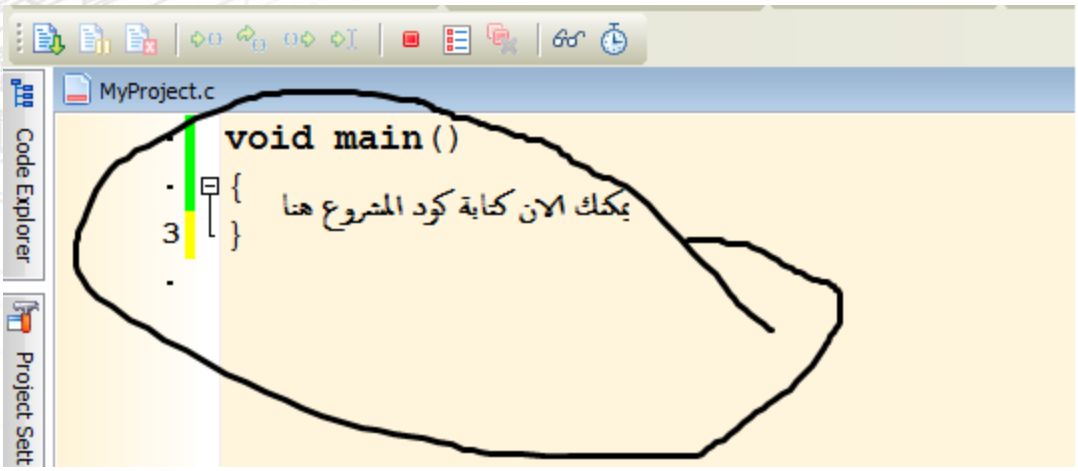

New Project Wizard
Step 1: Project Settings:
اسم المشروع
Project Name: MyProject
Project folder: C:\Users\Engineer2011\Desktop\New folder\
Browse
Device Name: P16F877A
Device Clock: 8.000000 MHz
المسار اذى سيتم حفظ المشروع فيه
لاختيار نوع الميكرو
لتعديل قيمة الكريستالة
Note: Project name and project folder must not be left empty.
3
Back
Next
Cancel


New Project Wizard
Step 2: Select files you want to add to project.
Add File To Project:
Add
Remove
Remove All
File Name
4
Back
Next
Cancel

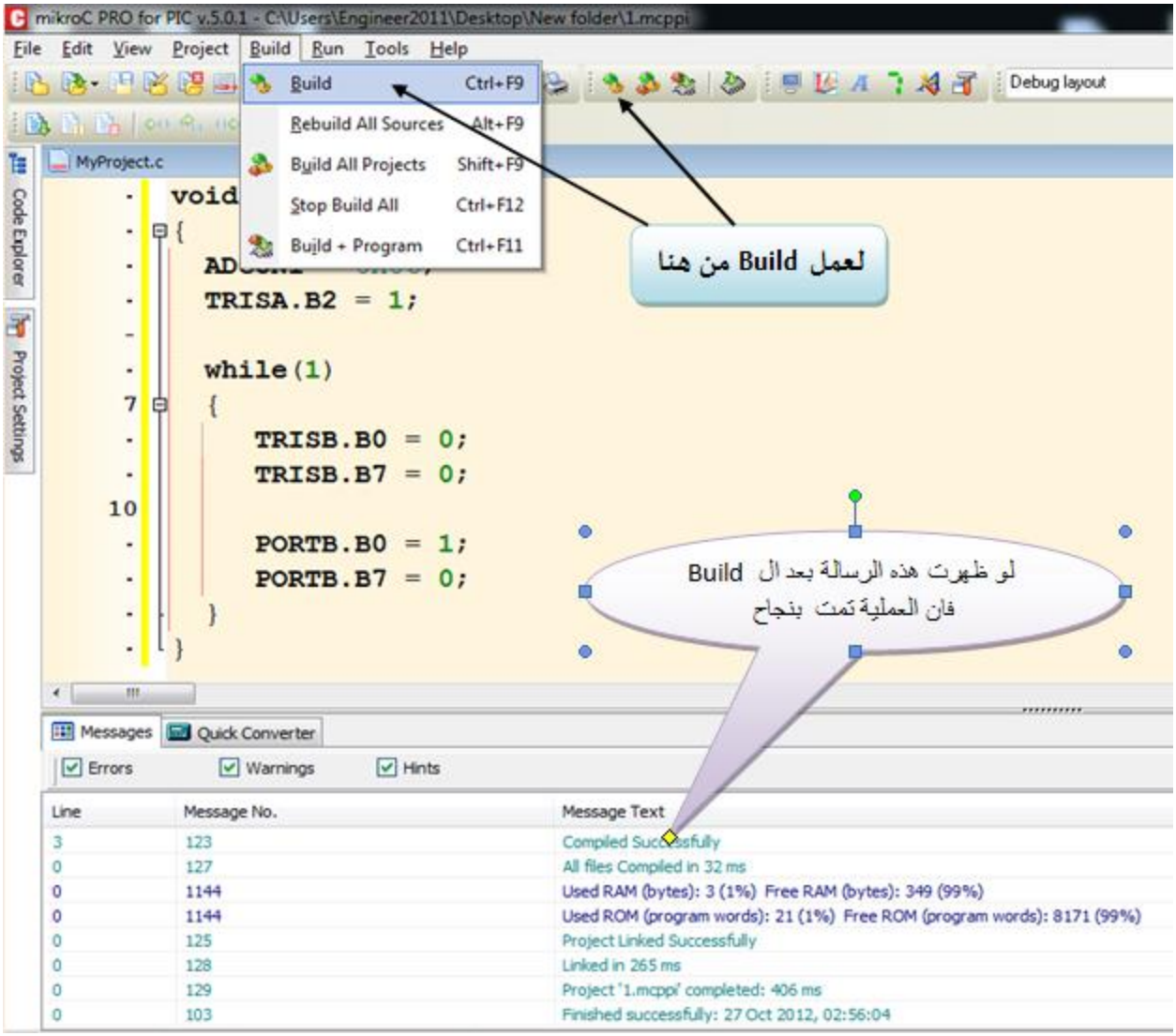

New Project Wizard
Step 3: Select initial state for library manager:
Include Libraries
Include All (Default)
Include None (Advanced)
Selecting all libraries is recommended for beginners.
Selecting libraries manually using Library Manager
(recommended for advanced users) results in faster compilation.
Library Manager Help
5
Back
Next
Cancel


New Project Wizard
Step 4: You have successfully created a new project. Click "Finish" to close a wizard.
Open Edit Project window to set Configuration bits
6
Checking "Open Edit Project window" will open "Edit project form" after closing this wizard. This enables you to set device configurations bits.
7
Back
Finish
Cancel

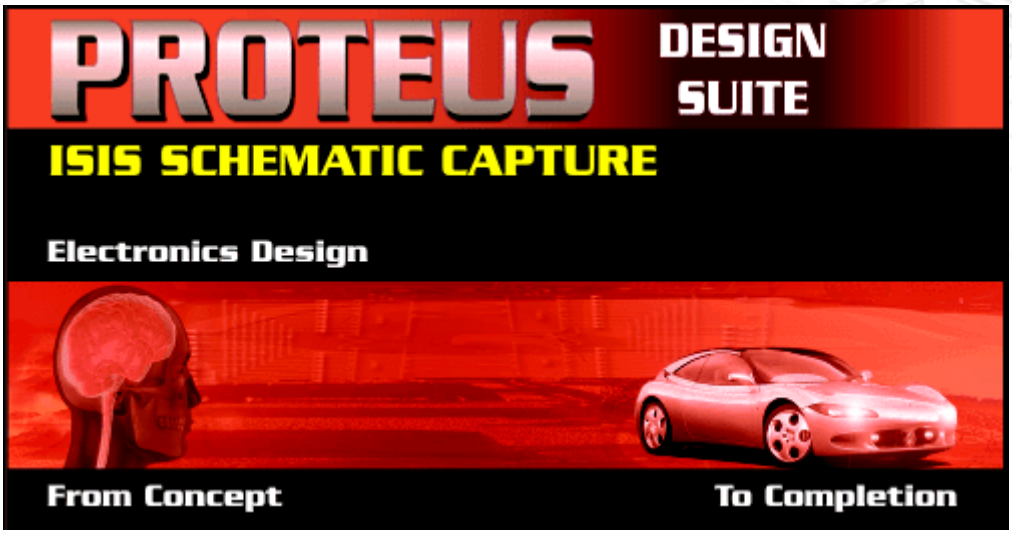

Edit Project
Oscillator Selection
HS oscillator
اجعل نوع الكريستالة HS
Watchdog Timer
Disabled
Power-up Timer
Brown-out Reset
Low-Voltage (Single-Supply) In-Circuit Serial Program...
Data EEPROM Memory Code Protection
Flash Program Memory Write
In-Circuit Debugger Mode
Flash Program Memory Code Protection
كله يكون Disable كما هو مبين
MCU Name
P16F877A
Oscillator Frequency [MHz]
8.000000
Build Type
Release
ICD Debug
Heap
Size
Configuration Registers
CONFIG :$2007 : 0x2F0A
Load Scheme
Save Scheme
Default
8
OK
Cancel
General Output Settings ...



وبعد الانتهاء من كتابة الكود يتم عمل build للبرنامج والتأكد من خلوه من الأخطاء:



برنامج البروتس



بر نامج Proteus Professional من تصـــميـم وانـتـاج شـــركـة Labcenter وموقعها على الانترنت:

www.labcenter.com

يحتوي هذا البرنامج على برنامجين أحدهما يسمى ISIS والمستخدم لمحاكاة الدوائر الإلكترونية وهوا المطلوب والآخر يسمي ARES ويستخدم لتصميم الدوائر المطبوعة PCB.

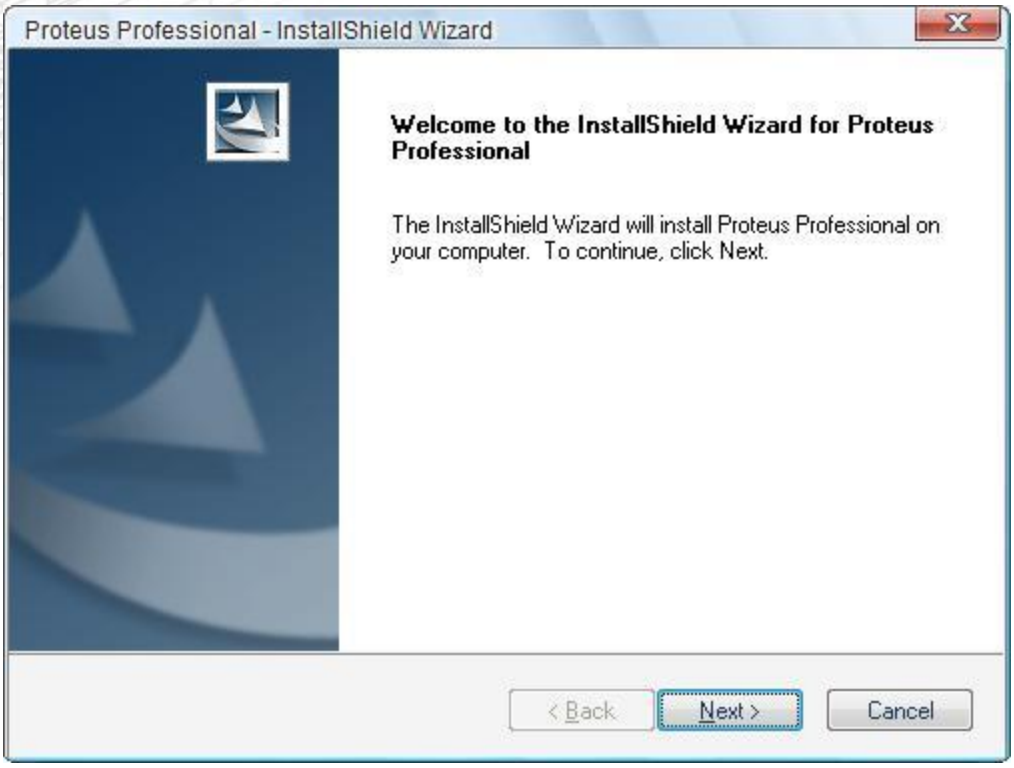
ولتحميل أحدث نسخة من البرنامج يمكن زيارة موقع الشركة والوصول إلى لينك التحميل أو مباشرة من موقع Softpedia على هذا اللينك:

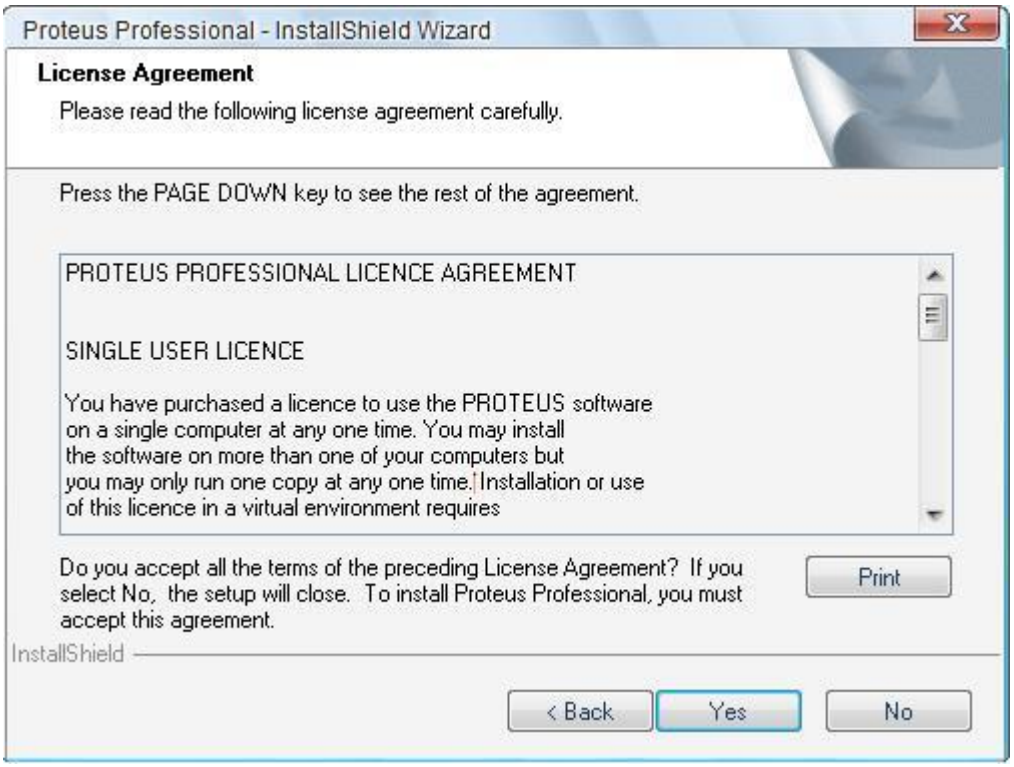
http://www.softpedia.com/get/Science-CAD/Proteus-PCB-Design.shtml

رقم الإصدار المستخدم في هذا الكتاب هو 7.7 SP2

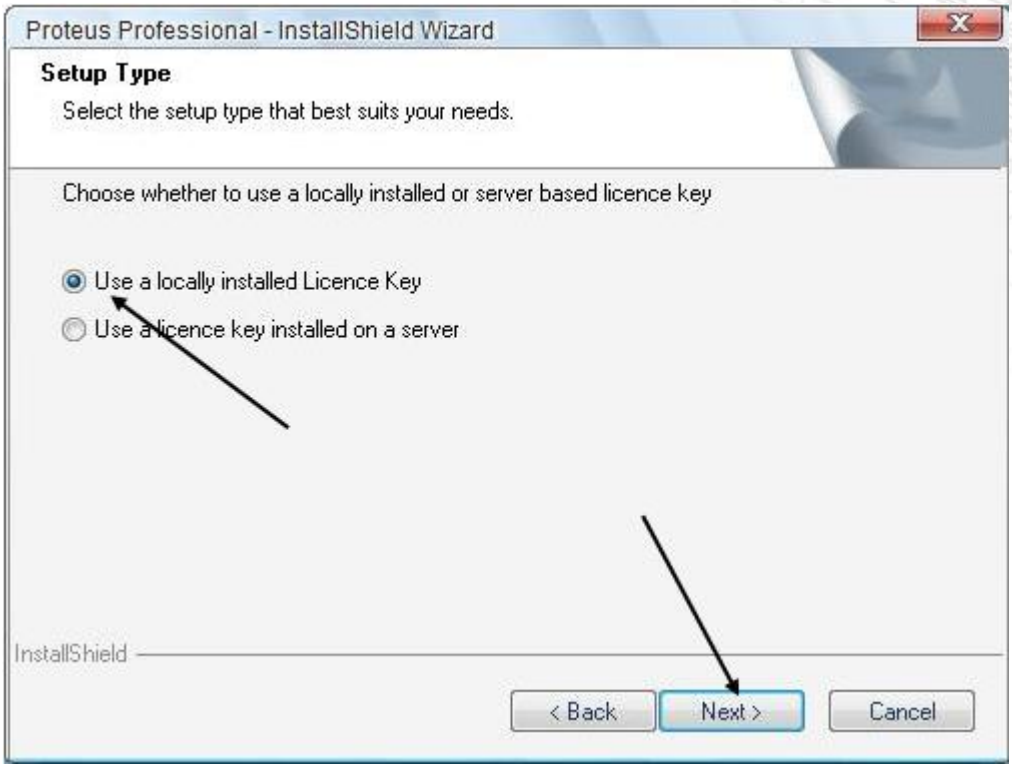
تسطيب البرنامج

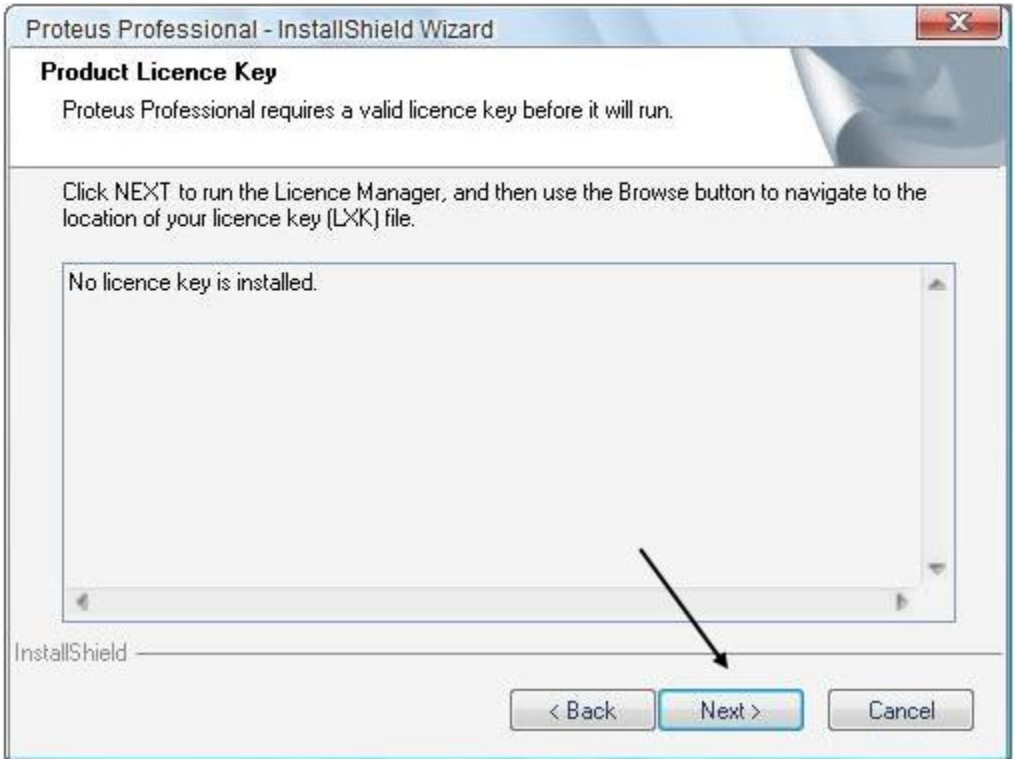
قم بفتح ملف البرنامج الذي تم تنزيله وكأي برنامج آخر اضغط Next:

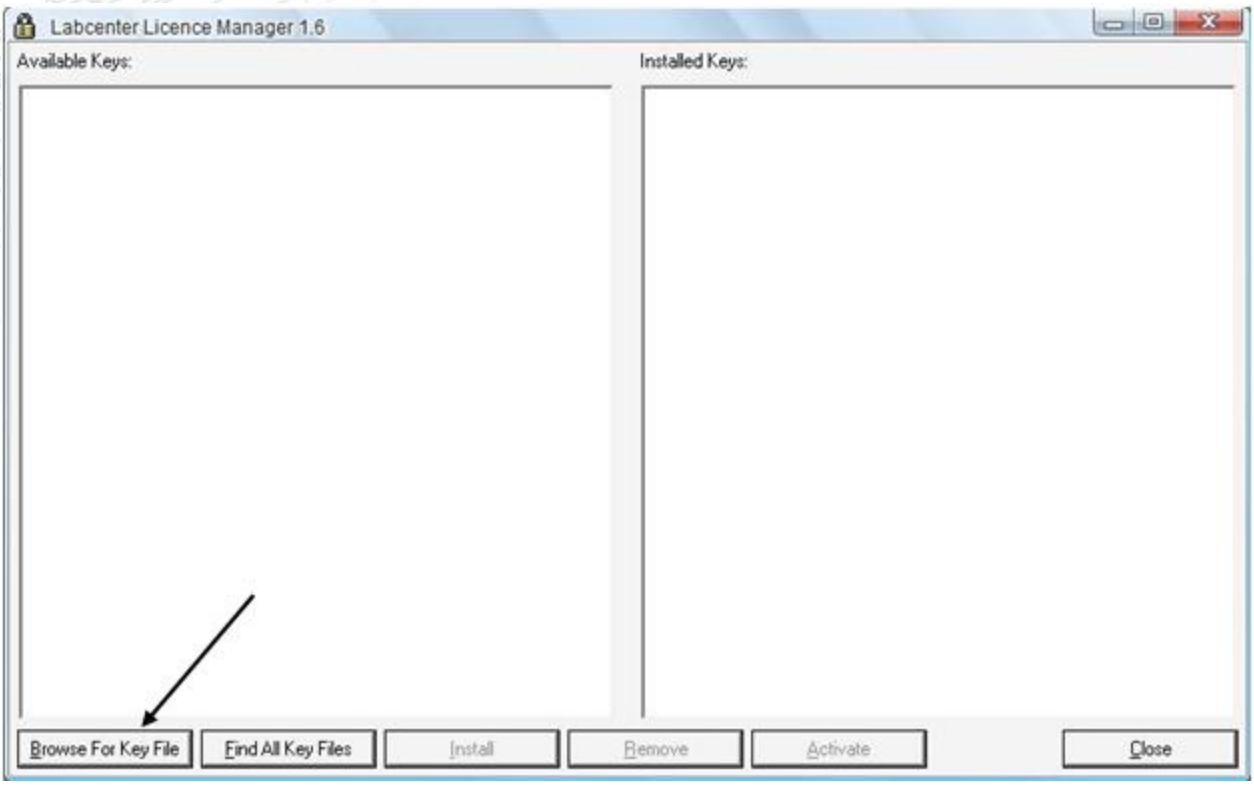




نوافق على اتفاقية ترخيص البرنامج بالضغط على Yes، فتظهر صفحة اختيار الرخصة license

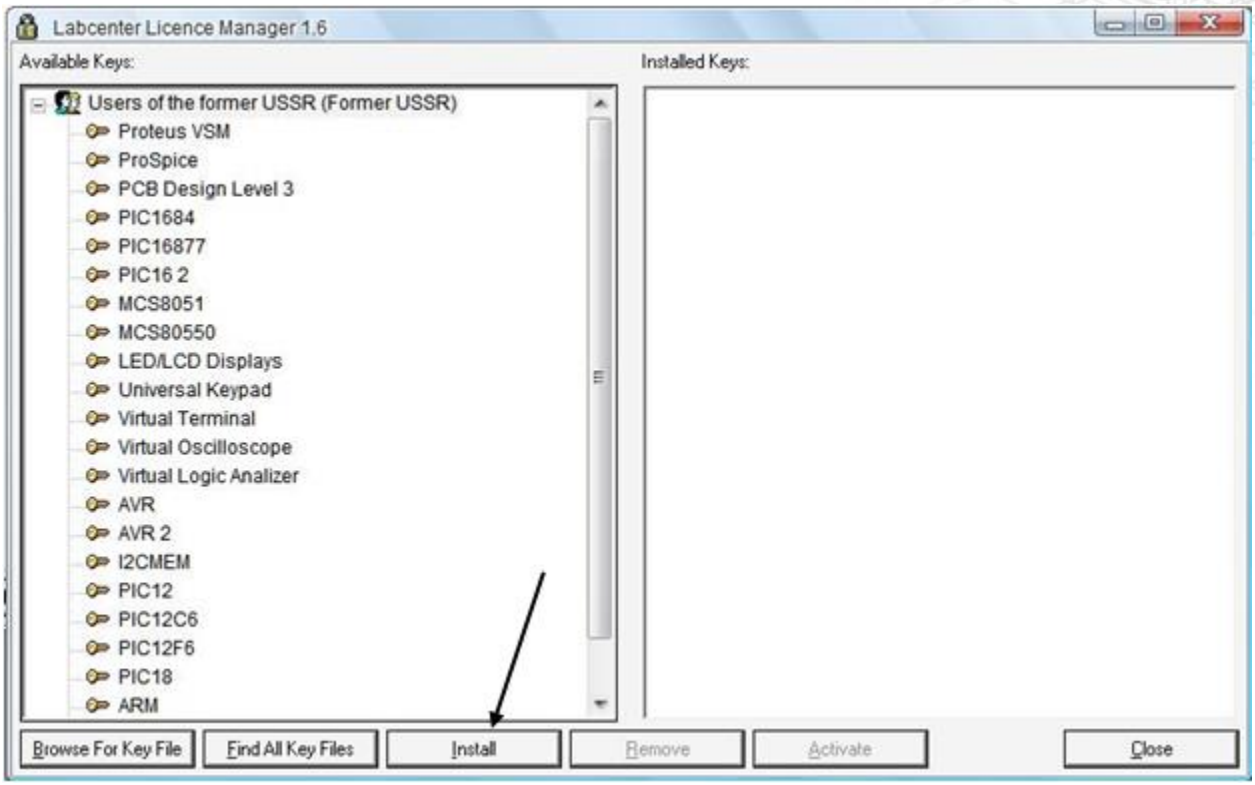

Proteus Professional - InstallShield Wizard
Setup Type
Select the setup type that best suits your needs.
Choose whether to use a locally installed or server based licence key
Use a locally installed Licence Key
Use a licence key installed on a server
InstallShield
< Back
Next >
Cancel

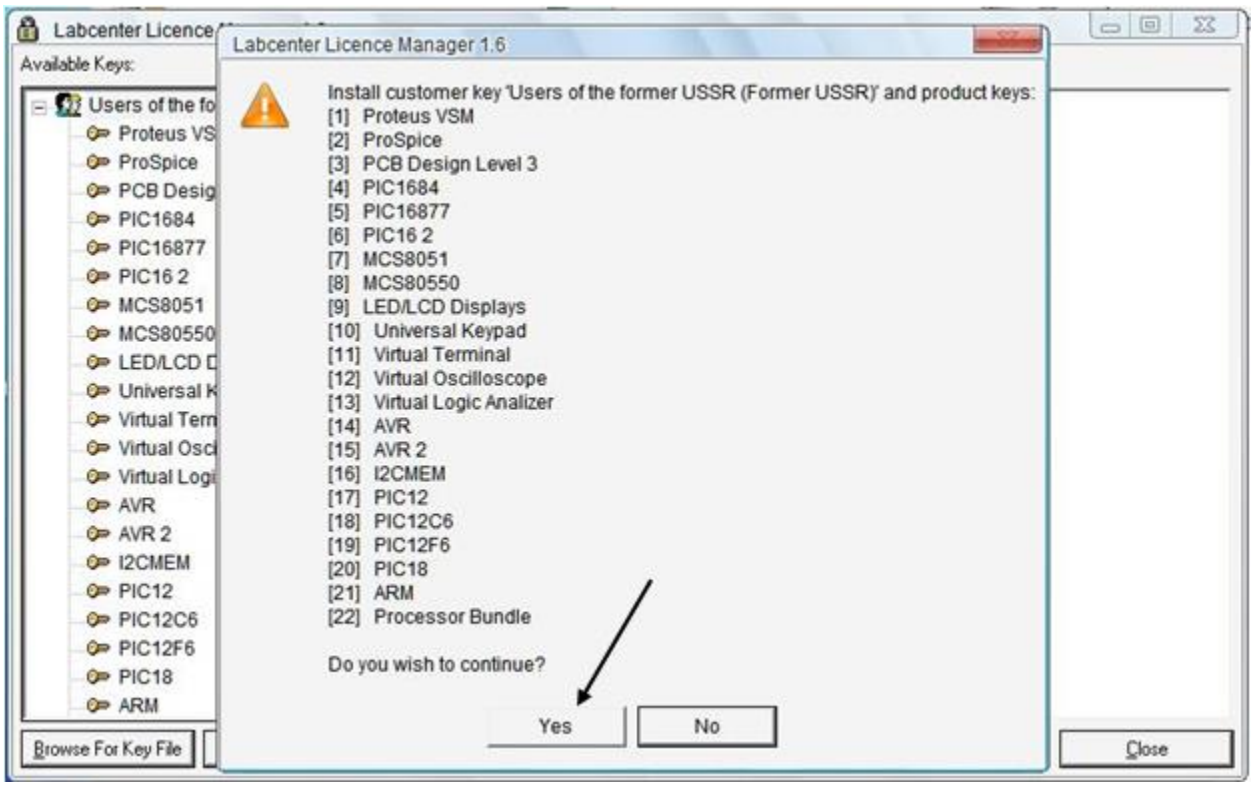

Proteus Professional - InstallShield Wizard
Product Licence Key
Proteus Professional requires a valid licence key before it will run.
Click NEXT to run the Licence Manager, and then use the Browse button to navigate to the location of your licence key (LXK) file.
No licence key is installed.
InstallShield
< Back
Next >
Cancel

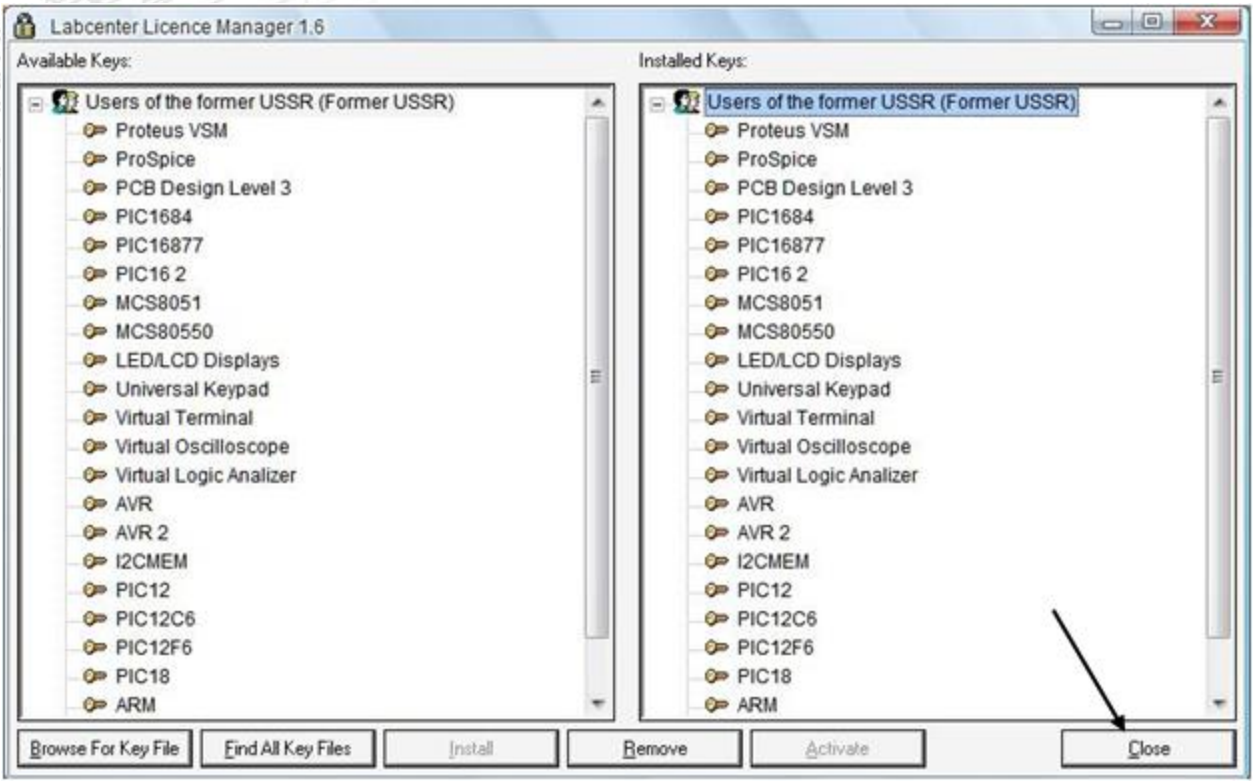


نضغط لإيجاد ملف الرخصة المخزنة على الكومبيوتر

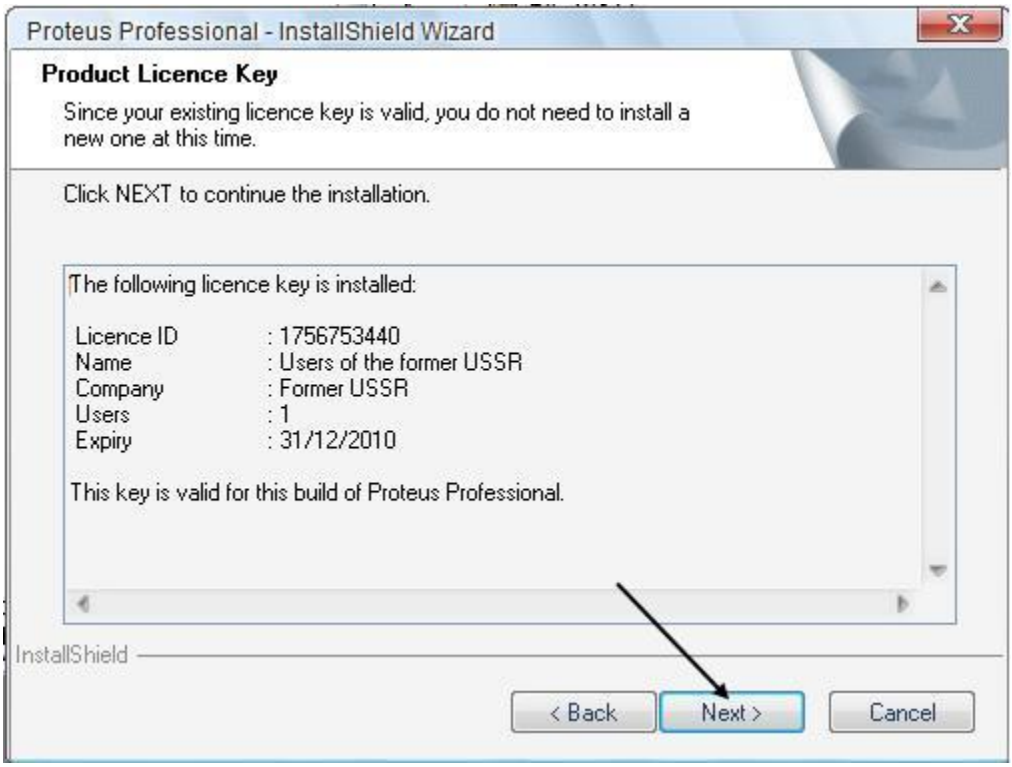
يظهر على اليسار ما تحتويه الرخصة من دعم للمنتجات داخل البرنامج فنضغط install





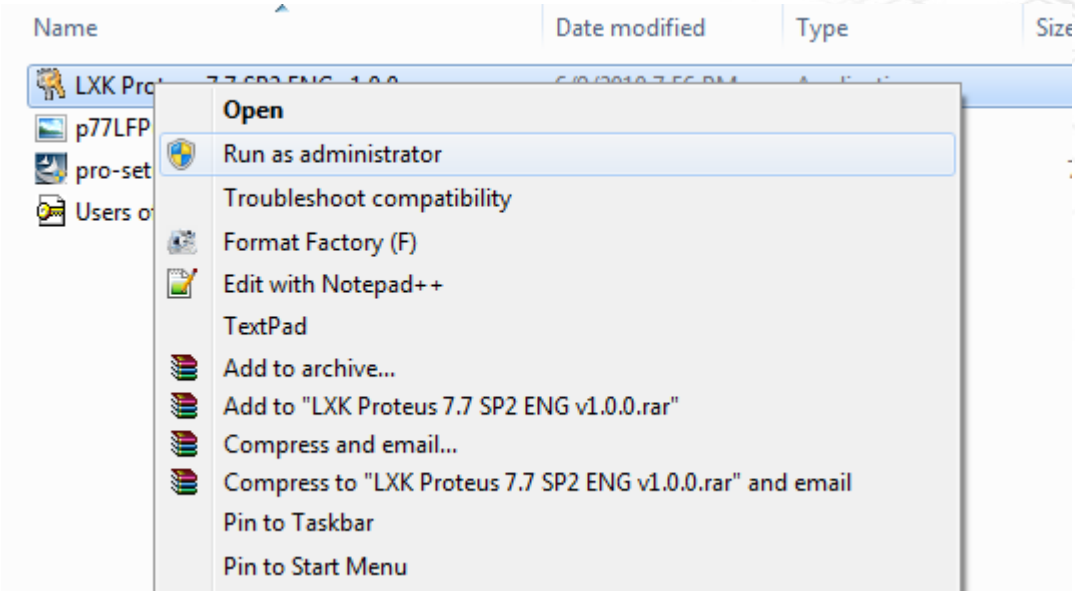


نغلق هذه النافذة بعد تنزيل الرخصة لاستكمال إجراءات التنصيب

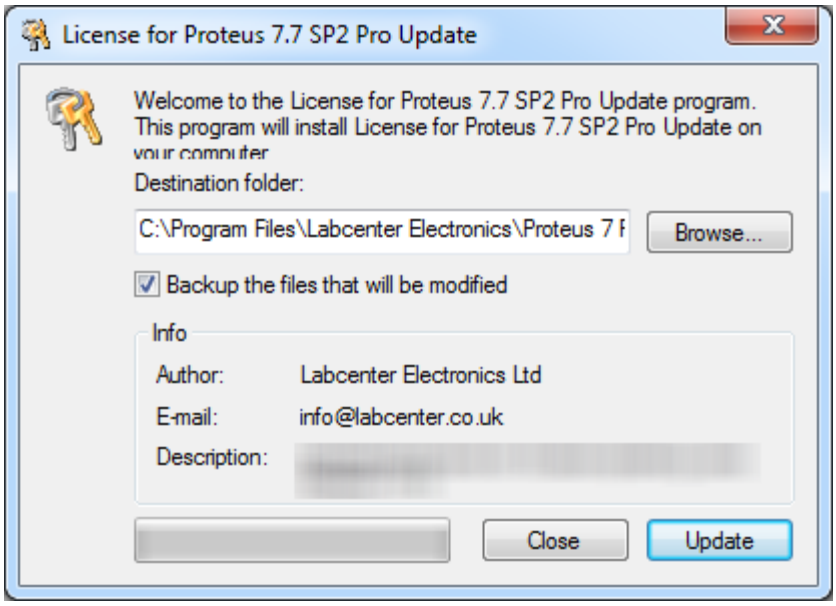


ثم نكمل باقي الخطوات بالضغط على Next حتى الانتهاء من التنصيب

استكمال تنزيل الرخصة:

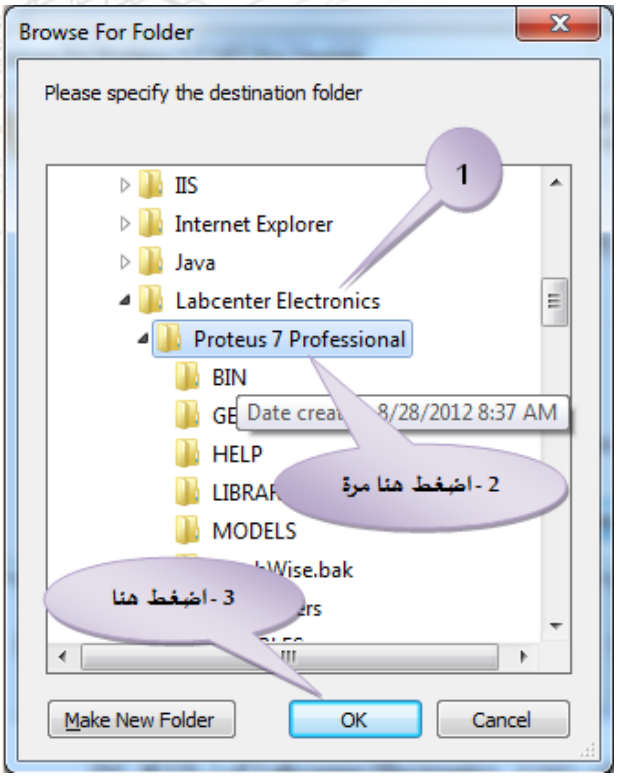


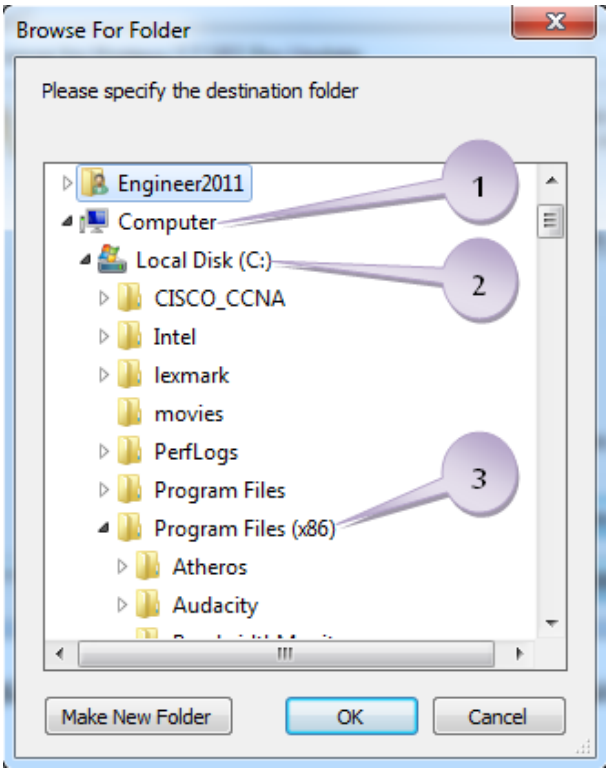
لابد أن تكون مستخدم Administrator على الجهاز حتى يمكن استكمال التنصيب حيث تظهر نافذة قاتمة يتم الضغط فيها على yes فتظهر نا فذة جديدة كما بالشكل:

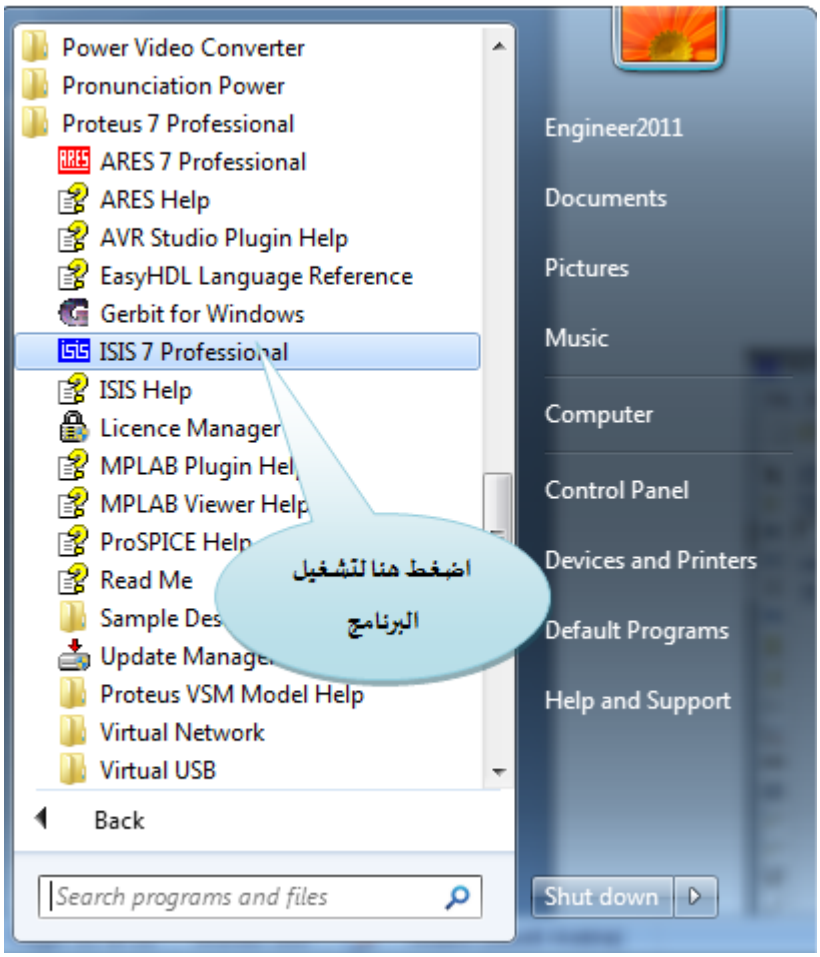


اضغط على Browse في النافذة المجاورة لتظهر نافذة جديدة تحدد منها مسار تنزيل البرنامج

في الصورتين التاليتين إن لم تجد المجلد Program Files (x86) فقم بالضغط على Cancel ثم تخطى هاتين الصورتين وأكمل الخطوات التالية





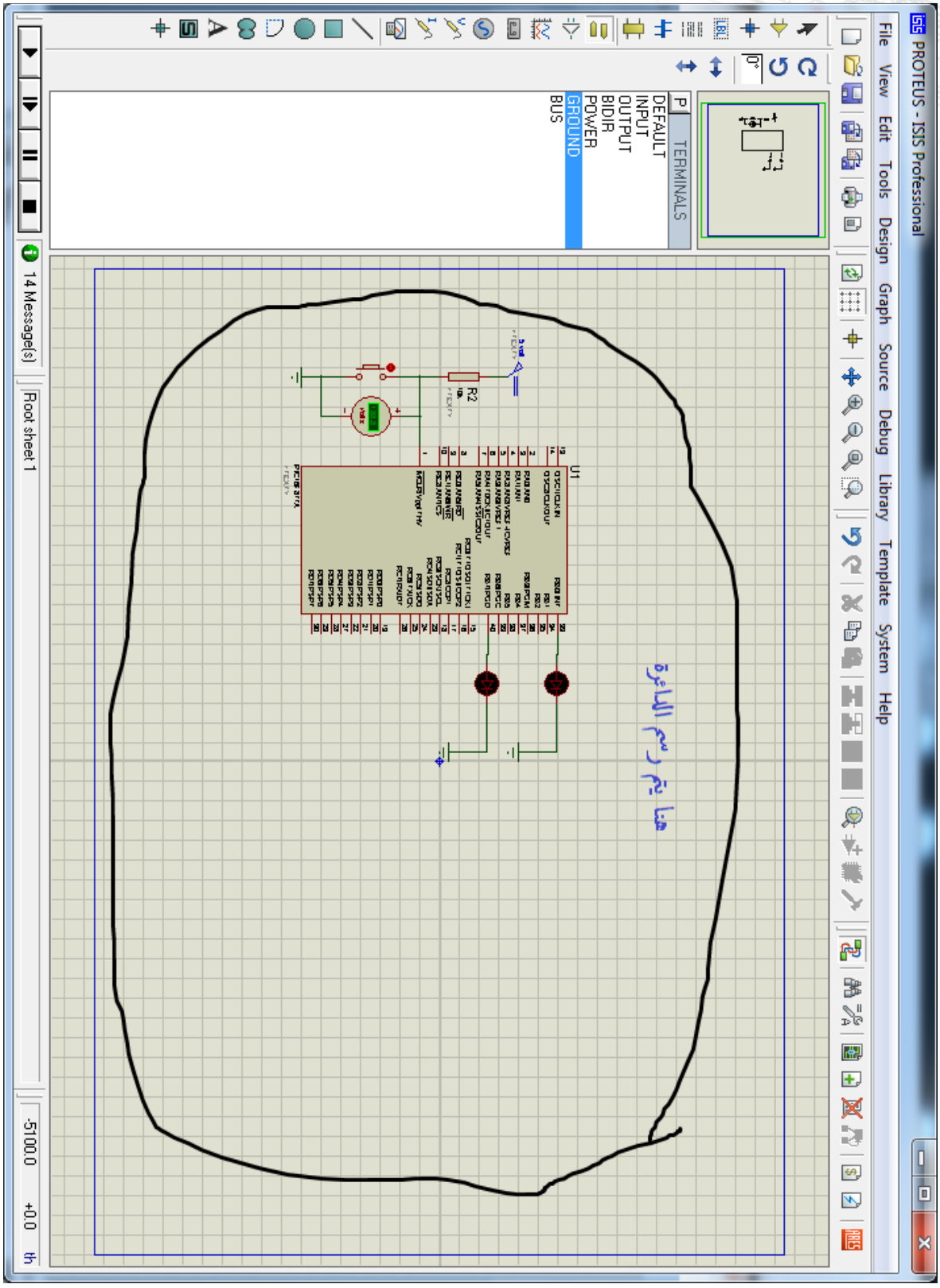


ثم اضغط Update ثم OK في النافذة التي ستظهر ثم Close وهنا تنتهي عملية تنصيب البرنامج وتنصيب الرخصة.

تشغيل برنامج المحاكاة:

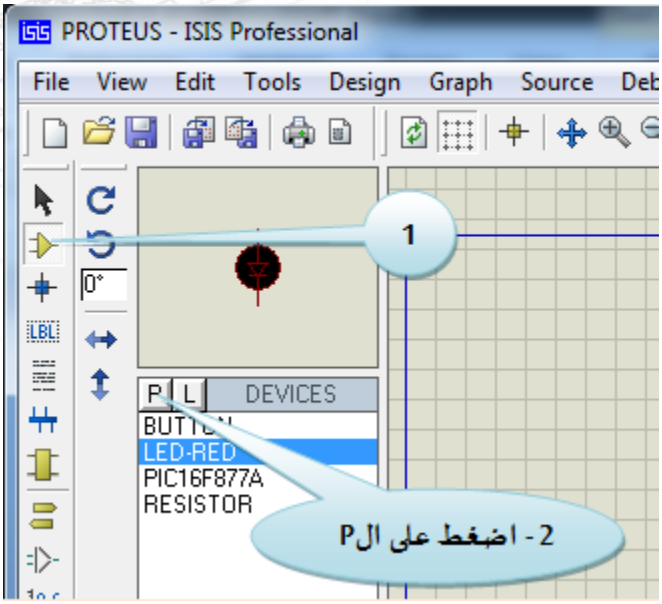
من قائمة Start اختار All programs ومنها اختار Proteus 7 Professional ثم ISIS 7 Professional كما بالشكل

واجهة البرنامج



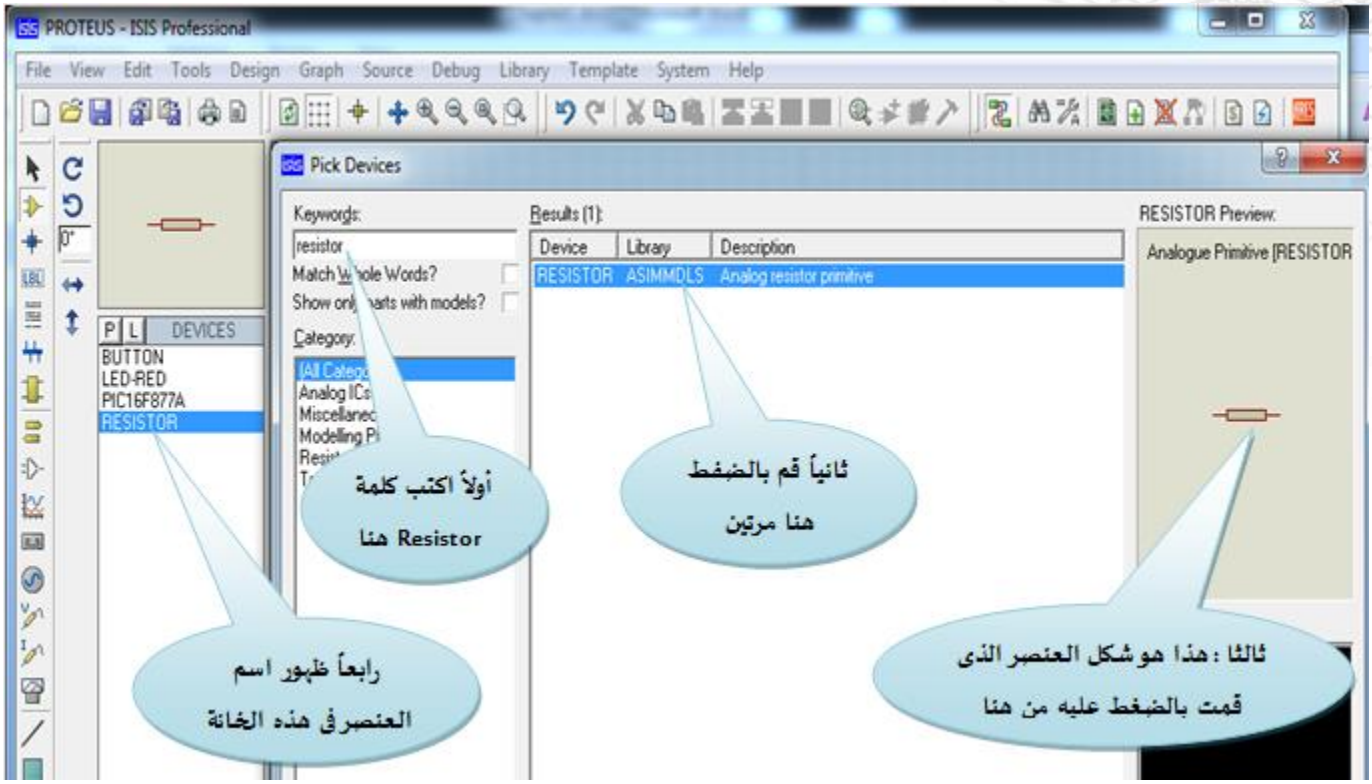
الحصول على المكونات الإلكترونية للدائرة

بداية يتم إضافة المكونات الإلكترونية المكونة للدائرة اللي نريد محاكاتها إلى المشروع ثم تجميعها معا لتكوين التصميم النهائي للدائرة.

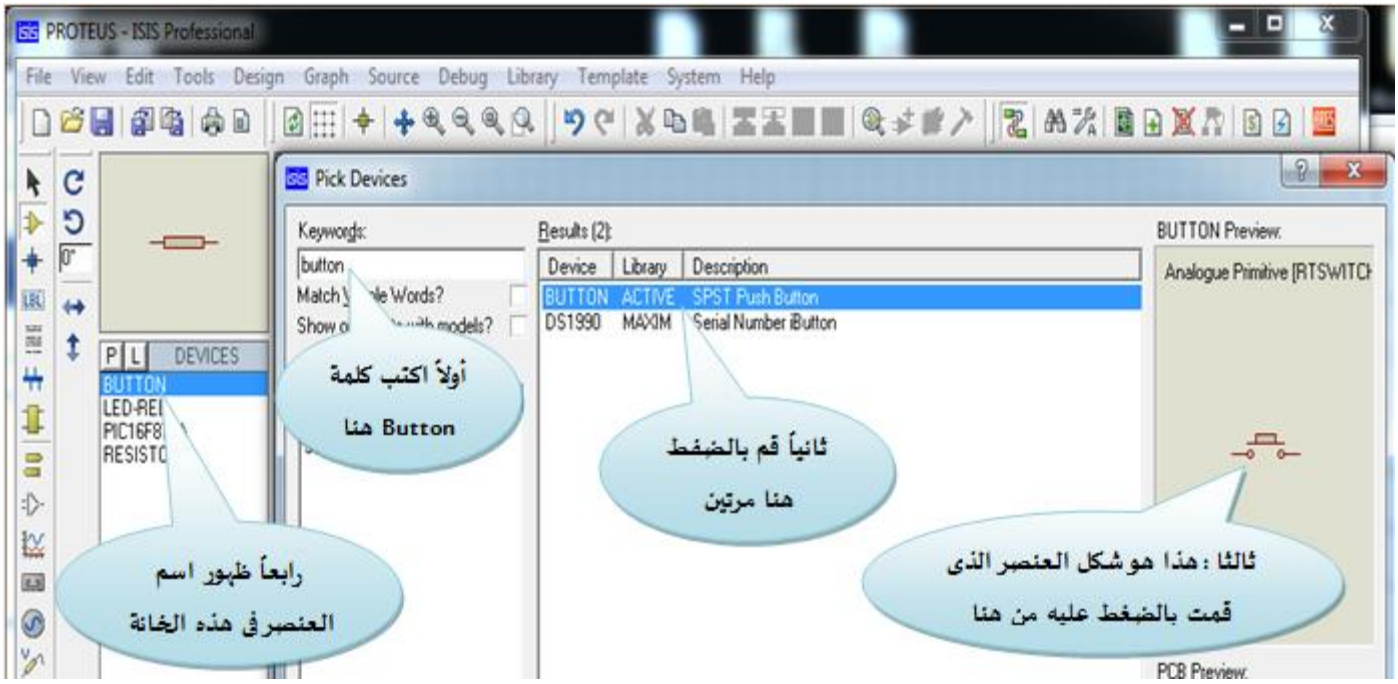


لإضافة هذه المكونات اتبع خطوات الصورة المجاورة، بعد الضغط على حرف P تظهر نافذة جديدة تحتوي على قائمة بكل المكونات الإلكترونية المتاحة في البرنامج وللحصول إلى المكون المطلوب من هذه القائمة الكبيرة نتبع ما في الشكل التالي:

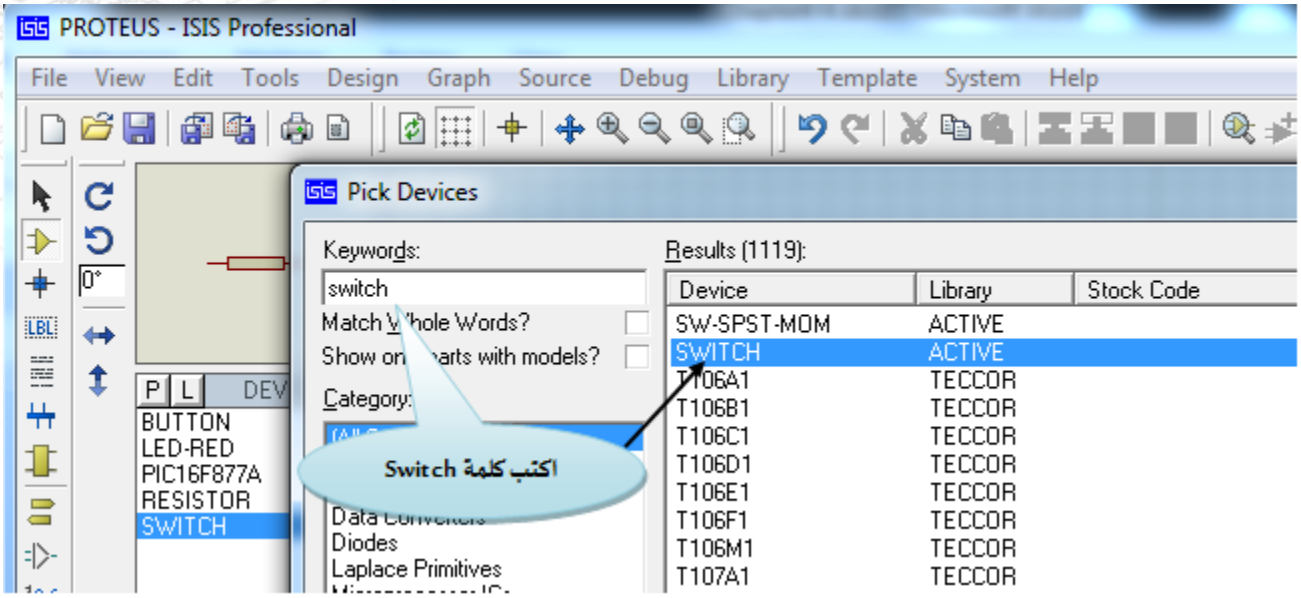
إضافة مقاومة



إضافة مفتاح من النوع Pushbutton:

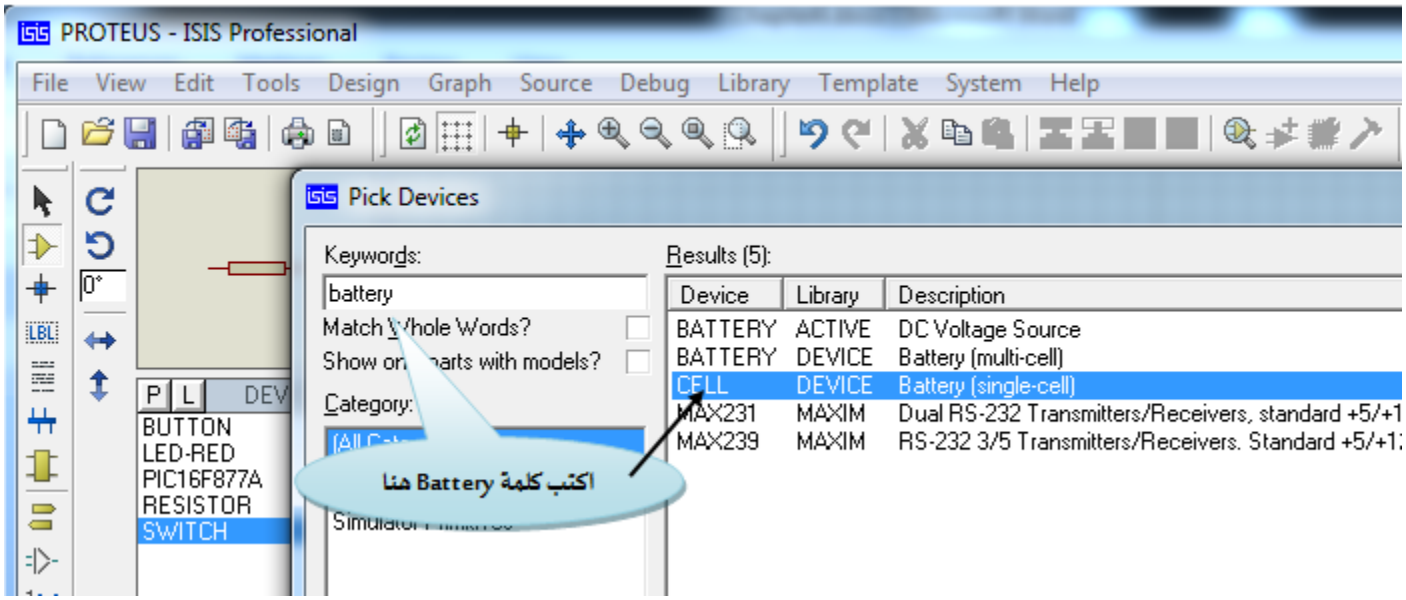


إضافة السويتش Switch

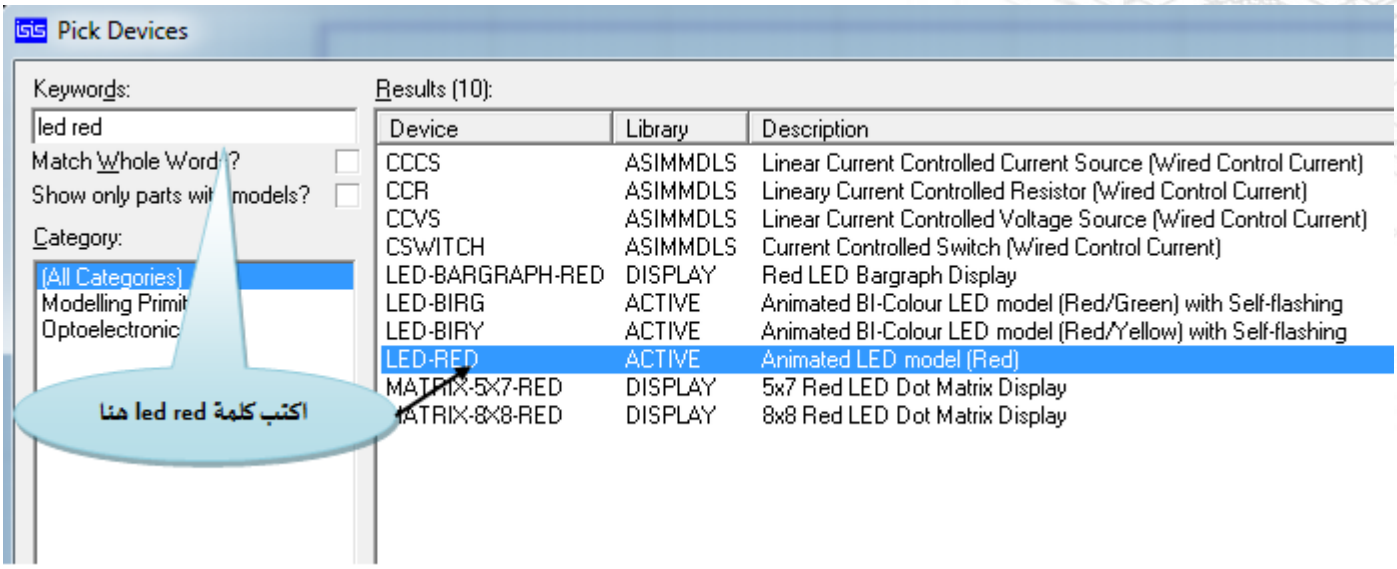


ملحوظة: الـ pushbutton عندما يتم الضغط عليه فانه يغلق طالما استمريت في الضغط عليه ويرجع لمكانه الأصلي بعد الضغط مباشرة أما السويتش عندما يتم الضغط عليه فانه يظل مضغوط ولا يرجع لمكانه الأصلي إلا بعد أن تضغط عليه مرة أخرى.

إضافة البطارية

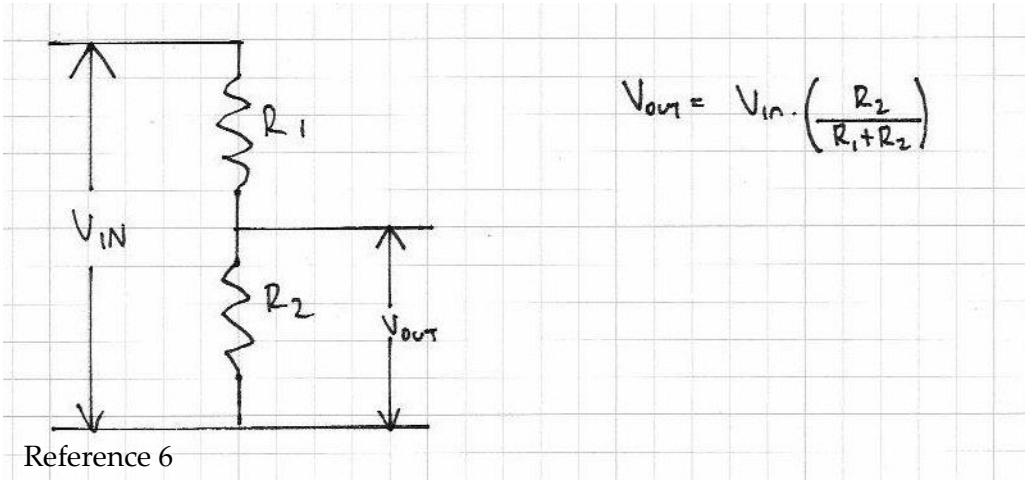


إضافة الليد



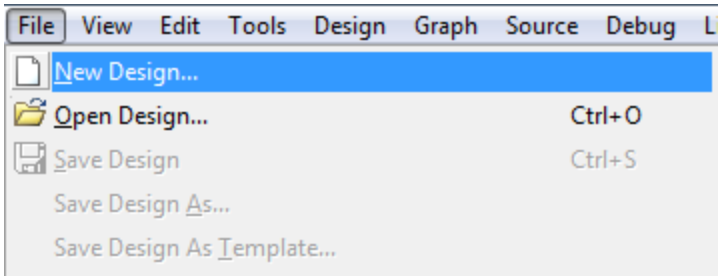
مثال

هنا سنقوم بعمل مشروع نشرح فيه كيفية عمل دائرة الـ Voltage divider وتشغيلها على بروتس وهذا هو تصميم الدائرة

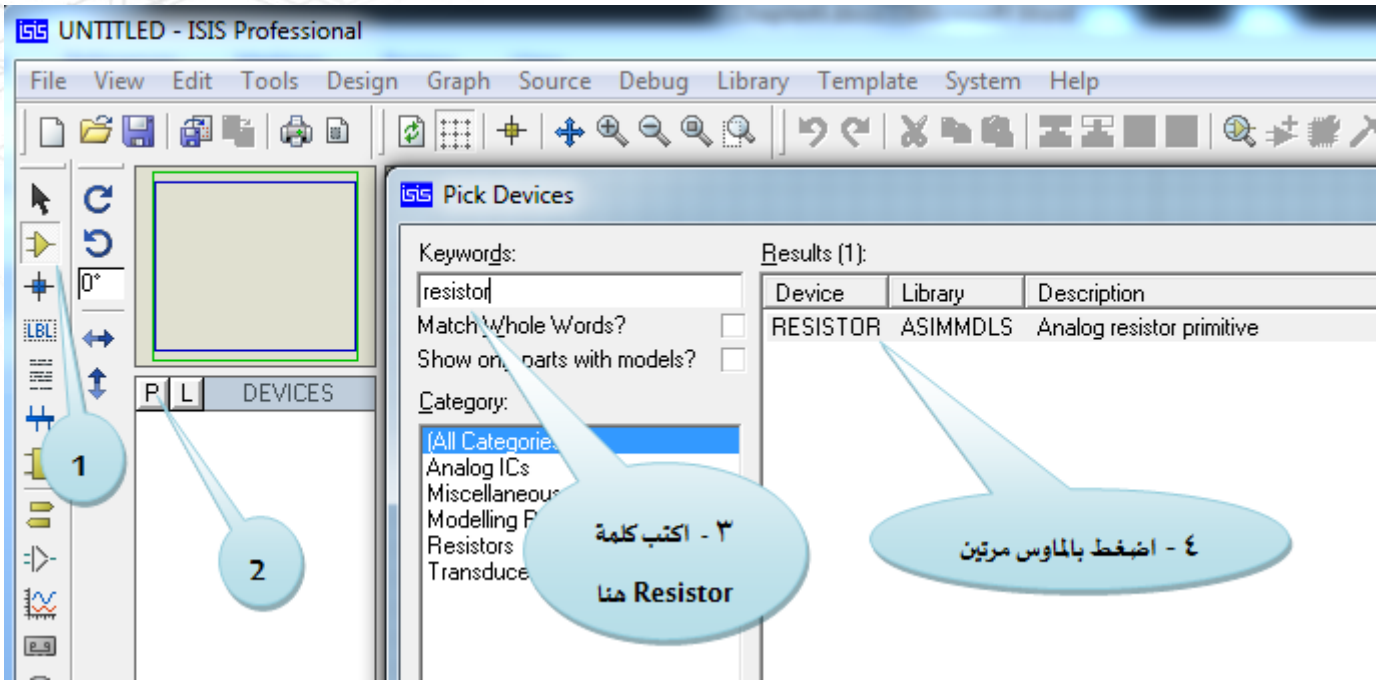


Reference 6

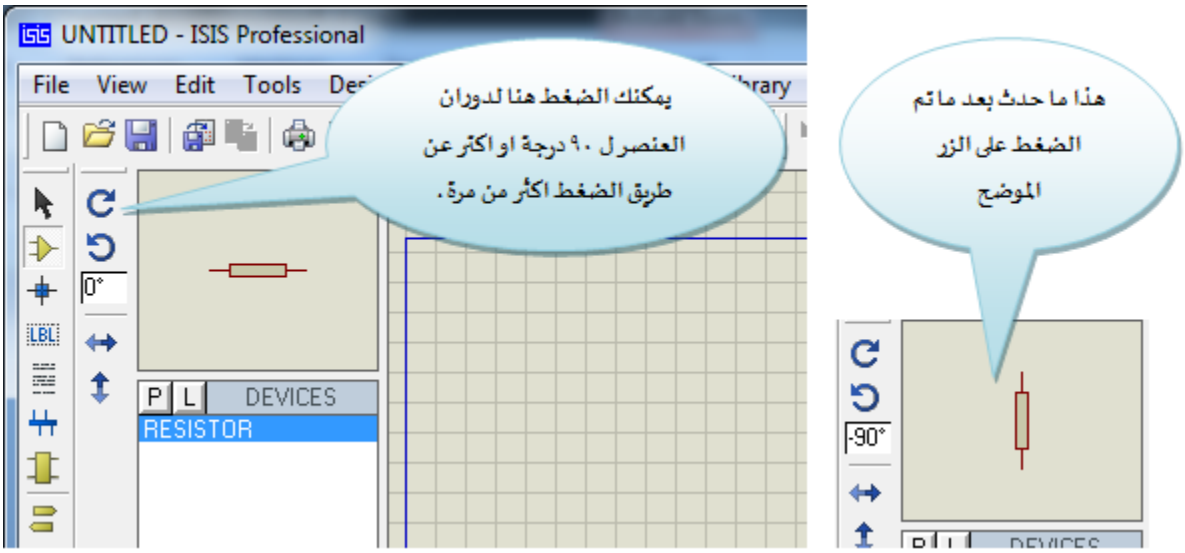
أولا قم بإنشاء مشروع جديد وذلك بالضغط على New design من قائمة File كما في الشكل:



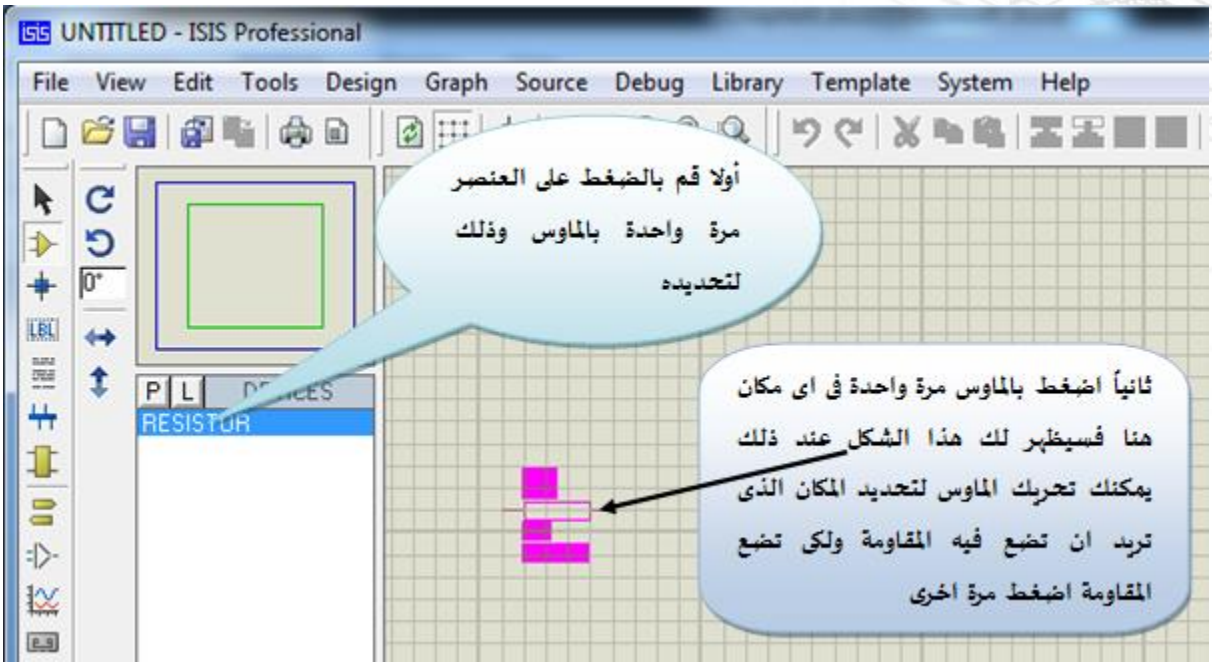
ثانيا: إضافة المكونات الإلكترونية المطلوبة: ننظر للتصميم المعطى وفيه تظهر المكونات المطلوبة لعمل هذا سواء كهاردوير أو للمحاكاة، فنقوم بإضافاتها باتباع الخطوات السابقة:



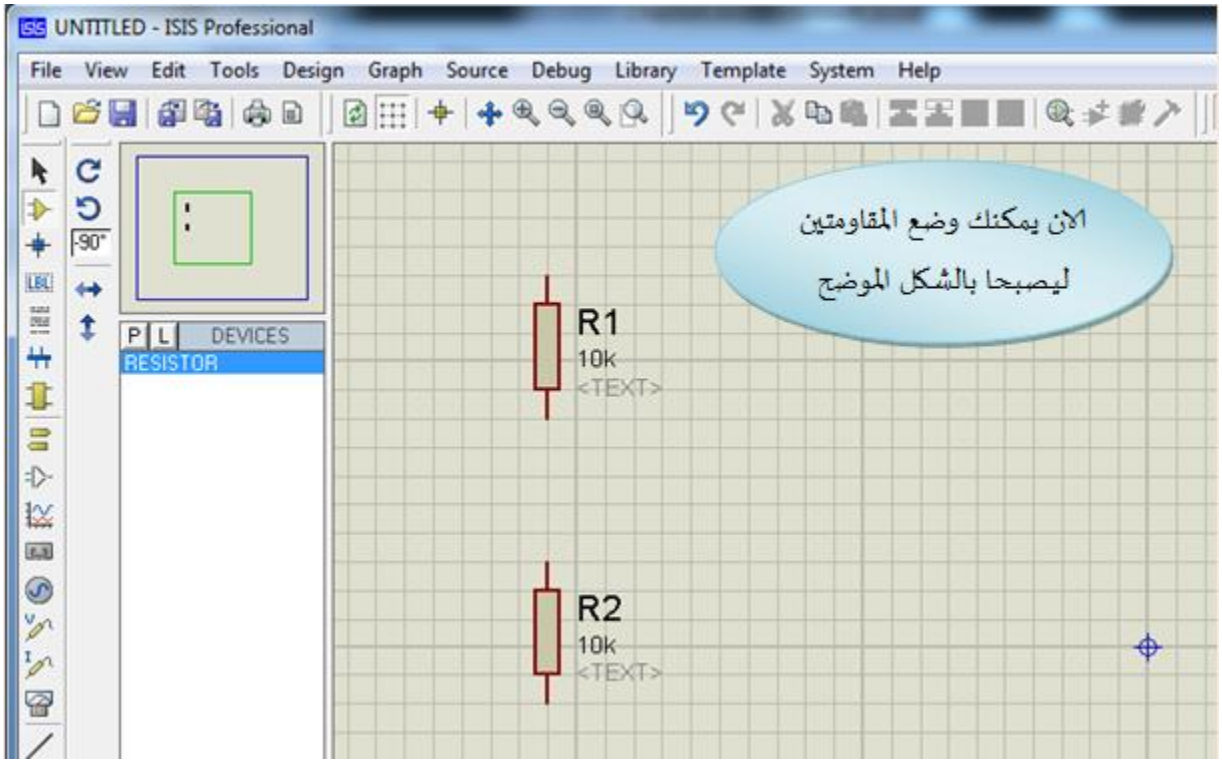
بعض المكونات تظهر افتراضيا بشكل رأسي وبعضها بشكل أفقي كما في الشكل التالي وقد نحتاج تعديل هذه الاتجاه تبعا لشكل المكون في التصميم وذلك لتوصيلها بأفضل طريقة في الدائرة، فيتم أخيار العنصر ثم استخدام أزرار التدوير لليمين أو لليسار الموجودة كما في الصورة التالية:



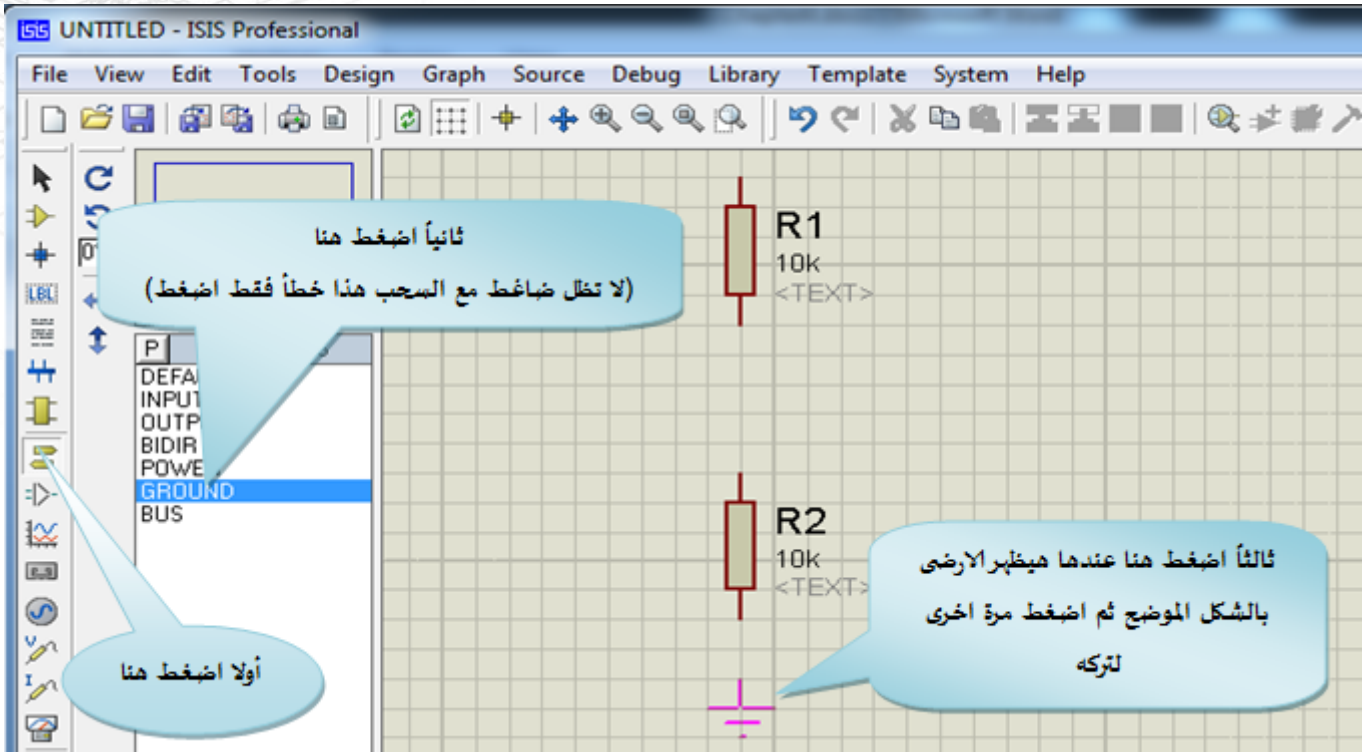
وضع العناصر الإلكترونية في المكان المخصص لها في التصميم:



وتتكرر نفس الخطوة لإضافة جميع المكونات الأخرى



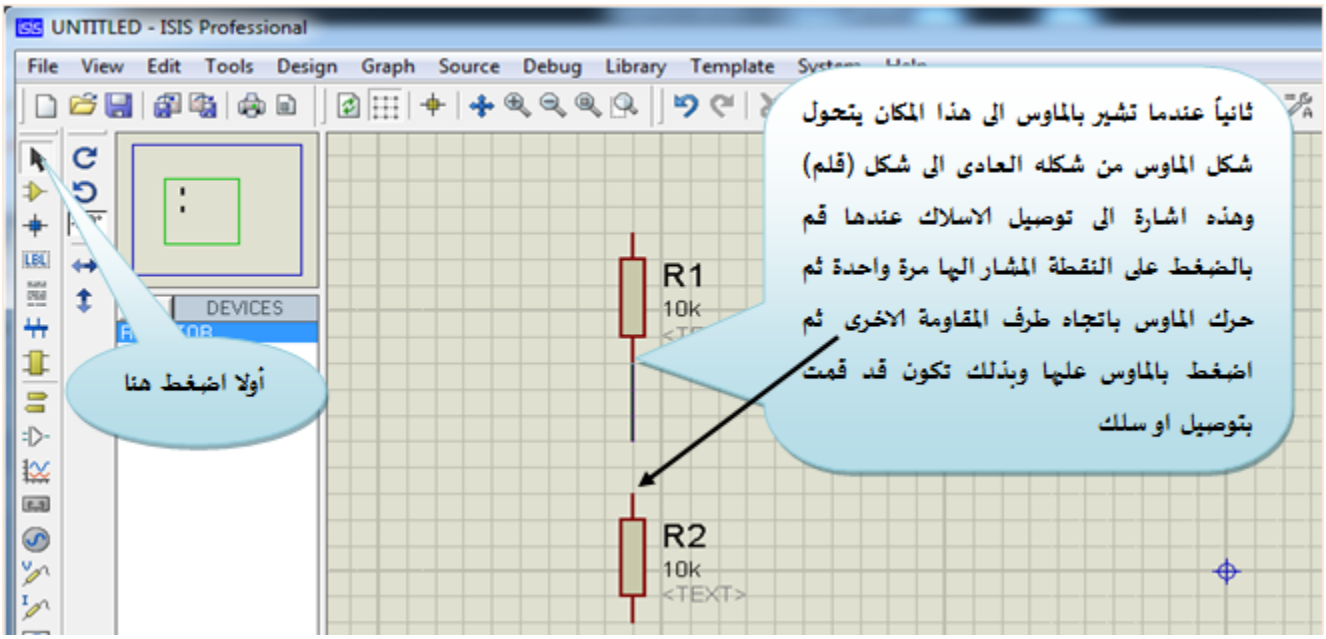
إضافة الأرضي (Ground)



سلطان

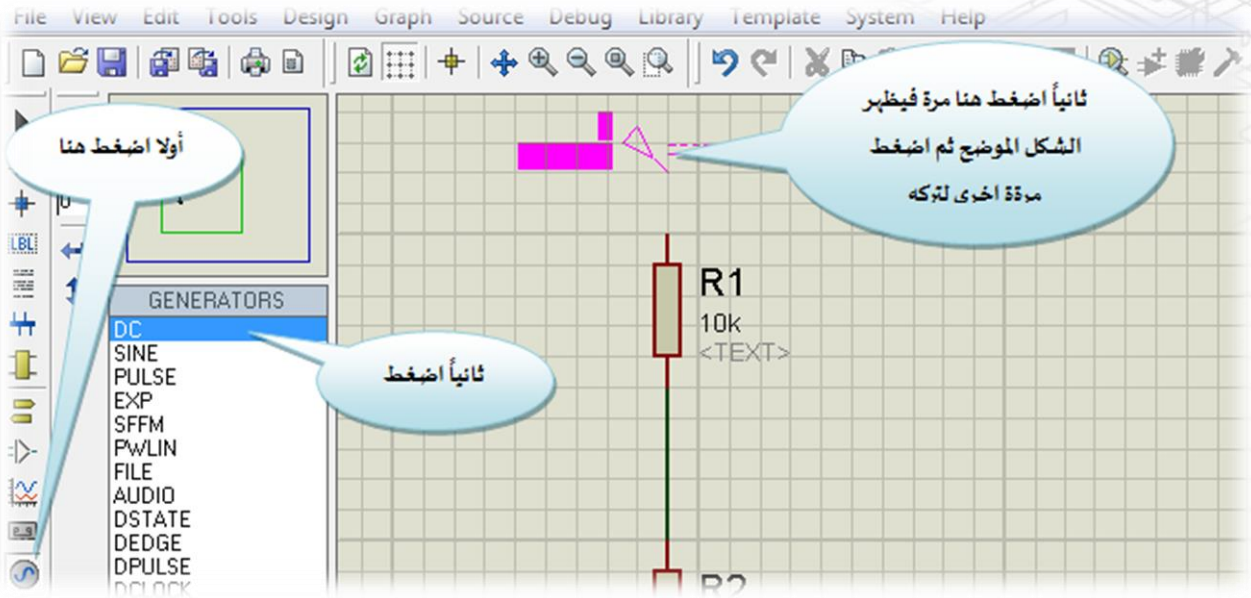
سلطان

توصيل المكونات

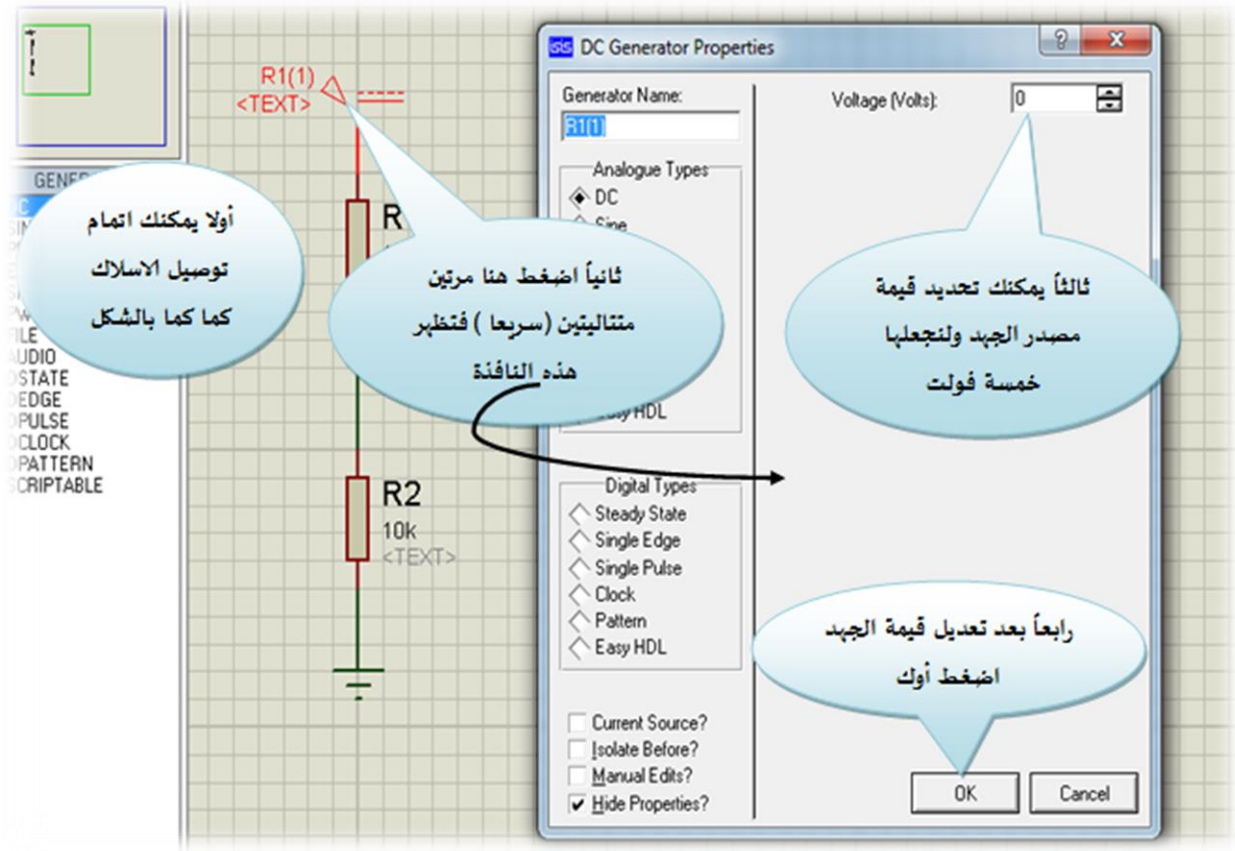


الحصول على جهد ثابت خمسة فولت:

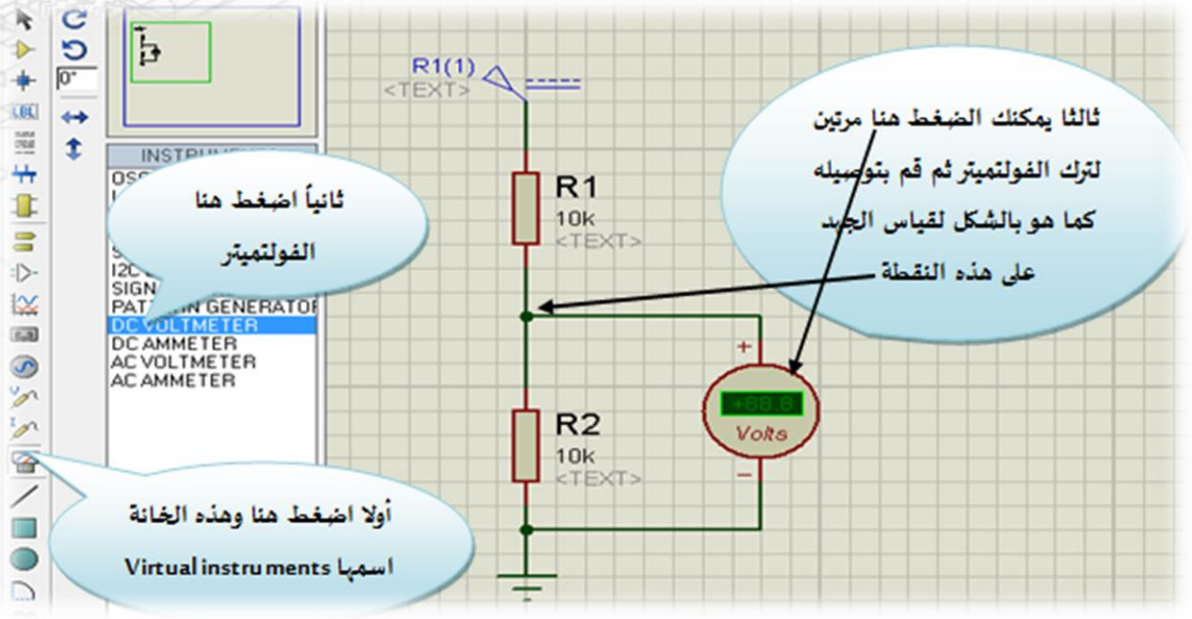
(تم شرحه سابقا كيفية الحصول على بطارية) لكن هذه طريقة أخرى وهى الأفضل:



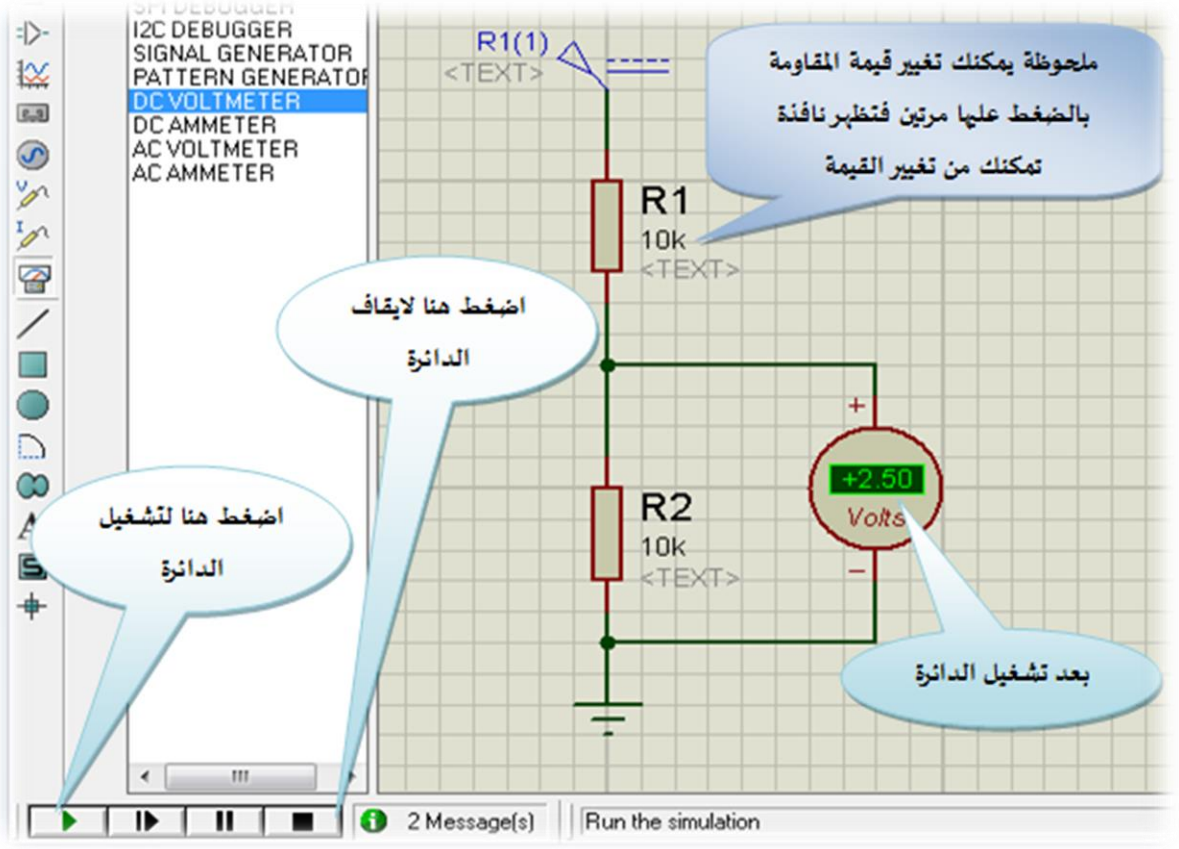
تحديد قيمة مصدر الجهد



إضافة جهاز قياس (كالفولتميتر):



تشغيل وإيقاف تشغيل المحاكاة



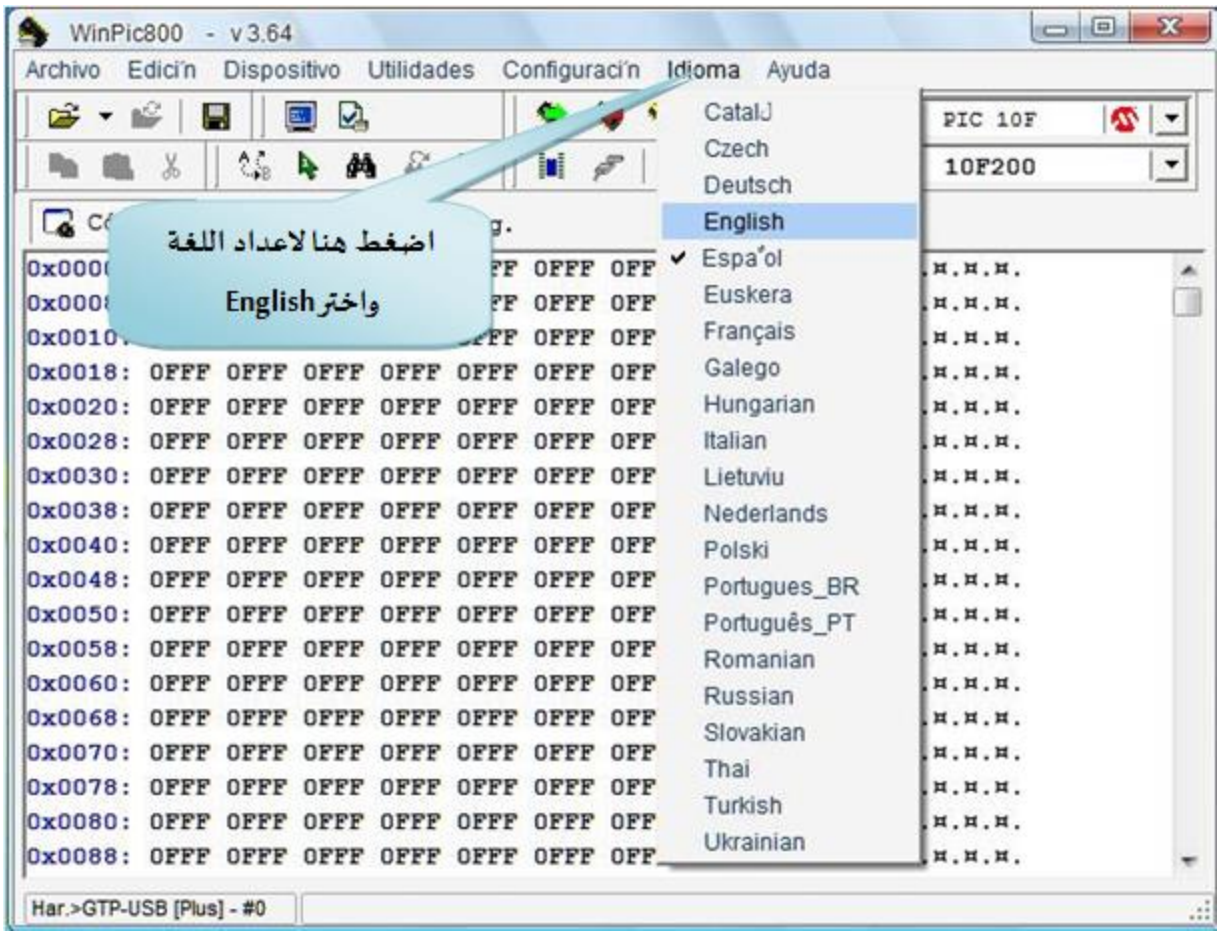
برنامج الحرق Winpic

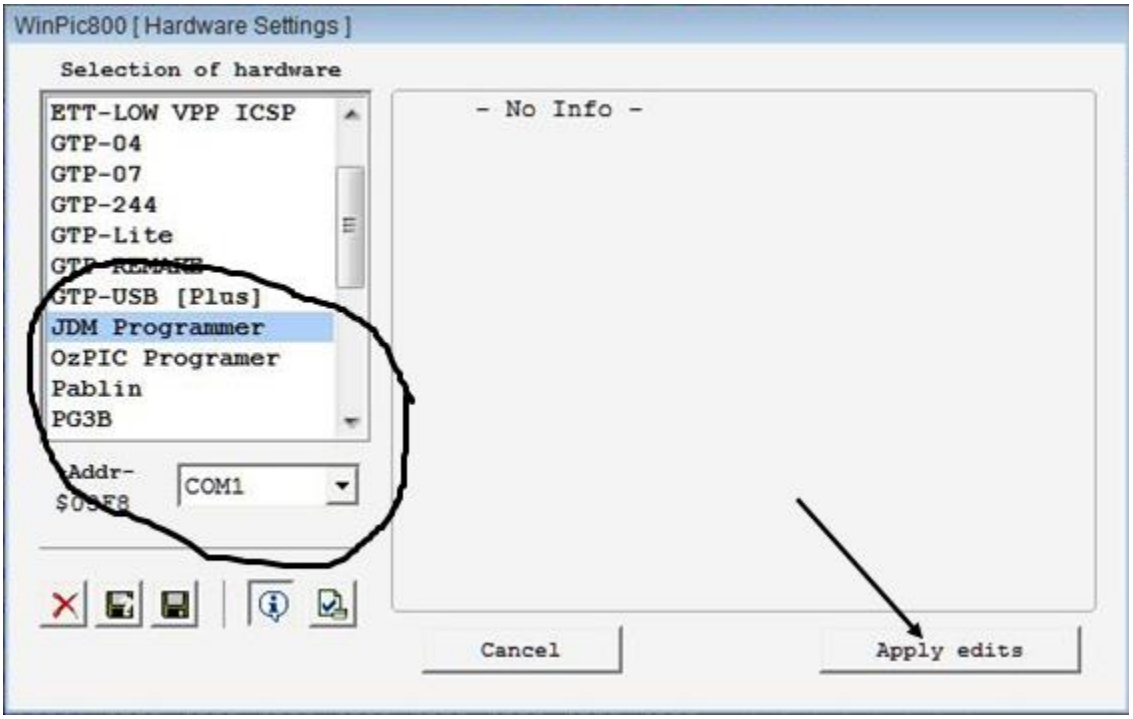
برامج الحرق تستخدم لحرق البرنامج (الكود) على الميكروكنترولر، فكما علمنا من قبل أن الميكرو عبارة عن معالج وذاكرة ويقوم المعالج بتنفيذ الأوامر المخزنة على هذه الذاكرة ... وبناء عليه فإننا نقوم بكتابة كود البرنامج الذي نريد للميكرو أن يقوم بتنفيذه من خلال برنامج الميكرو سي ثم تقوم برامج الحرق بنقل كود البرنامج (في صورة الملف الـhexadecimal) من الكومبيوتر إلى ذاكرة الميكرو عن طريق عملية تسمى الحرق أو البرمجة (Programming).

برنامج Winpic800 هو أحد هذه البرامج ويمكن تنزيله من اللينك التالي أو من خلال البحث على الإنترنت عن اسمه:

www.winpic800.com

طريقة تنصيبه سهلة ومثل أي برنامج، بعد انتهاء التنصيب قم بتشغيله، عند تشغيله لأول مرة ستظهر الشاشة بالشكل التالي:

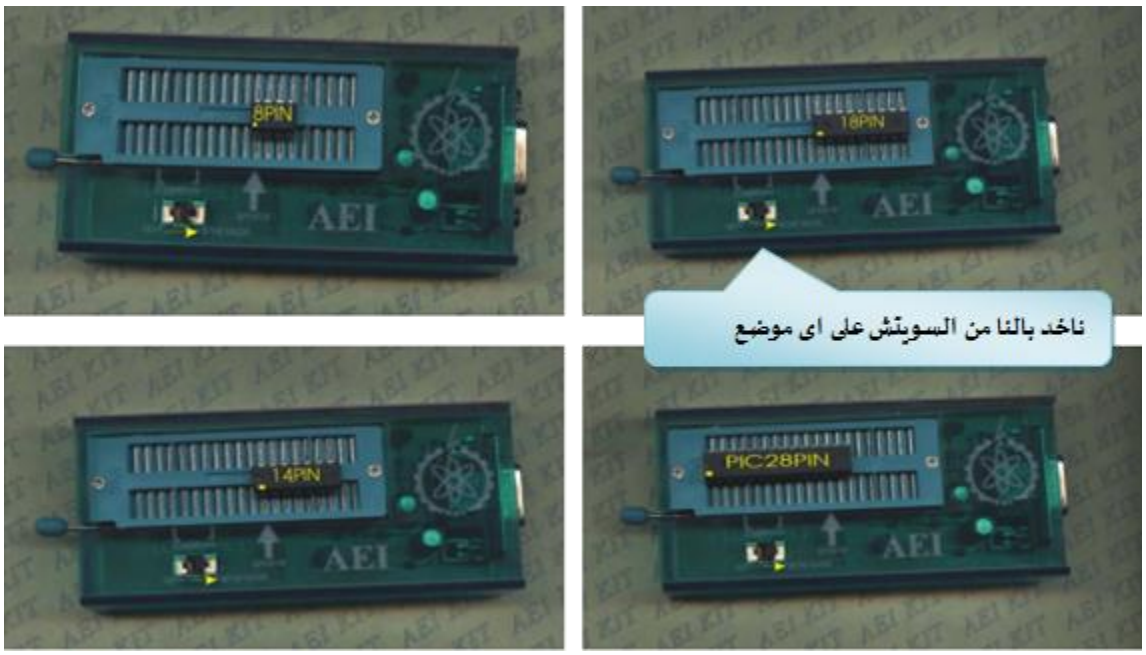


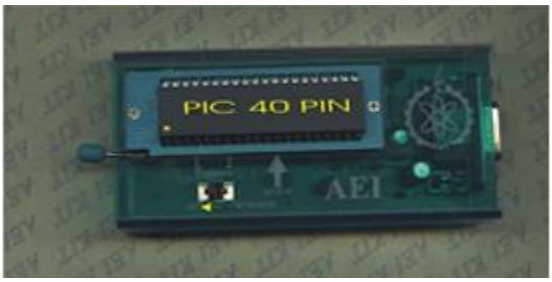


البروجرامر

الجهاز المستخدم لتنفيذ عملية الحرق أو البرمجة يسمي البروجرامر، وللبروجرامر أنواع منها ما يمكن استخدام برنامج الـ winpic معه ومنها ما يستخدم غيره، ومنها أيضا ما يتم توصيله بفتحة الـserial interface في الكومبيوتر كما في الصورة العليا ومنها ما يعمل بالـ USB كما بالصورة السفلى وميزة هذا الأخير هي استخدامه مع اللاب توب حيث أن معظم اللاب توب المنتشر حاليا لا تحتوي على وصلة سيريال، وفيما يلي صورة لبروجرامر منتشر يتم توصيله سيريال:

أشكال توصيل أنواع الميكروكنترولر المختلفة على هذا البروجرامر





توجد أنواع أخرى من البروجرامر يمكنك أن تجدها في الأسواق واستخدامها ويوجد برامج أخرى يمكن استخدامها مع هذه الأنواع مثل برنامج IC Prog لكنى لا أفضله لأنه يعطى أخطاء كثيرا.

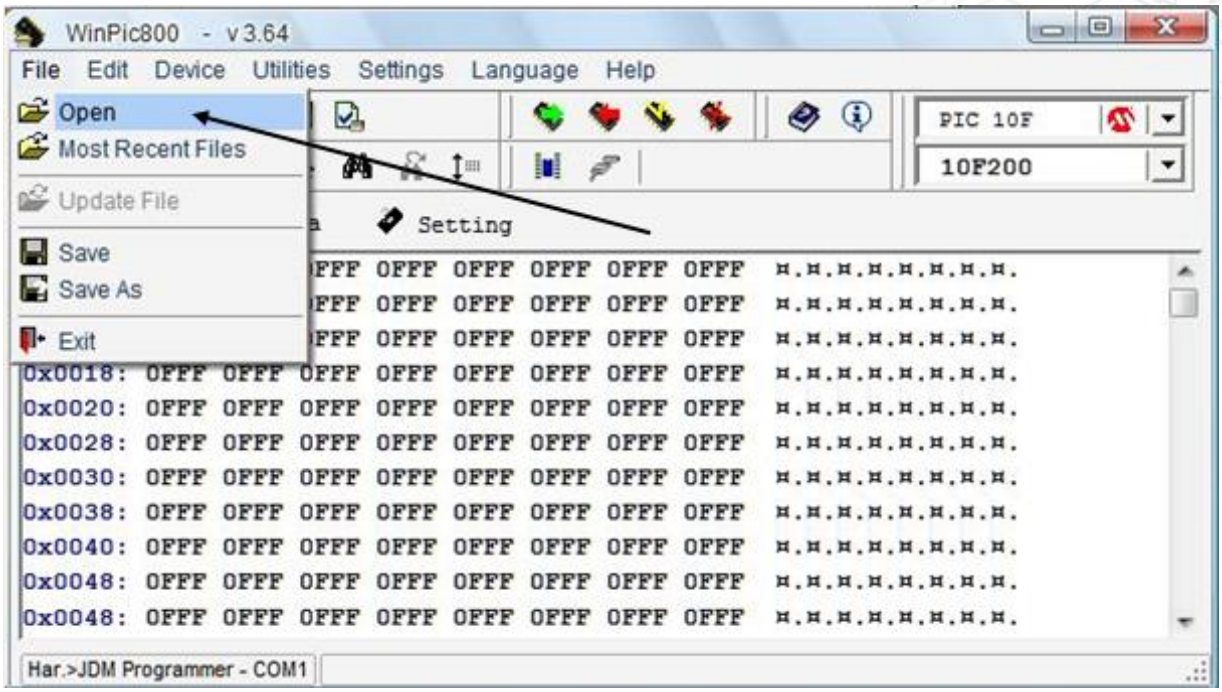
طريقة الحرق

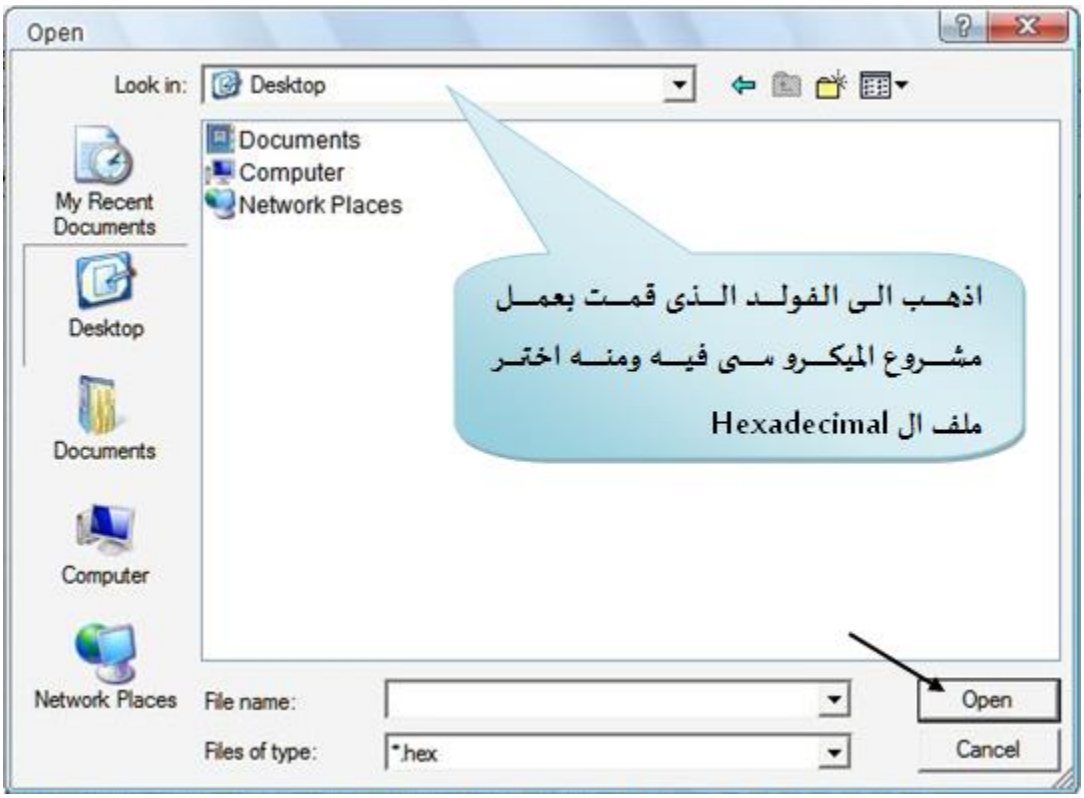
قم بتوصيل البروجرامر بالكمبيوتر عن طريق كابل السيريال المرفق معه والمشابه للموجود في الشكل.

قم بوضع الميكروكنترولر في وضعه الصحيح كما في الصور السابقة تبعا لحجمه وتأكد من أن السويتش أيضا في الوضع الصحيح.

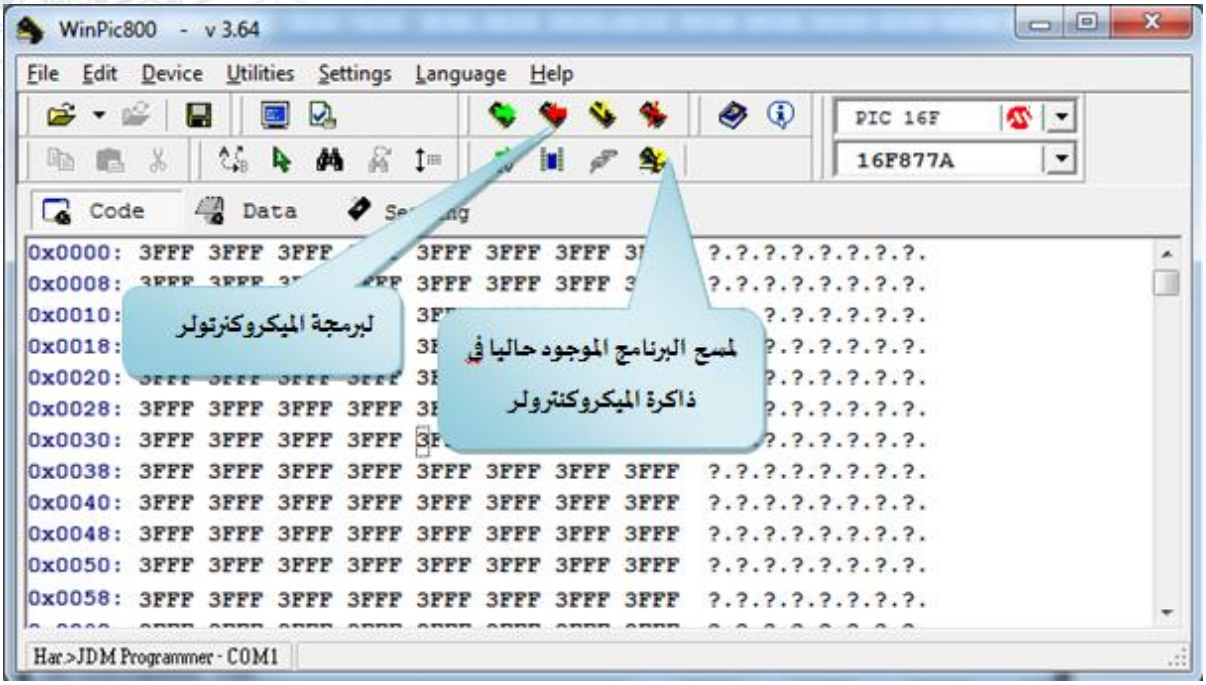
قم بتشغيل البرنامج ثم قم باختيار نوع عائلة الميكرو ونوعه ثم اضغط detect للتأكد من أن توصيل الميكرو والبروجرامر:

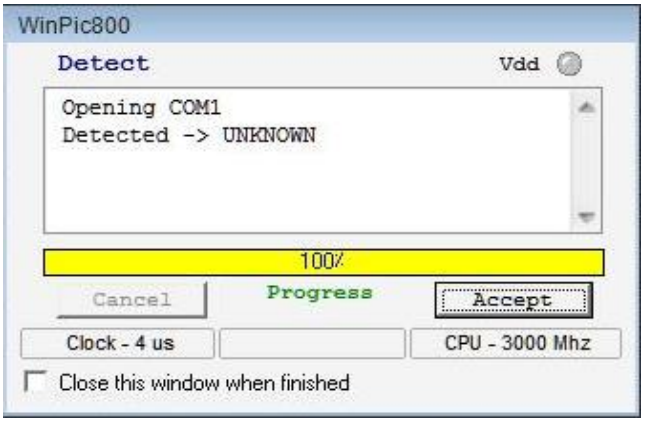
قم بتحميل الملف الـ Hexadecimal للمشروع المطلوب من الميكرو تنفيذه وذلك بالضغط على Open من قائمة File:





ثم اضغط زر البرمجة:



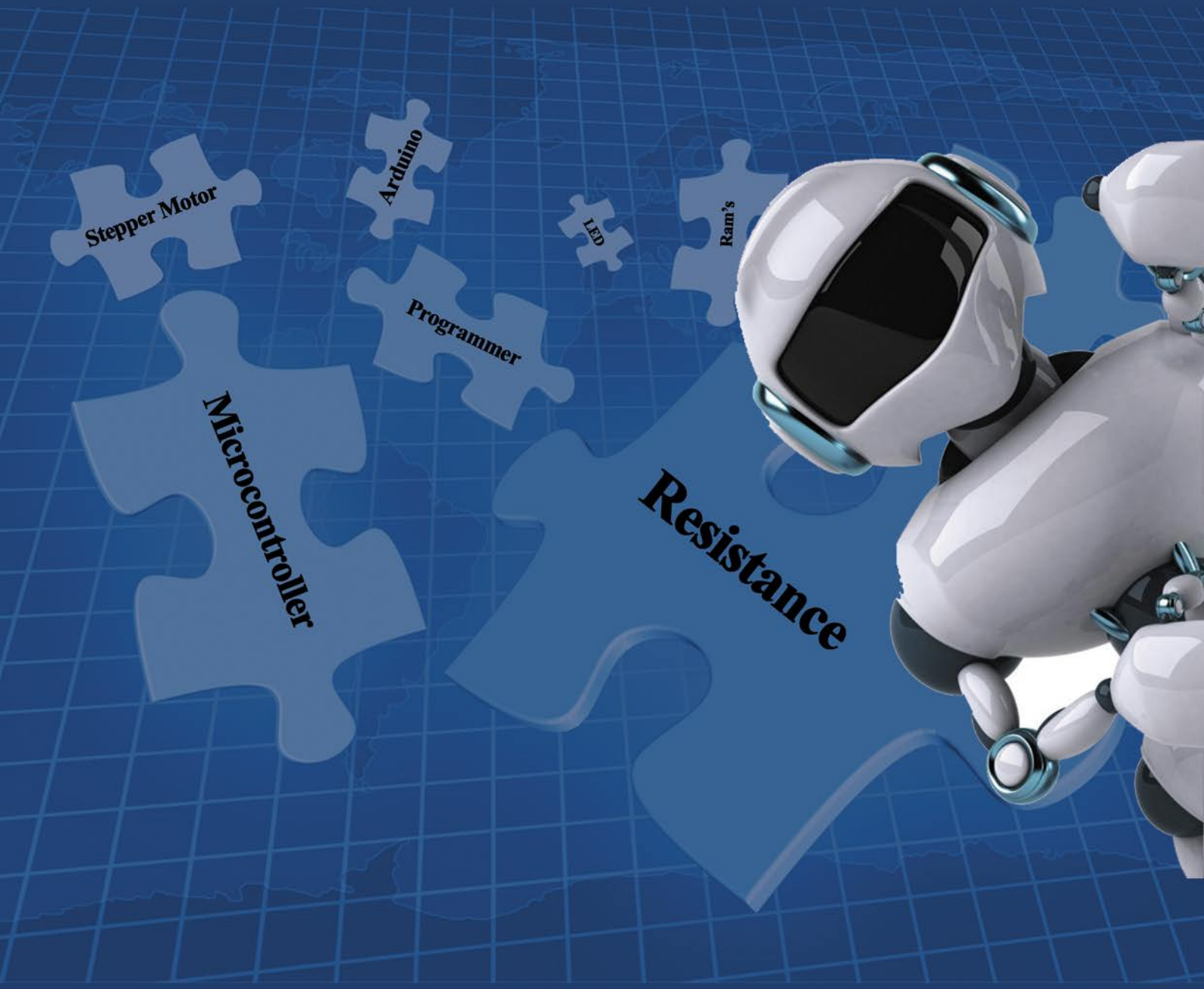


في حالة عدم وجود أي مشكلة في عملية البرمجة فستظهر رسالة مفادها انه تمت البرمجة بنجاح.

Since 1975

ElGammal Electronics

elgammalelectronics.com



Main Office
5 Mansheet Almahrany
Altahrer Cairo Egypt
Tel 002 02 27922213
Fax 002 02 27959279

Store 1
1 Boston Ebn Korish
Altahrer Cairo Egypt
Tel 002 02 27943760

Store 2
2 El Amir Kadadar
Altahrer Cairo Egypt
Tel 002 02 27960753

Store 3
23 Abd El Salam Aref
Altahrer Cairo Egypt
Tel 002 02 23961908

www.elgammalelectronics.com
Info@elgammalelectronics.com

الفصل الرابع

أساسيات برمجة البك بلغة السي

يوجد العديد من لغات البرمجة المستخدمة في برمجة الميكروكنترولر منها الأسمبلي والبيزك والسي ... وغير ذلك، لكننا سنركز في هذا الكتاب على البرمجة بلغة السي وذلك عن طريق الـCompiler المعروف وهو MikroC

الدالة الرئيسية

أي برنامج من برامج لغة السي لابد أن يحتوي على دالة تعرف بالدالة الرئيسية وهي الدالة التي يبدأ المعالج تنفيذ البرنامج منها ومنها يمكن تنفيذ أي أمر أو أي دالة أخرى وهي تكتب كالاتي:

```
void main()
{
                هنا يتم كتابة الأوامر المطلوب من الميكروكنترولر تنفيذها
}
```

وبناء على هذا نستطيع أن نقول أن أي برنامج سنكتبه سوف يحتوي على هذه الدالة، ولكن الأوامر التي بداخلها فستكون على حسب المشروع الذي نريد أن نقوم بتنفيذه.

الحلقات التكرارية

لن نقوم بشرحها بالتفصيل في هذا الجزء وإنما سنكتفي فقط بشرح إحداهم وهي while والتي تستخدم لتنفيذ مجموعة من الأوامر بعد اختبار شرط معين فطالما كان الشرط محقق تستمر في تنفيذ هذه الأوامر مرة بعد مرة ولكن عندما يتغير الشرط ويصبح غير محقق تنتهي هذه الحلقة من تكرار هذه الأوامر وينتقل التنفيذ للسطر التالي لها، وهي تكتب بالشكل الاتي:

```
while (      الشرط المراد اختباره      )
{
                مجموعة الأوامر المطلوب تنفيذها إذا تحقق الشرط
}
```

وهذا مثال لتوضيح ذلك:

مثال: نريد من الميكروكنترولر أن يقوم بتشغيل التكييف إذا زادت درجة الحرارة عن ٣٠ درجة:

```
while( temperature > 30 )
{
    Operate the air conditioning code ;
}
```

فإذا كان درجة الحرارة أكبر من ٣٠ درجة فان الميكروكنترولر سوف يستمر في تنفيذ الأوامر التي بين القوسين { } والتي هي أوامر تشغيل التكييف.

ملحوظة: عندنا تكون فعلا درجة الحرارة أكبر من ٣٠ درجة ففي هذه الحالة يكون الشرط (Temperature > 30) قيمته تساوى واحد فيتم تنفيذ الأوامر ولكن إذا كانت درجة الحرارة أقل من ٣٠ درجة فإن قيمته ستكون بصفر، ولكن while تنفذ الأوامر التي بداخل أقواسها فقط إذا كان الشرط محقق أي إذا كانت قيمته تساوى واحد، ومن هنا نسأل سؤال وهو إذا كانت while مكتوبة كما بالشكل الآتي فكم مرة سيتم تنفيذ الأوامر التي بداخلها:

```
while( 1 )
{
    move the robot forward 3 seconds ;
}
```

الإجابة ...سيتم تنفيذها عدد لانهائي من المرات دون توقف ... لماذا ؟؟؟

لأنه في المرة الأولى ستقوم while باختبار الشرط فتجد قيمته واحد فتنفذ ما بداخلها ثم تقوم باختبار الشرط فتجده بواحد فتنفذ ما بداخلها ... وهكذا، وحيث أن الشرط دائما محقق (يساوي واحد) ولا يوجد أي طريقة تجعله يتغير (كتغير الحرارة في المثال السابق) فإن الحلقة تستمر في التكرار إلى ما لانهاية !!!

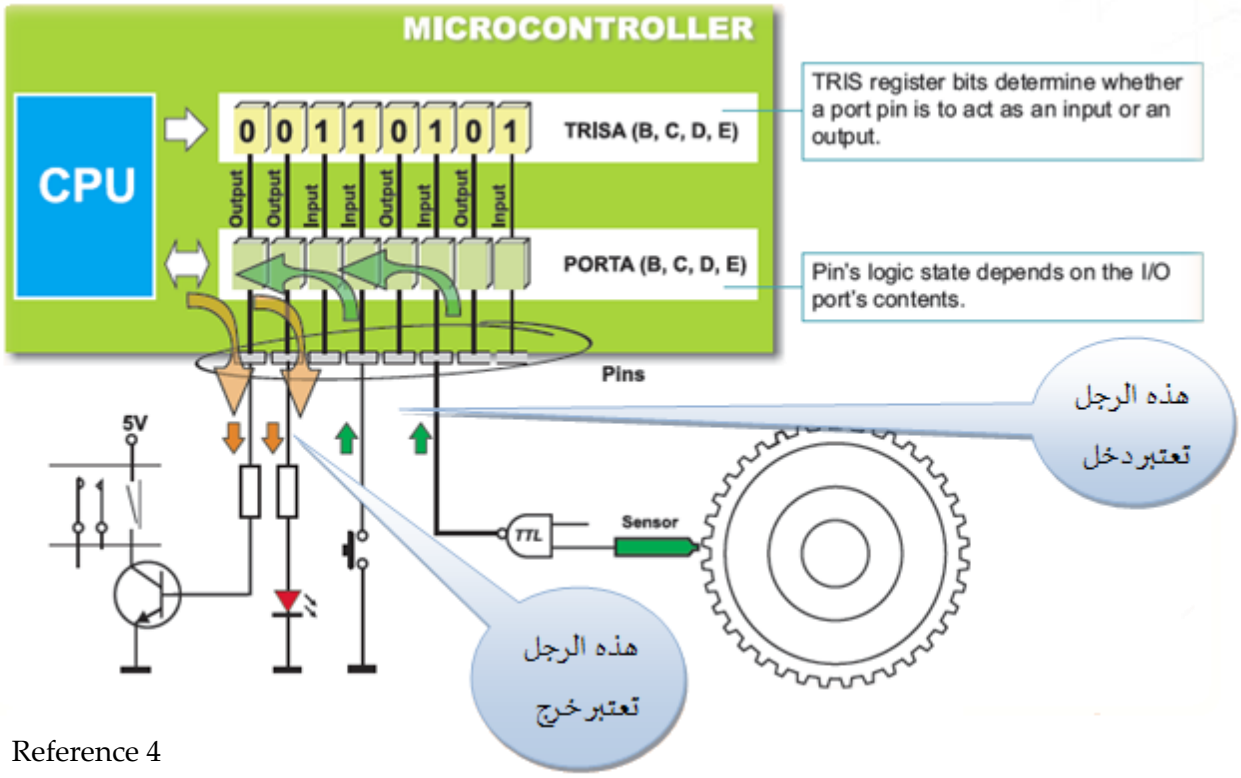
في نهاية هذا الجزء نحب أن نوضح أنه بطبيعة عمل الميكروكنترولر الذي ينفذ وظيفة معينة باستمرار فإن معظم البرامج التي يقوم الميكروكنترولر بتنفيذها ينفذها ما لانهاية من المرات وليس فقط مرة واحدة ثم يقف بعد ذلك عن العمل، ولذا فان الغالبية العظمى من البرامج التي سنكتبها ستكون بالشكل التالي:

```
void main()
{
                هنا تكتب مجموعة من الأوامر سنتعرض لها لاحقا تعرف بـ
                Initialization & data direction
    while( 1 )
    {
                        الأوامر المراد تنفيذها
    }
}
```

برمجة رجول الميكروكنترولر:

تحديد اتجاه الداتا Data direction:

كمثال يوجد ٣٣ رجل في البك 16F877A تؤدي وظيفة الد يجيتال أي تعمل كدخل رقمي (لقراءة حالة سويتش مثلا) أو كخرج رقمي (للتحكم في إضاءة الليد مثلا)، السؤال هنا: بما أن الرجل الواحدة يمكن أن تعمل كدخل أو كخرج فكيف يمكن تحديد أن رجل محددة من رجول الميكروكنترولر تعمل كدخل وأخرى تعمل كخرج ؟؟؟



Reference 4

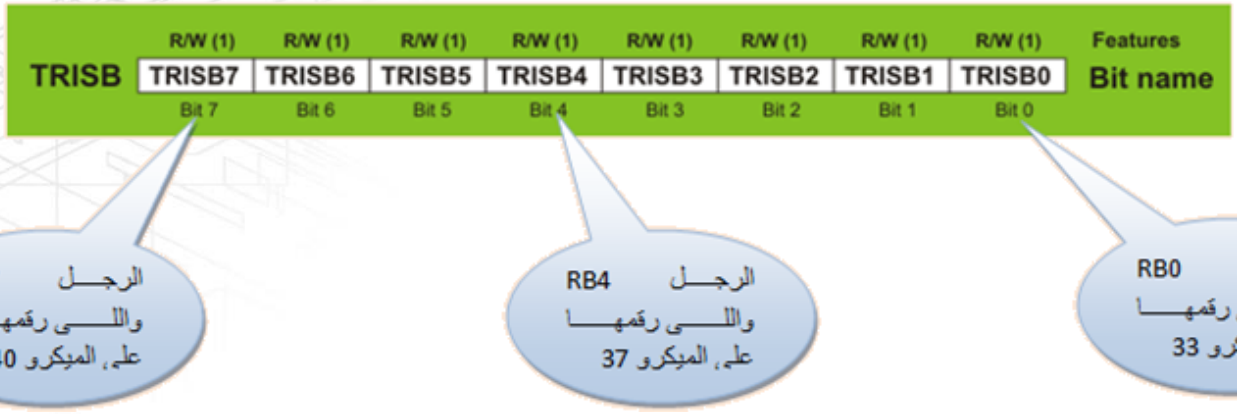
يتم ذلك من خلال أمر معين وهو الأمر TRIS وله صيغة معينة وهي كالآتي:

```
TRISX = 0B10101010;
```

حيث يمثل الـ X اسم المخرج الذي نريد التحكم في اتجاه الداتا على رجولة ويتم استبداله في الأمر بهذا الاسم (A-B-C-D-E) وتستخدم الـ 0B وهي عبارة عن صفر وحرف البى لتدل على أن الرقم الذي يليها هو رقم ممثل بالنظام الثنائي أي يكون صفر أو واحد كما هو مكتوب ولا ينظر له كقيمة عشرية كما هو المعتاد.

وحيث أن أغلب المخارج يتكون من ٨ رجول فغالبا ما يكون بعد ال 0B يوجد ٨ قيم مقابلة لكل رجل في المخرج وكل قيمة منها تأخذ إما واحد وإما صفر حسب عمله كدخل أو كخرج، فلو افترضنا مثلا أننا نحدد المخرج B ونريد التحكم في وظيفة رجوله فسيكون الأمر كالتالي:

Reference 4



فمثلا إذا كنت تريد أن تجعل الرجل RB7 تعمل كدخل وباقي رجول المخرج B تعمل كخرج فسوف يكون الأمر كالتالي:

```
TRISB = 0B10000000;
```

وهذا معناه أنك إذا كنت تريد أن تجعل رجل معينة كدخل فاجعل القيمة المقابلة لها في الأمر TRIS تكون بواحد، واجعل القيمة تساوى صفر إذا كنت تريد هذه الرجل أن تعمل كخرج.

مثال: الرجل رقم ٠ و١ و٢ و٣ في المخرج B تعمل كخرج والباقي دخل:

```
TRISB = 0B11110000;
```

مثال: المخرج B كله يعمل كدخل:

```
TRISB = 0B11111111;
```

مثال: المخرج B كله خرج:

```
TRISB = 0B00000000;
```

مثال: الرجول ٠ و٢ و٤ و٦ في المخرج C تعمل كخرج والباقي كدخل:

```
TRISC = 0B10101010;
```

هكذا تعلمنا كيفية التعامل مع مخرج كامل ... لكن هل يمكن التعامل مع رجل واحدة فقط؟؟؟
... الإجابة: نعم وذلك من خلال الأمر الآتي:

```
TRISX.BN = 0;
```

حيث يمثل الـ X اسم المخرج ويمثل الـ N رقم الرجل في المخرج

مثال: لجعل الرجل RD3 تعمل كخرج:

```
TRISD.B3 = 0;
```

مثال: لجعل الرجل RC7 تعمل كدخل:

```
TRISB.B7 = 1;
```

ملحوظة هامة فيما يخص مخارج الأنالوج:

ذكرنا من قبل أن كلا من المخرج A والمخرج E من الممكن يستخدموا مع الأنالوج وكذلك مع الد يجيتال بخلاف المخارج B, C, D الذين يعملون كد يجيتال فقط، وبالتالي فإنه للتعامل أي من المخرجين A والمخرج E لابد أولا أن نحدد هل سنوصل عليهم أنالوج أم د يجيتال وإذا حددناهم كد يجيتال نقوم بعد ذلك بتحديد ما إذا كانوا سيستخدموا كدخل أم كخرج.

يتم تحديد المخرج A والمخرج E كد يجيتال عن طريق الأمر التالي:

```
ADCON1 = 0x06;
```

وبالتالي فإنه عند استخدام المخرج A والمخرج E كد يجيتال يجب كتابة الأمر السابق أولا ثم كتابة الأمر TRIS لنحدد اتجاه الداتا.

يجب معرفة أن هذا الأمر بهذا الشكل وهذه القيمة يستعمل مع البك 16F877A وإذا كان هناك ميكرو آخر فان التعامل قد يكون مختلف وسنعرف لاحقا كيف نتعامل مع أي ميكروكنترولر.

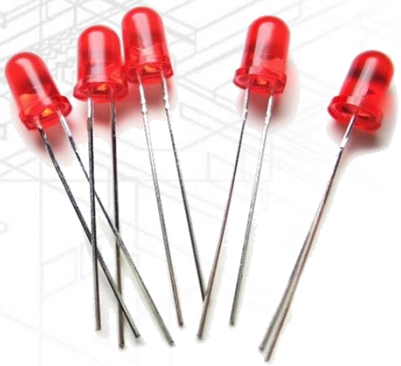
مثال: لجعل الرجل RA2 تعمل كخرج (د يجيتال):

```
ADCON1 = 0x06;
TRISA.B2 = 1;
```

إدخال وإخراج داتا على أرجل الميكروكنترولر

بالأوامر السابقة نحن فقط قمنا بتحديد اتجاه الداتا هل ستكون كخرج أم دخل، بمعنى هل الميكرو سيخرج قيم على رجوله (خرج) أم سنضع نحن القيم على رجوله (باستخدام أي مكون آخر كالسويتش) ويقوم الميكرو بقراءتها (دخل).

ولكن إذا أردنا إخراج قيمة على رجول خرج أو قراءة قيمة من رجول دخل فماذا نفعل؟؟ فمثلا لو قلنا أننا سنجعل المخرج B كخرج وسنوصله بمجموعة من الليدات لكي يقوم بإضاءتها فكيف سنخبر الميكرو أن يخرج على رجول المخرج B خمسة فولت ليضيء الليدات ...



يتم هذا من خلال أمر مشابه في كتابته امر TRIS وهو الأمر PORT حيث يكتب بنفس الطريقة:

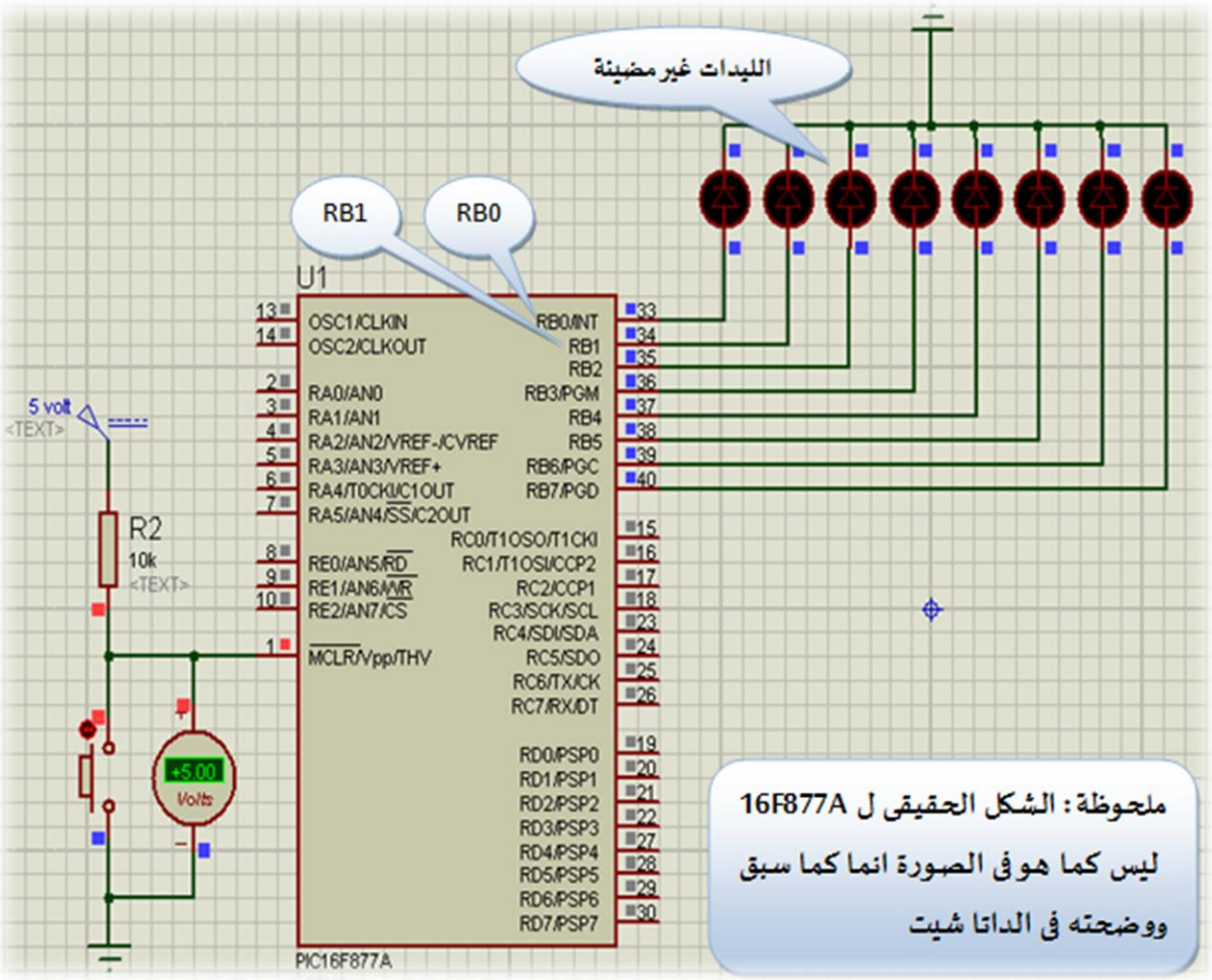
```
PORTB = 0B11111111;
```

فمثلا فإن هذا الأمر يستخدم لإخراج خمسة فولت على كل رجول المخرج B (فمن المعروف أن القيمة ١ في النظام الثنائي تناظر خمسة فولت كجهد والصفر يناظر الصفر فولت).

نذكر بأن هذا الأمر لابد أن يسبقه أمر تحديد الاتجاه المذكور في القسم السابق.

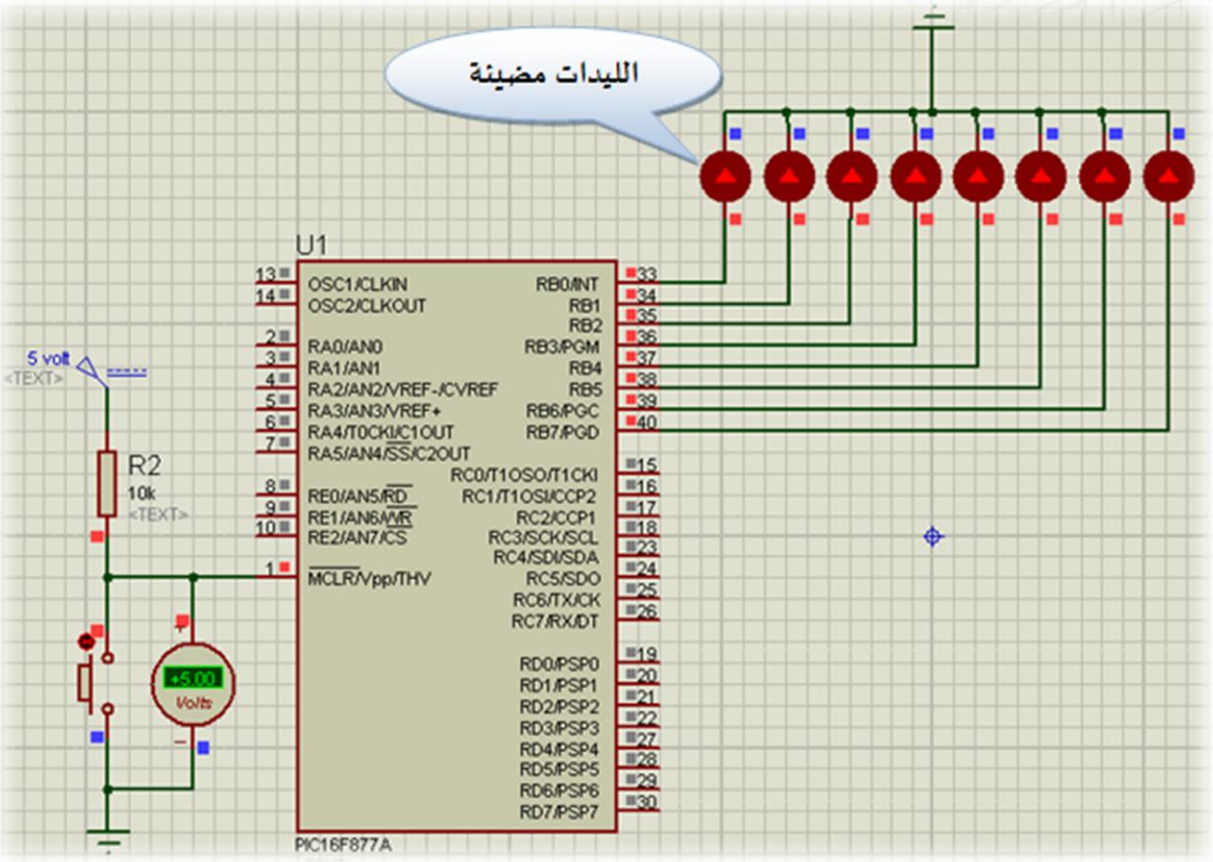
مثال: لإطفاء كل الليدات الموجودة على المخرج B:

```
TRISB = 0B00000000;
PORTB = 0B00000000;
```

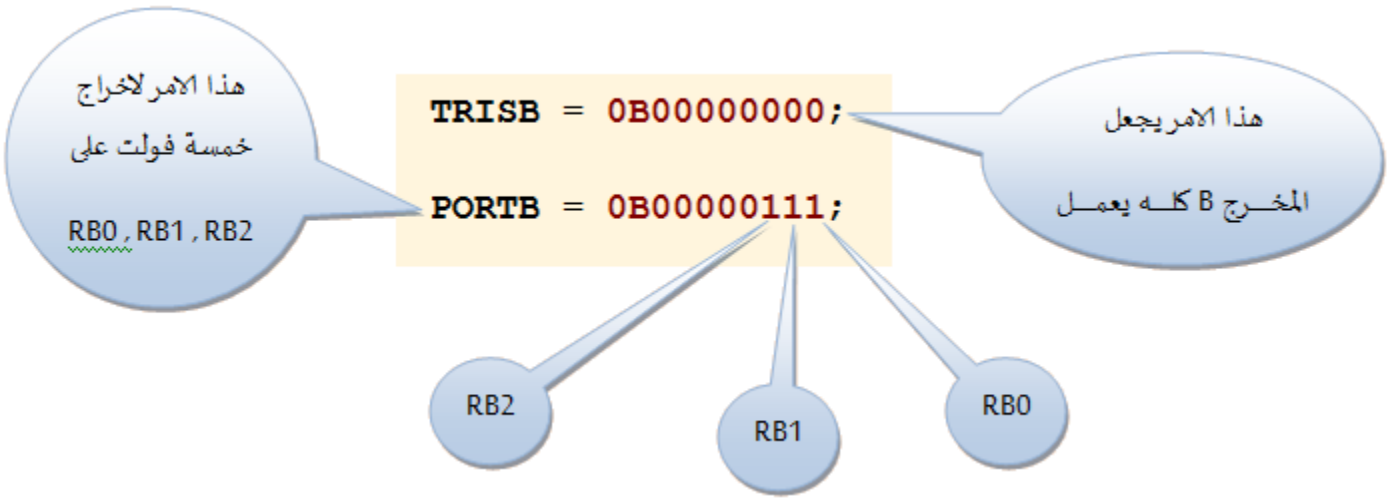


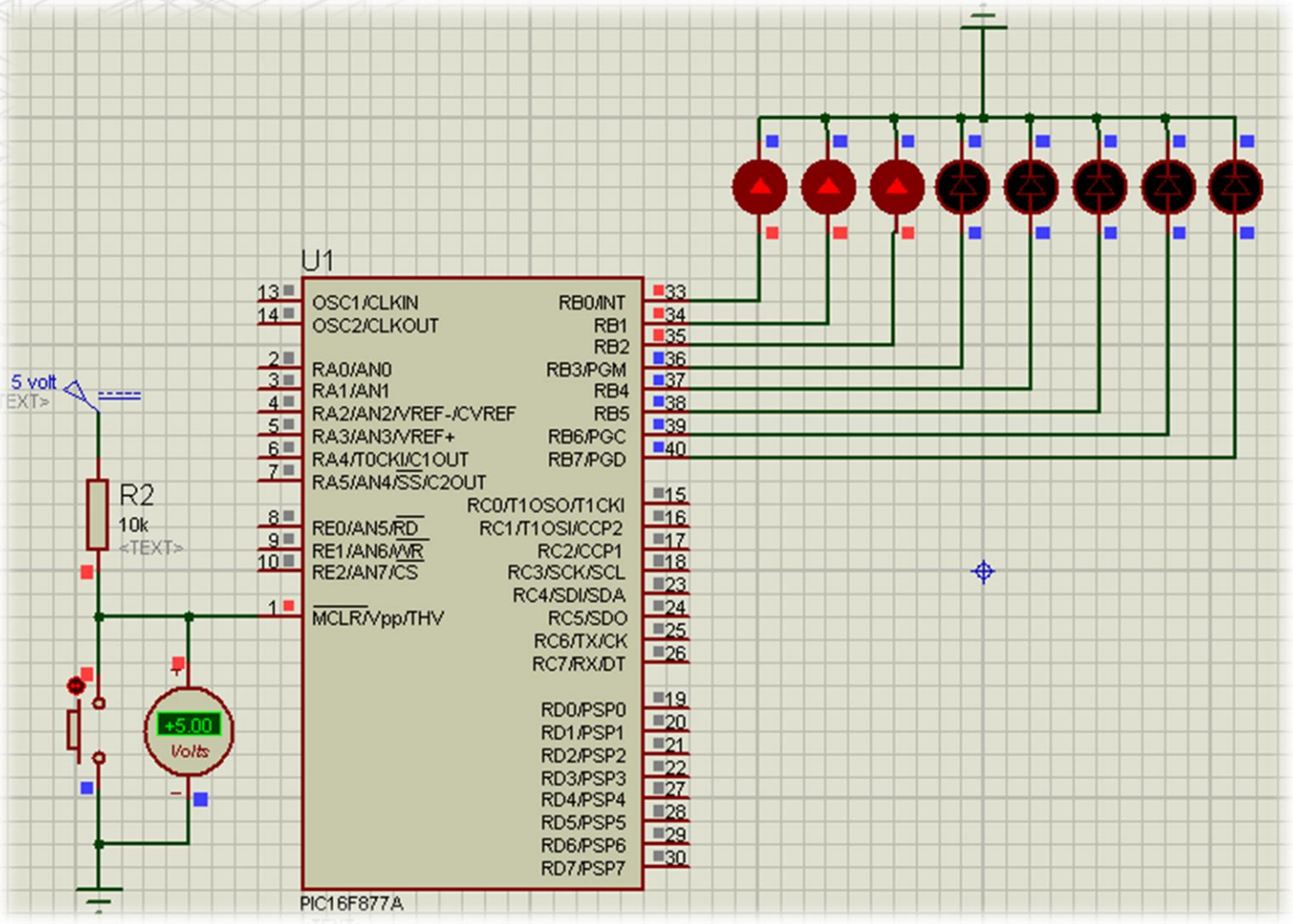
مثال: لإنارة الليدات الموجودة على المخرج B كله:

```
TRISB = 0B00000000;
PORTB = 0B11111111;
```



مثال: لإضاءة الليدات الموصلة على RB0, RB1, RB2:

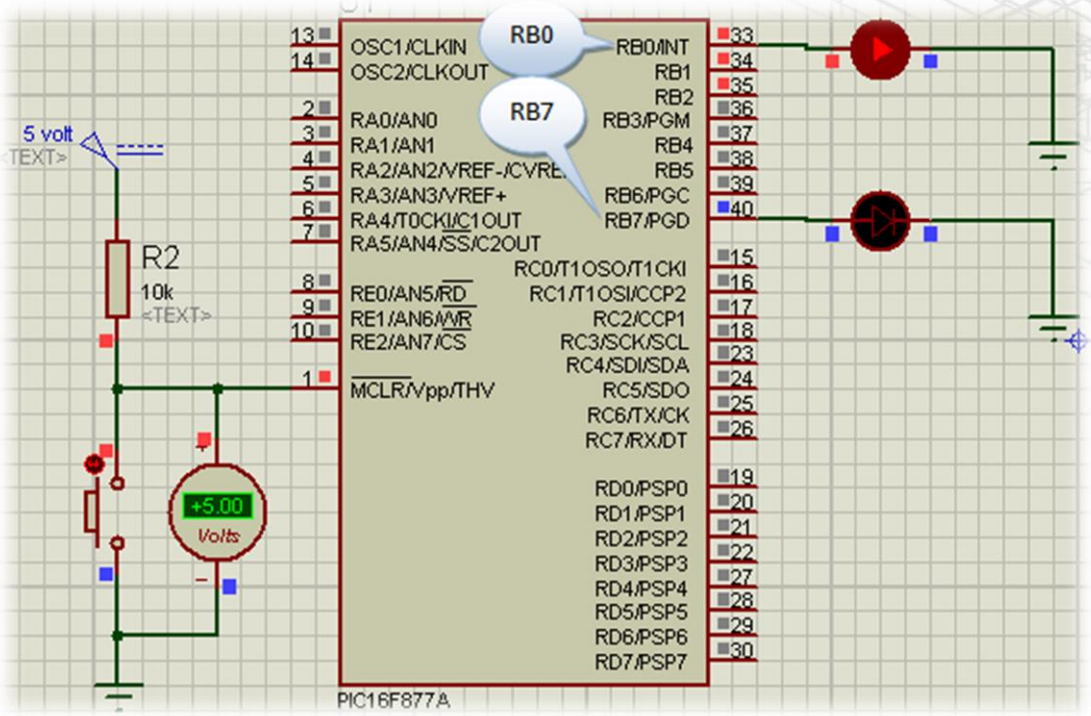




وبنفس طريقة الأمر TRIS يمكن التعامل بالأمر PORT مع رجل واحدة فقط.

مثال: لإضاءة الليد الموصلة على الرجل RB0 وإطفاء الليد الموصلة على RB7: أولا لابد من وصف الرجول RB0, RB1 كخرج عن طريق الأمر TRIS ثم بعد ذلك نخرج عليهم القيم المطلوبة عن طريق الأمر PORT:

```
TRISB.B0 = 0;
TRISB.B7 = 0;

PORTB.B0 = 1;
PORTB.B7 = 0;
```



الدوال الفرعية

عرفنا أن أي برنامج مكتوب بلغة السي لابد أن يحتوي على الدالة الرئيسية وهي الدالة التي يبدأ المعالج التنفيذ بها، إلا إن البرنامج من الممكن أن يحتوي على دوال أخرى تقوم بوظائف معينة – نقوم نحن بتحديدها – ويتم تنفيذ وظيفة أي دالة فرعية عن طريق ندائها في الدالة الرئيسية ...

من فوائد الدوال الفرعية

- تساعد في اختصار كود الدالة الرئيسة حيث يكتفى بذكر اسمها فقط ليتم تنفيذها
- تسهيل مراجعة وتصحيح الأكواد واكتشاف الأخطاء فيها.
- تستخدم لتلاشى عمليات تكرار الكود في الدالة الرئيسية.

كيفية تعريف دالة فرعية

```
type NAME ( معاملات الإزاحة )
{
                    الأوامر المراد من الدالة تنفيذها
   return result;
}
```

حيث تمثل الكلمة type نوع البيانات التي ستقوم الدالة بإرجاعها ولتفسير ذلك نأخذ هذا المثال: بفرض أننا نريد عمل دالة نعطيها قيمتين تقوم بجمعهم وتعطينا الناتج، ما يهمنا هنا هو هل الناتج الذي ستعطينا إياه من النوع int أم float أم char أم غير ذلك ؟؟ فإذا كان نوعه int نستبدل كملة type بكلمة النوع int وهكذا باقي الأنواع على حسب نوع البيانات الراجعة.

وتمثل كلمة NAME اسم الدالة الذي سنستخدمه لندائها من الدالة الرئيسية، ويمكنك كتابة أي اسم تريده يحقق بعض الشروط ومنها:

- أن يعبر الاسم عن الوظيفة التي تؤديها الدالة فلو كانت تستخدم في عمليات الجمع نسميها على سبيل المثال Add أو نسميها مثلا Add_Salary لو كنا نستخدمها لجمع الرواتب ... وهكذا.
- أيضا لا يمكن للاسم أن يبدا برقم.
- لا يمكن أن يحتوي الاسم على مسافة أو بعض العلامات الأخرى مثل (*) أو (.) أو (/) أو (&) أو(%) ... أما العلامة (_) فهي مقبولة في الاسم كما في النقطة السابقة.

أما معاملات الإزاحة فهي عبارة عن القيم التي سنعطيها للدالة لإجراء العمليات عليها، وفى مثال جمع العددين تكون معاملات الإزاحة هي الرقمين الذين نريد جمعهما.

ثم يأتي القوسين الموضحين واللذان يسميان بأقواس المجموعة { } وبينهما يتم كتابة الأوامر التي نريد للدالة أن تقوم بتنفيذها، هذه الأوامر تنتهي بالأمر:

```
return result;
```

الذي يرجع النتيجة من الدالة، فلو أخذنا مثال جمع العددين السابق ذكره وفرضنا أننا جمعنا العددين ووضعنا الناتج في المتغير result فان هذا الأمر يقوم بإرجاع هذه القيمة التي تمثل ناتج الجمع والتي يمكن استقبالها في الدالة الرئيسية في متغير آخر لنكمل عليه إجراء العمليات.

> **ملحوظة:** يمكن للدالة أن تقوم بالوظيفة المنوطة لها دون الحاجة لإرجاع نتائج وفي هذه الحالة لن نستخدم الأمر return في نهاية الأوامر وسيكون نوع الدالة هو void ...

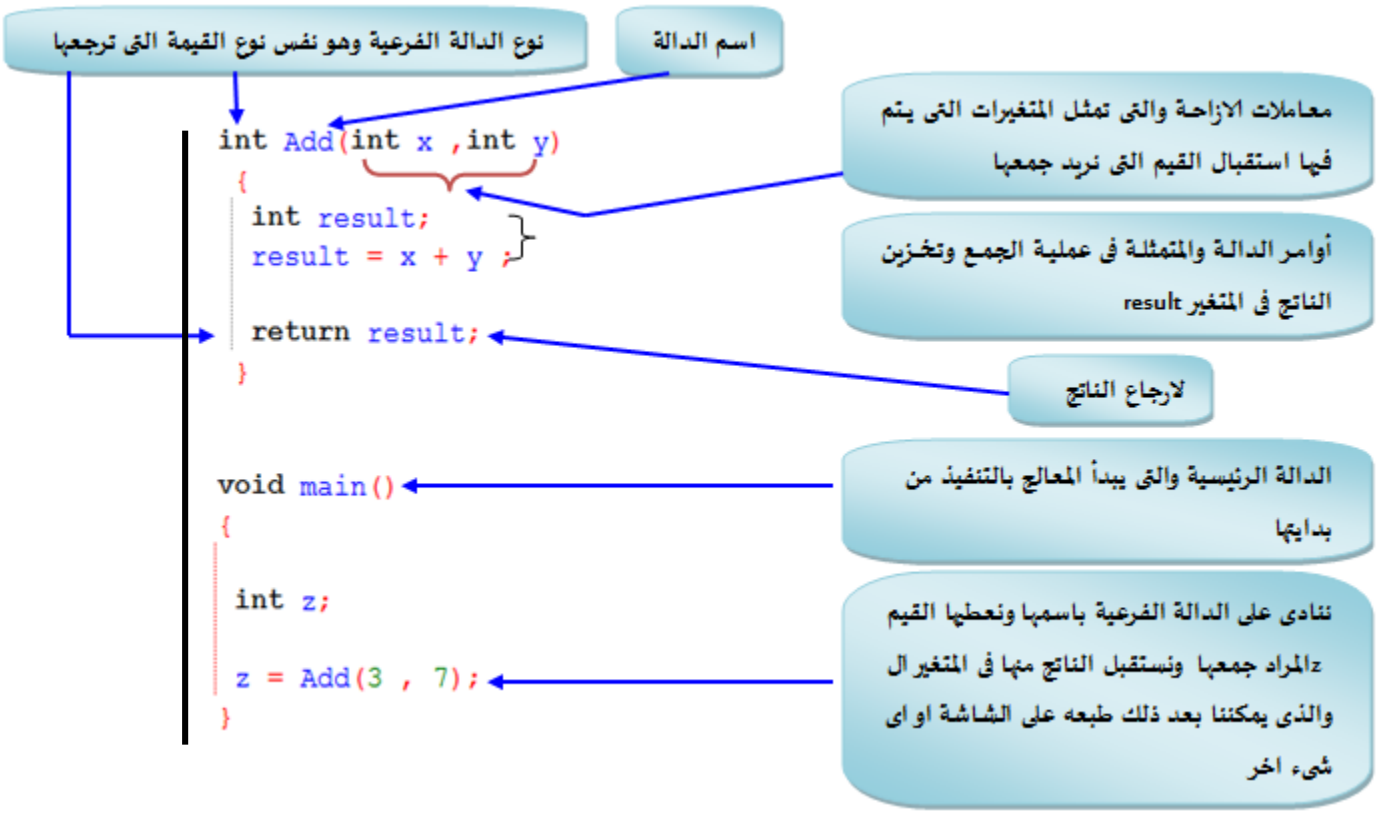
كيفية تنفيذ دالة فرعية

علمنا أن المعالج يقوم ببدء التنفيذ من الدالة الرئيسية فكيف سيقوم المعالج اذا بتنفيذ الدوال الفرعية ؟؟؟ يتم ذلك بنداء الدالة الفرعية داخل الدالة الرئيسية بذكر اسمها وتخصيصه لمتغير من نفس نوع الداتا الراجعة (في حال إرجاع داتا) ...

وفيما يلي مثال لنوضح به كيفية استخدام الدوال الفرعية، وهو المثال الذي المذكور سابقا والخاص بجمع عددين، يمكن النظر لشكل البرنامج إجمالا أولا:

```
int Add(int x ,int y)
 {
  int result;
  result = x + y ;

  return result;
 }


void main()
{

 int z;

 z = Add(3 , 7);
}
```

ثم نأتي لتفسير أوامر هذا البرنامج على الشكل الآتي:



ملحوظة: قد يتم كتابة الدالة الفرعية داخل البرنامج قبل الدالة الرئيسية أو بعدها ولكن إذا تم كتابتها بعدها لابد من كتابة أمر قبل الدالة الرئيسية لكن هذا الجزء لن نتعرض له ...

ملحوظة: المتغير الذي يتم تعريفه داخل أي دالة (رئيسية كانت أم الفرعية) لا يمكن استخدامه خارج هذه الدالة، ففي البرنامج السابق المتغير z تم تعريفه داخل الدالة الرئيسية وبالتالي لا يمكن استخدامه داخل الدالة الفرعية Add، ولذلك يطلق على هذه المتغيرات اسم local variables أي المتغيرات المحلية ..
وإذا أردنا أن نجعل متغير ما متاحا لكل الدوال استخدامه وتغيير قيمته فإننا نقوم بتعريفه قبل جميع الدوال ... أي لا يتم تعريفه داخل أي دالة ..

مثال

نريد عمل برنامج يحتوي على دالة تقوم بحساب القيمة العظمي بين قيمتين وإرجاعها للدالة الرئيسية.

```
float Max(float x ,float y)
 {
  float result;

  if(x > y)
    return x;
  else
    return y;
 }


void main()
{

 float z;

 z = max(3.5 , 3.8);
}
```

مثال

دالة تقوم بجمع عددين ولكن هنا العددين داخل من الدالة الفرعية ولا يتم أرسالهم من الدالة الرئيسية وبالتالي لن يكون هناك أي معاملات إزاحة:

```
int add( )
 {
  int result,x,y;

  x = x + y;
 }
```

لا توجد معاملات ازاحة

```
void main()
{
 float z;
 z = add();
}
```

لم نعطى الدالة أى قيم لانها فى تعريفها لم يوجد بها معاملات ازاحة

مثال

برنامج به دالة فرعية تقوم بضرب قيمة متغير في ١٠٠ وبعد ذلك تقوم الدالة الرئيسية بطبع الرقم على الشاشة:

```
int number=10;

void MUL_100( )
{
  number = number*100;
}

void main()
{
 MUL_100();
 print the number
}
```

متغير معرف قبل كل الدوال وبالتالى يمكن لكل الدوال ان تتعامل معه وتغير فى قيمته

الدالة نوعها void لانها لا ترجع قيما لانه لاحاجة لارجاع المتغير number حيث يمكن للدالة الرئيسية التعامل معه مباشرة

لضرب الرقم فى 100

تنادى على الدالة الفرعية وبالتالى يتم تنفيذها فاذا لم تناديها بكتابة اسمها كما هو موضح فانها لن يتم تنفيذها

لطباعة الرقم بعد ان تم ضربه فى 100 باستخدام الدالة الفرعية لكن هذا ليس امر لكنها جملة قصدت منها التعبير على امر الطباعة الذى سنتعرف عليها لاحقا

الفصل الخامس

مشاريع عملية على برمجة مخارج الميكرو

تعلمنا في الفصول الماضية ما هو الميكروكنترولر وكيفية برمجة رجوله، وسنركز في هذا الفصل على توضيح هذه الكيفية عن طريق اختيار مجموعة من المشاريع وشرحها وتنفيذها ببعض من التفصيل

مشروع الفلاش (بأكثر من طريقة)

المطلوب من هذا المشروع هو برمجة ليد ليقوم بعمل إضاءة فلاشيه، وتتلخص فكرته في أن الميكرو يقوم بإضاءة الليد لفترة زمنية معينة ولتكن ثانية، ثم يقوم بإطفائه لمدة ثانية أخرى، ثم يعاود إضاءته مرة أخرى ... وهكذا.

أول خطوة لتنفيذ أي مشروع هي تحديد عدد الرجول التي ستحتاجها كدخل أو كخرج، وفى هذا المشروع لا يوجد أي دخل (لأنه لا يوجد أي سويتشات أو سنسورات) والخرج هو خرج واحد فقط خاص بالليد.

لنتذكر سويا أن أي برنامج ميكروكنترولر بلغة السي يكون بالشكل الآتي:

```
void main()
{
        أوامر تحديد الاتجاه على رجول الميكروكنترولر أو بمعنى أدق أوامر
                                TRIS
    while( 1 )
    {
                الأوامر المراد تنفيذها عدد لا نهائي من المرات
    }
}
```

نقوم باختيار رجل من رجول الميكروكنترولر لنقوم بتوصيل الليد عليها، ولتكن RC0 وهو الرجل رقم ١٥ في الميكرو PIC16F877A، وبالتالي نقوم بكتابة الأمر الذي يحددها كخرج كالاتي:

```
TRISC.B0 = 0;
```

ولإضاءة الليد يلزم إخراج ٥ فولت على الرجل RC0 وهذا يتم من خلال الأمر الآتي:

```
PORTC.B0 = 1;
```

ولإطفاء الليد ثانية نكتب الأمر الآتي:

```
PORTC.B0 = 0;
```

وفى هذه الحالة يصبح البرنامج كالآتي:

```
void main()
{

    TRISC.B0 = 0;

  while(1)
  {
    PORTC.B0 = 1;

    PORTC.B0 = 0;
  }
}
```

لتحديد RC0 كخرج

لاضاءة الليد

لاطفاء الليد

يوجد مشكلة بسيطة في البرنامج المذكور وهي أننا لم نحدد المدة الزمنية التي سيكون الليد فيه مضيء وكذلك التي سيكون فيها غير مضيء.
أمر لغة السي الذي يستخدم لذلك هو الأمر التالي:

```
delay_ms(1000);
```

معنى الامر هو الانتظار لمدة ثانية

اختصار ميللى ثانية millisecond

1000 ميللى ثانية تساوى ثانية

والسؤال هنا: أين يتم كتابة هذا الأمر ...؟؟؟ يتم كتابته بعد امر الإضاءة وأيضا بعد امر الإطفاء حتى يضئ الليد لمدة ثانية ثم يطفئه لمدة ثانية أخرى وعليه يكون البرنامج كالآتي:

```
void main()
{

    TRISC.B0 = 0;

  while(1)
  {
    PORTC.B0 = 1;
     delay_ms(1000);

    PORTC.B0 = 0;
     delay_ms(1000);
  }
}
```

يتم تكرار الاوامر التى بداخلها عدد لا نهائى من المرات

اضاءة الليد لمدة ثانية

اطفاء الليد لمدة ثانية

شرح تفصيلي لخطوات تنفيذ البرنامج

يبدأ الميكرو تنفيذ البرنامج بداية من الدالة الرئيسية وهى دالة الـ main فيجد أن أول أمر هو أمر `TRISC.B0=0;` وبتنفيذه يقوم الميكرو بجعل الرجل RC0 تعمل كخرج، ثم يدخل على حلقة `while` ويختبر قيمة الشرط فيجدها واحد أي أن الشرط محقق فينفذ ما بداخل الـ while من أوامر، الأمر الأول داخلها هو `PORTC.B0=1;` وهو لإضاءة الليد عن طريق إخراج ٥ فولت على الرجل RC0 ثم الانتظار لمدة ثانية على هذه الحال دون تنفيذ أي أوامر عن طريق الأمر delay وبعد ذلك يقوم بإطفاء الليد عن طريق إخراج صفر فولت على RC0 بواسطة الأمر `PORTC.B0=0;` ثم ينتظر ثانية أخرى بواسطة أمر `delay` آخر وعند هذه النقطة يكون الميكرو قد انتهى من تنفيذ آخر أمر في الحلقة التكرارية `while` فيرجع مرة أخرى ليختبر الشرط فيجد قيمته واحد فيعيد تنفيذ كل أوامر `while` السابقة مرة أخرى ... وهكذا.

وبعد فهم البرنامج ندخل في خطوة المحاكاة ولكن لعمل simulation لأي مشروع ميكرو لابد أولا من عمل المشروع على برنامج الـ MikroC والذي سبق شرحه واستخراج الأخطاء فيه وعمل build له وذلك للحصول على ملف الـ hexadecimal الذي سيتم إمداد الميكرو به لعمل المحاكاة عن طريقه.

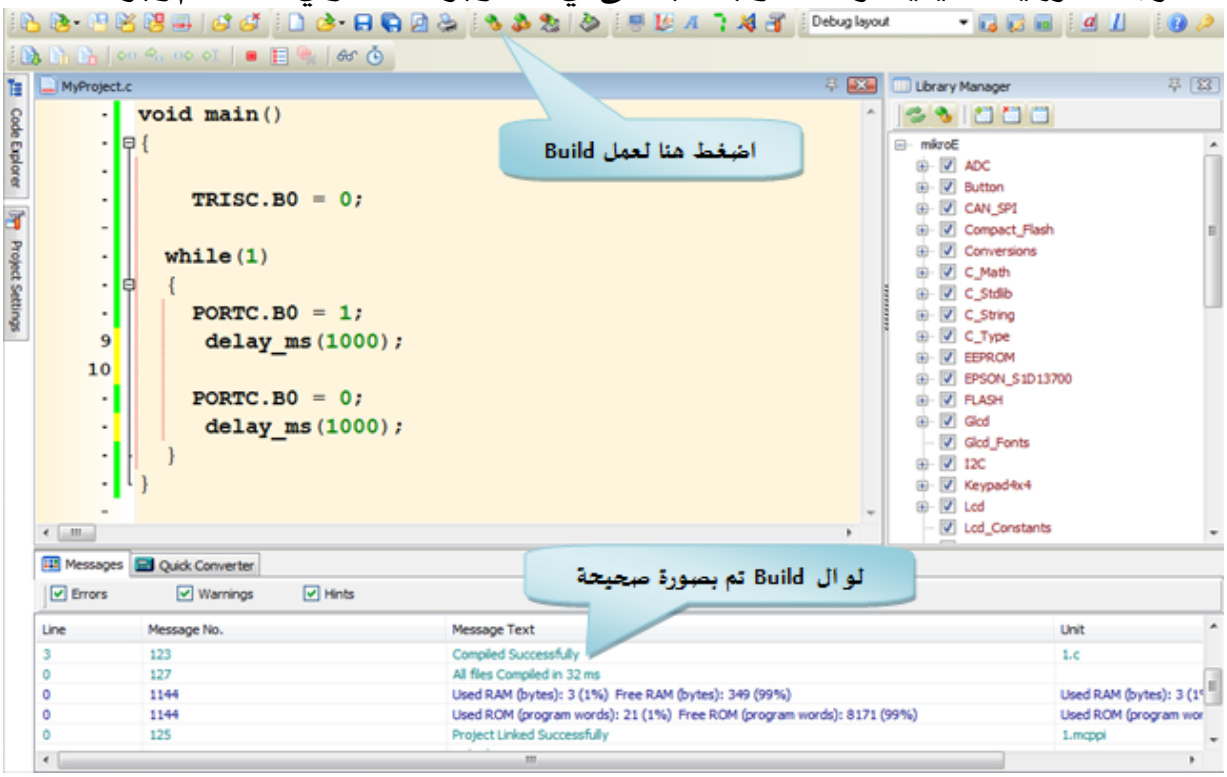
الحصول على ملف الهكسا من برنامج الميكرو سي

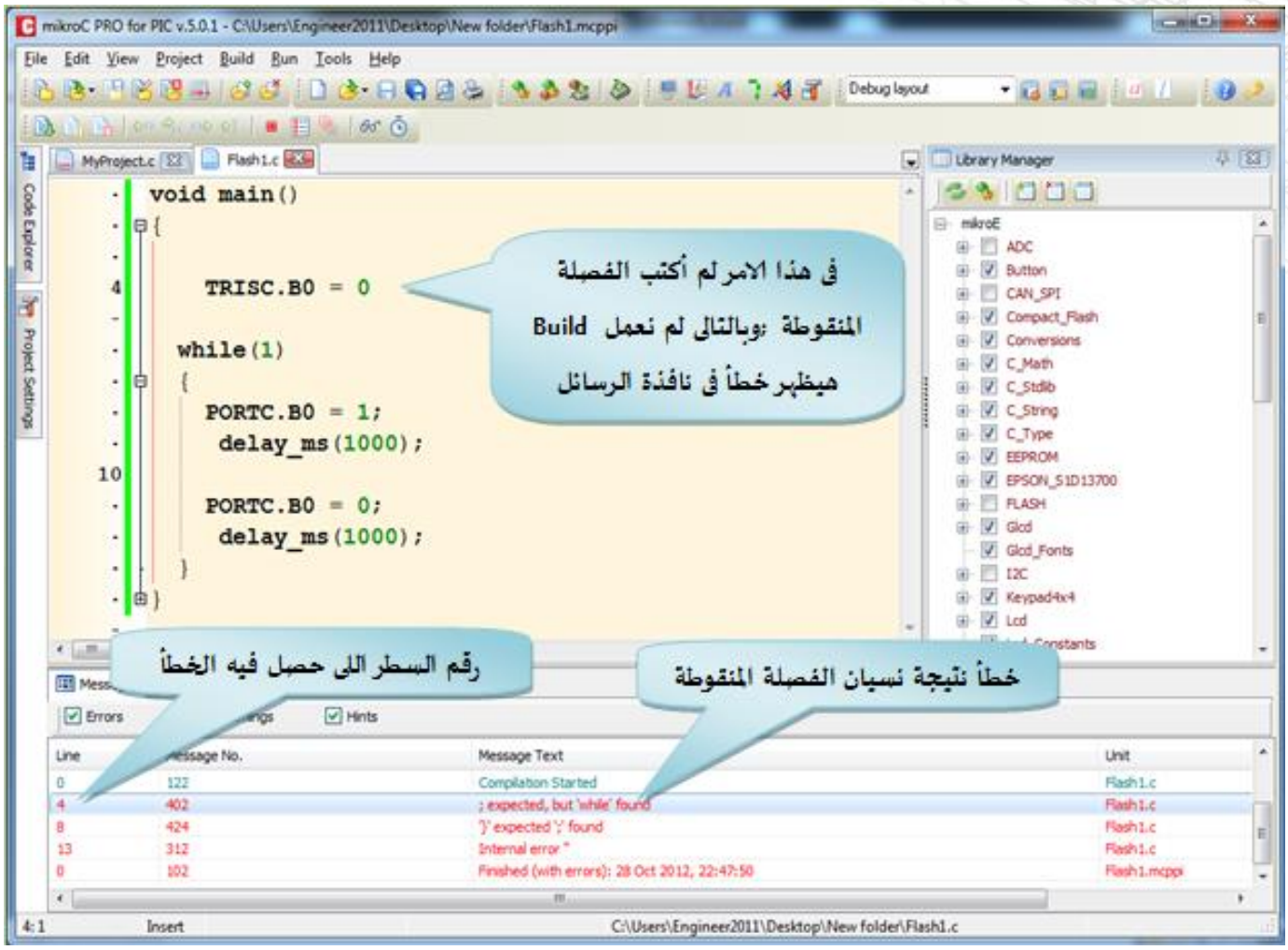
افتح برنامج الميكرو سي الذي سبق وأن قمت بتنصيبه من قبل وقم بإنشاء مشروع جديد كما تعلمت في الفصول السابقة مع مراعاة الخواص التالية للمشروع:

- اختيار نوع الميكرو 16F877A.
- اختيار تردد الكريستالة بالقيمة 8MHz واختيار نوعها HS.
- تحديد مسار سهل ومعروف يتم فيه تخزين المشروع، وذلك لاحتاجنا لهذا المسار عند عمل محاكاة للدائرة في برنامج البروتس.
- اختيار اسم مناسب لوظيفة المشروع، فمثلا في مشروعنا هذا من الممكن أن نسميه Flash_project_1.

في حال الخطأ في أحد هذه الاعتبارات عند عمل المشروع أو عند الاحتياج لتغييرها بعد عمل المشروع يمكن ذلك عن طريق اتباع أحد الطريقتين في الصورة التالية:

بعد الانتهاء من كتابة البرنامج نقوم بعمل Build للمشروع حتى يتولد ملف الـ hexadecimal المطلوب، الصورتين التاليتين توضحان واجهة البرنامج في حالة وجود أخطاء وفي حالة عدم وجودها:



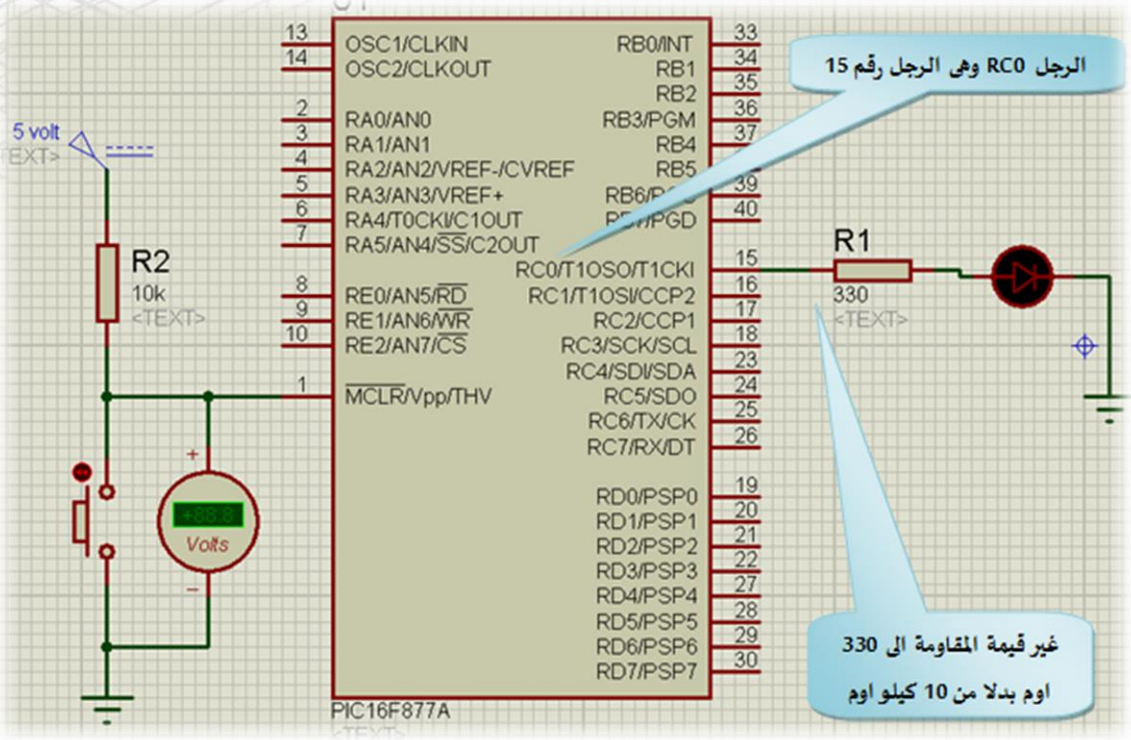


نلاحظ أن خطأ واحد أدي إلى حدوث عدة أخطاء في كذا سطر تالي ولكن عند تصحيح هذا الخطأ البسيط تمحى هذه الأخطاء المترتبة عليه أيضا.

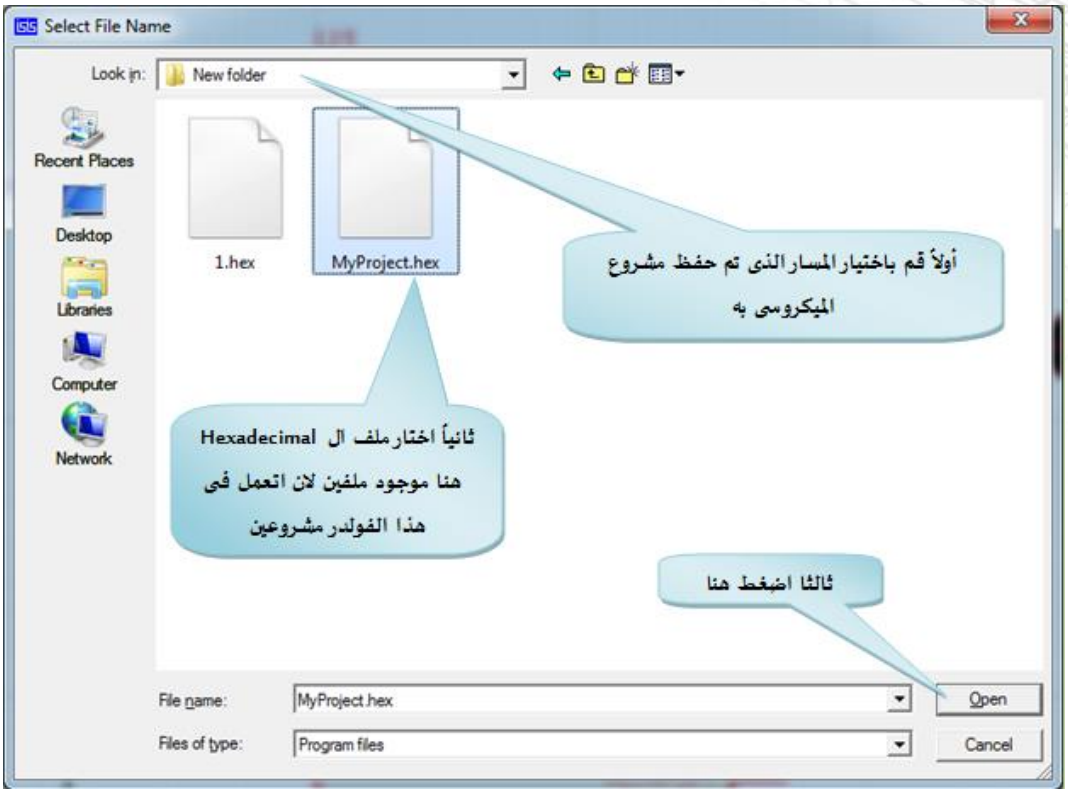
وبعد تصحيح الأخطاء في السطر المذكور أو في السطور المجاورة له ثم build مرة أخرى وعدم وجود أخطاء نكون قد حصلنا على ملف الهكسا المطلوب للمحاكاة وللحرق فيما بعد والموجود حاليا في مسار المشروع، نستطيع الآن أن نبدأ في عمل المحاكاة على بروتس.

المحاكاة

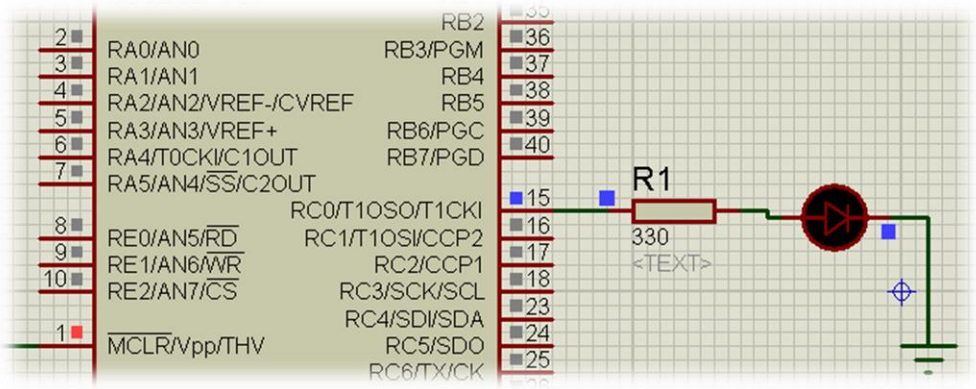
افتح مشروع جديد في بروتس ثم قم باختيار العناصر الإلكترونية المطلوبة كما تعلمنا من قبل وهي كما في الصورة التالية: الليد مع المقاومة التي تتصل معه والتي سنشرحها فائدتها الآن ودائرة الرجل MCLR المذكورة فيما سبق والميكرو مع مراعاة البحث عن الميكرو برقمه وهو 161F877A، ثم نقوم بتوصيل الدائرة كما في الشكل التالي:

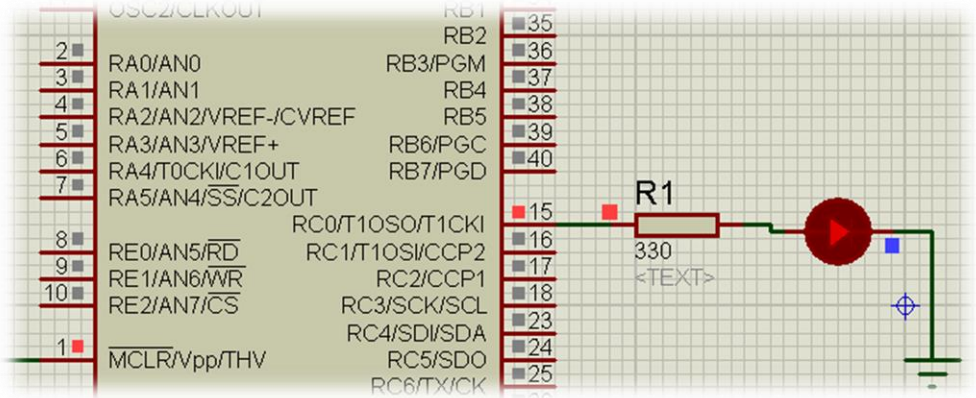


بقي خطوة إضافية وجديدة ولكن لابد منها: لكي يعمل الميكرو فلابد من وضع كود البرنامج بداخله كي ينفذه وهذا يتم بالضغط مرتين متتاليتين على الميكروكنترولر نفسه في بروتس لتظهر لك نافذة جديدة ... انظر الشكل الآتي ونفذ ما فيه:



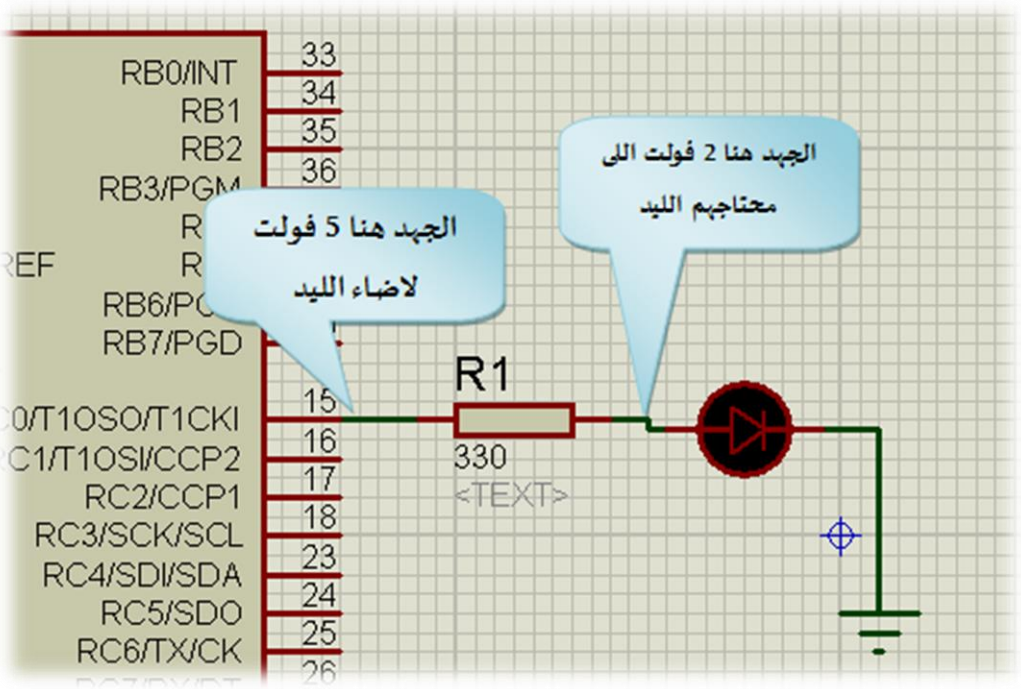
ثم قم بتشغيل المحاكاة لترى أن الليد يضئ ثانية ثم يطفئ ثانية أخرى.





ملحوظات هامة

- هناك أشياء أساسية لا يمكن للدائرة العمل بدونها ولكن يمكن للمحاكاة أن تعمل بدونها وهي:
  ١- الجهد الخمسة فولت الذي من المفترض أن يوصل على رجول الميكرو رقم ١١ و١٢، ولكن في بروتس هذا الجهد يتم توصيله تلقائيا.
  ٢- دائرة إعادة التشغيل: يمكن عدم توصيلها في بروتس ونوصلها فقط إذا كنا سنحتاج لعمل إعادة التشغيل.
  ٣- الكريستالة: يتم تحديدها يدويا كما في الخطوات السابقة ولا حاجة لتوصيلها.
- لكن عند تنفيذ الدائرة كهاردوير لابد من توصيل الثلاثة أشياء السابقة الذكر فبدون أي منهم لن تعمل الدائرة نهائيا، وبالتالي فان أي مشروع بالميكروكنترولر لابد أن يوصل بهذه الثلاثة.
- الليد يعمل على جهد من ١,٥ إلى ٢ فولت على حسب نوعه ومادته ولكن الميكرو يخرج ٥ فولت، وغالبا ما تكون قيمة التيار الذي يسحبه حوالي ١٠ ميللي أمبير ولكن الميكرو يخرج ٢٥ ميللي أمبير، هل هذا التفاوت في القيم يسبب مشاكل؟ ... بالنسبة للتيار لا توجد مشكلة لأن الليد يسحب التيار الذي يريده فقط أما الجهد فهو كبير ويلزم تقليله ولعمل ذلك تظهر فائدة المقاومة الموضوعة بين الميكرو والليد وهو المقاومة التي قيمتها ٣٣٠ أوم الموضحة في الدائرة والتي تستخدم لعمل ما يسمى voltage drop، ولكن لماذا القيمة ٣٣٠ أوم بالذات؟؟



- من الشكل السابق يمكن الحصول على قيمة المقاومة من العلاقة:

$$R = \frac{\text{Voltage}}{\text{Current}} = \frac{5-2}{10\text{mA}} = \frac{5-2}{10 * 10^{-3}} \cong 300\ \Omega$$

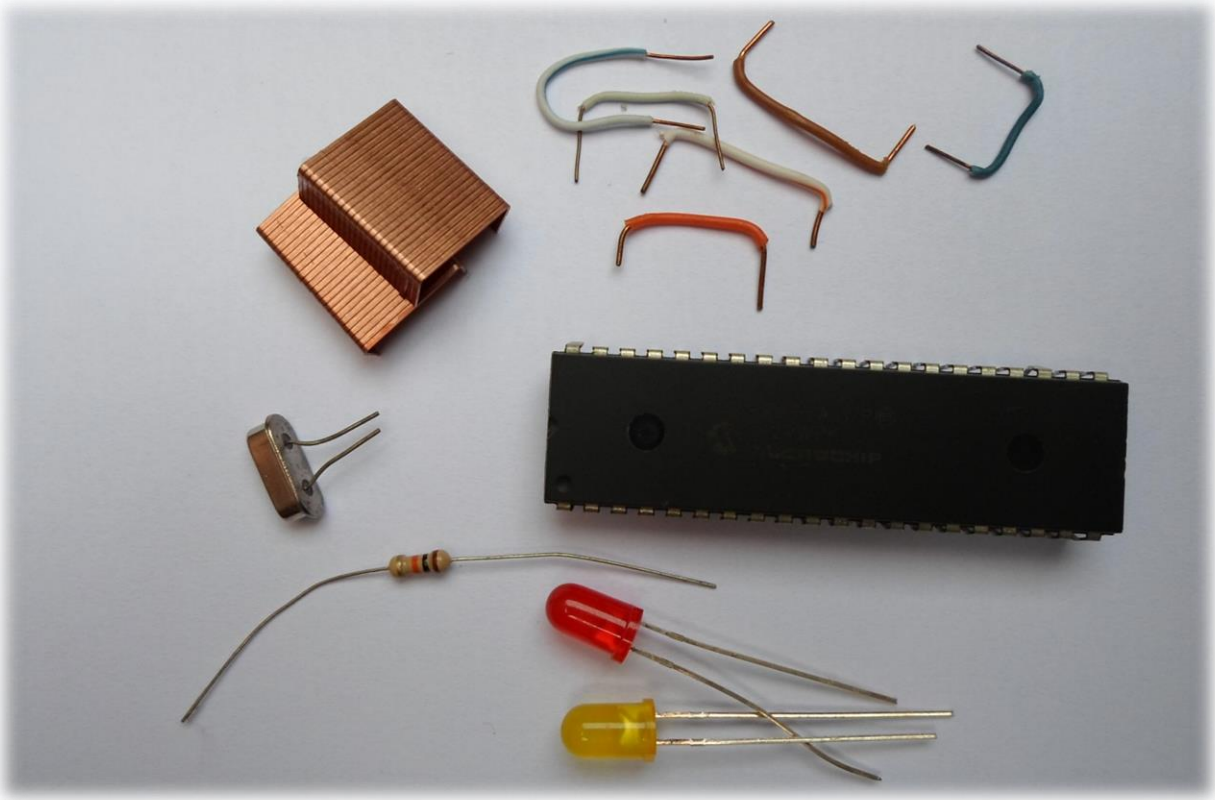
- وحيث أنه لا يوجد مقاومة متداولة بهذه القيمة في الأسواق، يمكن الاستعاضة عنها بالقيم القريبة منها فيمكن اختيار القيمة ٣٣٠ أو ٢٢٠ لان هذان هما القيمتان المتاحتان والقريبتان من القيمة ٣٠٠.
- لو لم تضع المقاومة في توصيل الهاردوير وقمت بتوصيل الليد مباشرة على رجل الميكروكنترولر فان الليد سيضيئ لكن عمره الافتراضي سيقل.
- لو تركت المقاومة بالقيمة ١٠ كيلو أوم فان الليد لن يضئ.

توصيل الدائرة كهاردوير

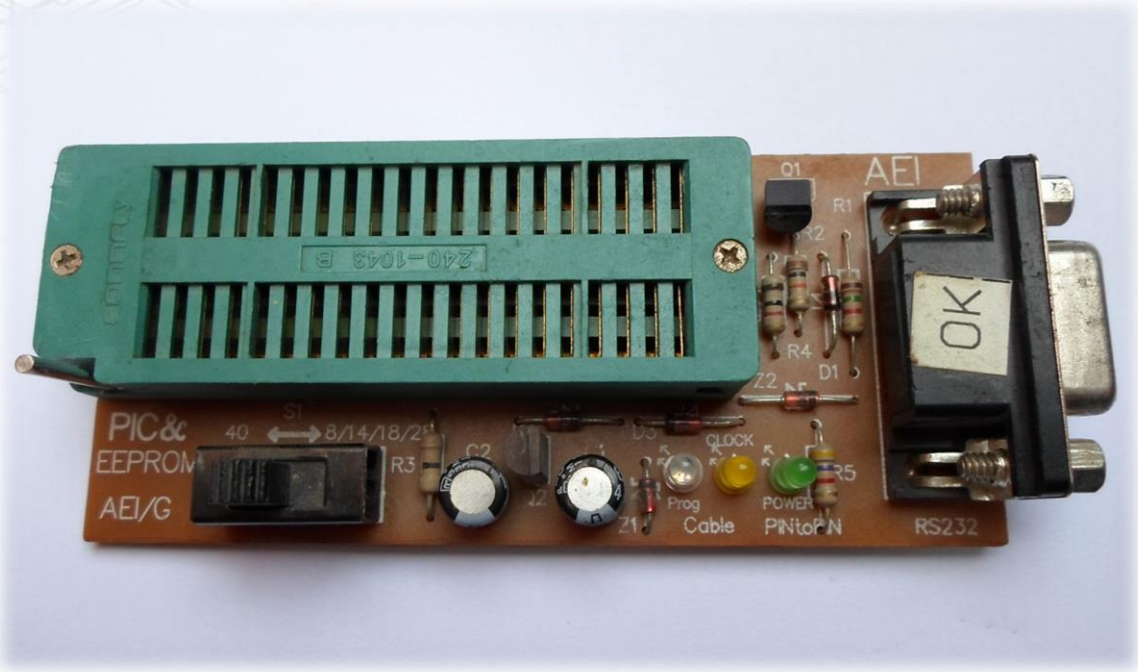
أولا: المكونات التي سنحتاجها في المشروع:

المكونات الأساسية

الميكروكنترولر والكريستالة والليد والمقاومات وبعض أسلاك التوصيل



البروجرامر وكابل السريال الخاص به

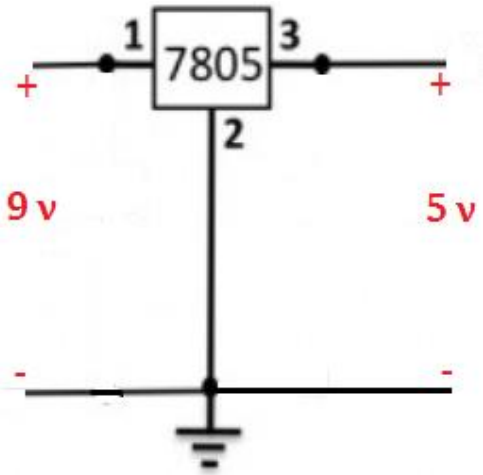


مصدر جهد (بطارية ٩ فولت)

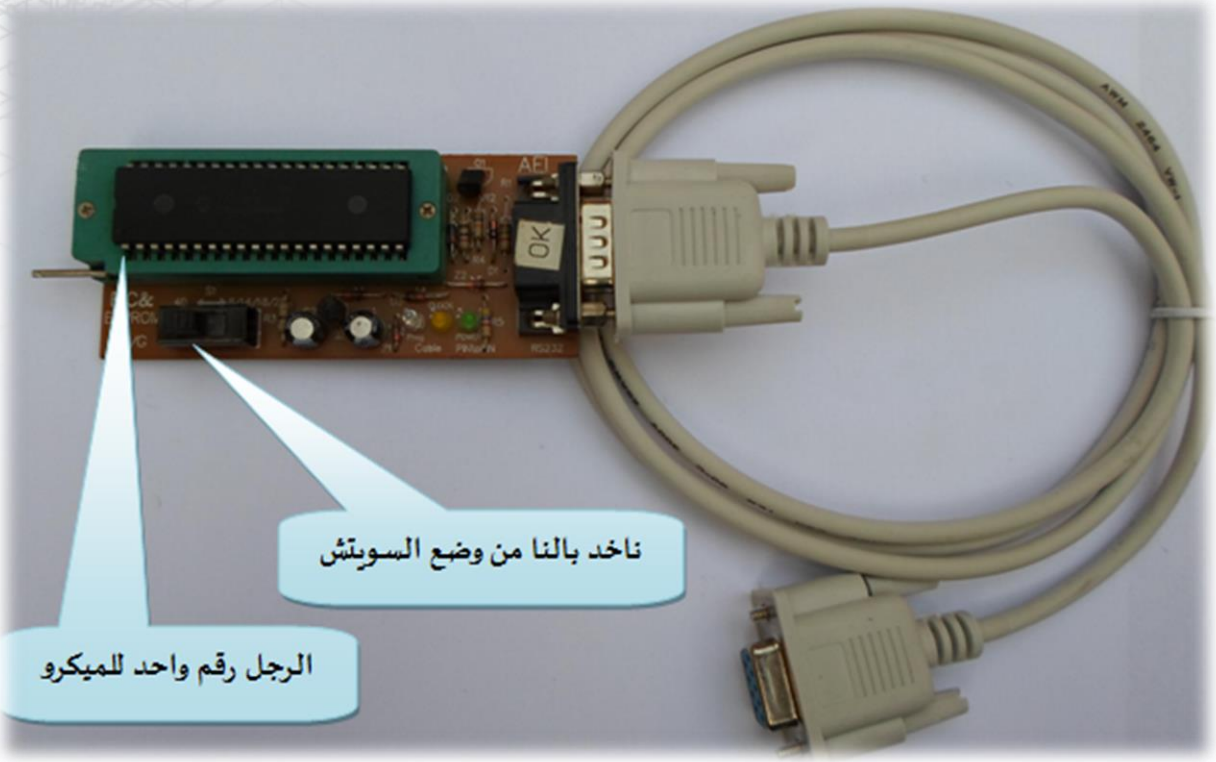
ملحوظة هامة: الميكرو يحتاج خمسة فولت لكي يعمل لكن معظم البطاريات أو مصادر الجهد تكون مثلا ٩ فولت أو ١٢ فولت ... فما الحل ؟؟؟؟

يستخدم IC يقوم بتحويل هذه القيم إلى ٥ فولت لتناسب الميكرو، يحمل مثل هذا الـ IC الرقم 7805 ويكون دخله هو الجهد الأكبر من خمسة فولت بينما الخرج منه هو خمسة فولت ...

والسؤال هنا: كيف قام هذا الـ IC بالتخلص من الجهد الزائد وقانون بقاء الطاقة يقول أن الطاقة لا تفنى ولا تستحدث من العدم ؟؟؟ يخرج الجهد الزائد في شكل حرارة منه، وفيما يلي دائرته المبسطة ويمكنك الاطلاع على دوائر أخرى له من خلال الإنترنت:

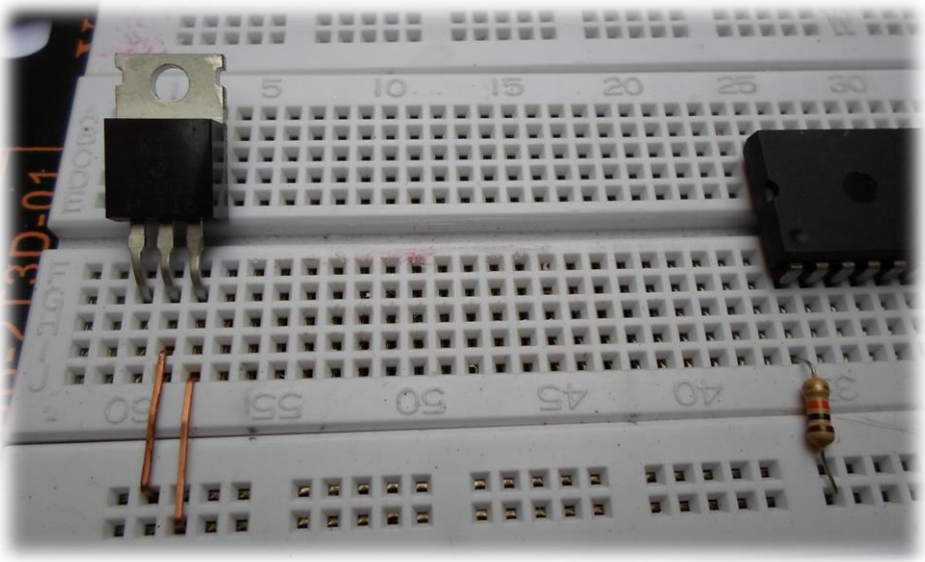


ثانيا: توصيل الميكرو بالبروجرامر لإتمام عملية الحرق:



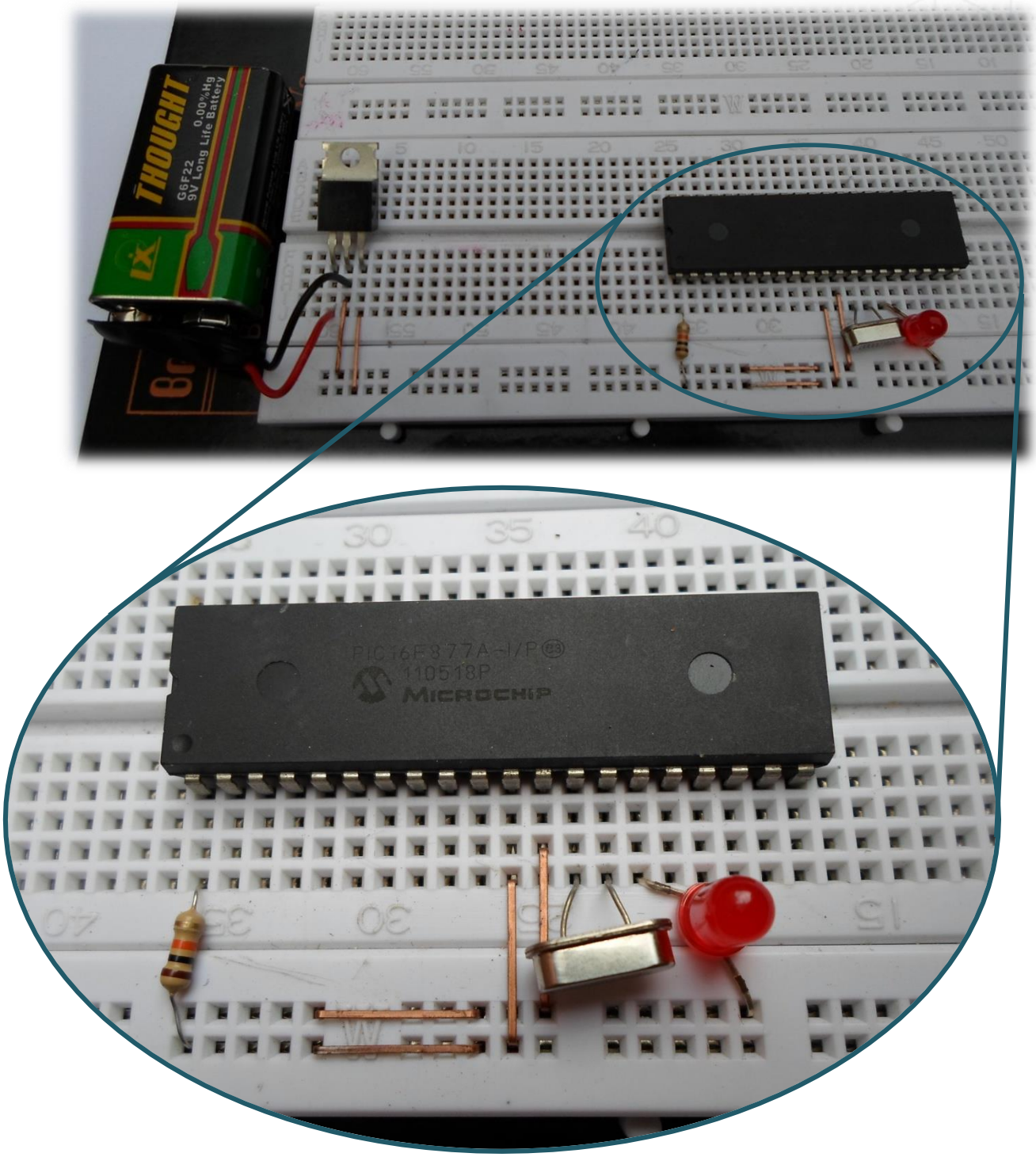
ثم نقوم بتوصيل الطرف الآخر للكابل بفتحة السريال في الكمبيوتر ومن ثم استخدام برنامج winpic800 كما سبق شرحه فيما سبق.

ثالثا: توصيل الـ IC 7805:

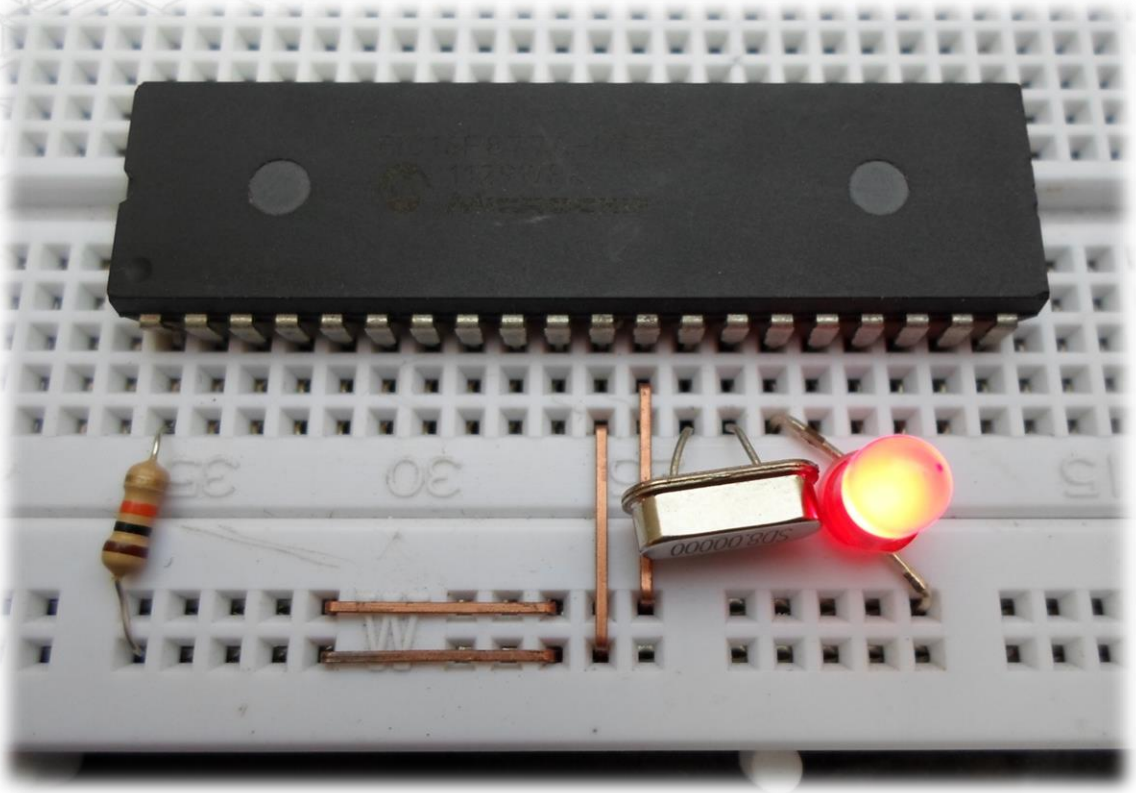


رابعا: توصيل المكونات الأساسية

توصيل المكونات اللازمة لتشغيل الميكرو وهي الكريستالة وتوصل على الرجلين ١٣ و١٤ والجهد ويوصل على الرجلين ١١ و١٢ ودائرة إعادة التشغيل على الرجل ١ والتي بدونها لن يعمل الميكرو كما ذكرنا من قبل ... ثم توصيل الليد ليصبح الشكل النهائي للمشروع كالآتي



وعند تشغيل المشروع:



إضافة مفتاح لبدء التشغيل

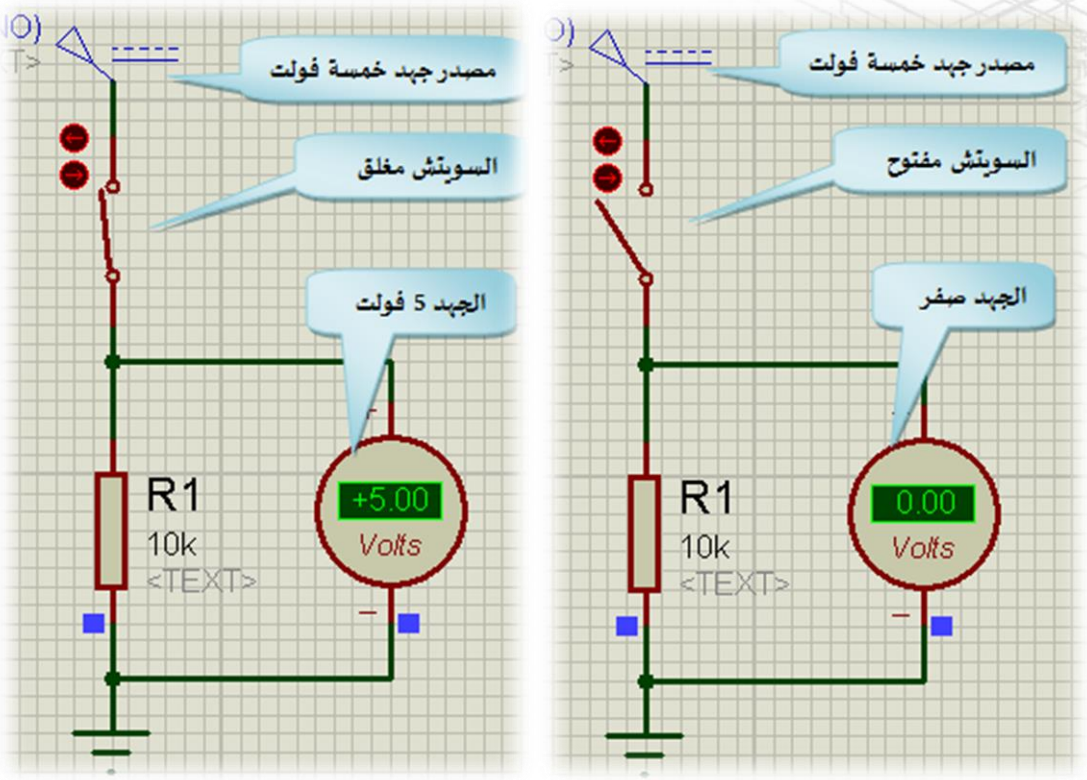
نفس المشروع الماضي ولكن المشروع الماضي يبدأ التنفيذ بمجرد توصيل الجهد ولكننا نريد إضافة مفتاح يتم الضغط عليه لبدء تنفيذ البرنامج ولا يبدأ التنفيذ قبل الضغط عليه.

ولترجمة وتنفيذ هذا المطلب سنجد أننا نريد من الميكرو أن يبدأ باختبار حالة السويتش ولا يفعل شيئا إذا كان مفتوحا حتى يغلق وهنا يبدأ في تنفيذ الأوامر السابقة، وبالتالي سيكون هناك أضافة في البرنامج تقوم بضمان ذلك

وحيث أنه تم إضافة سويتش كدخل للميكرو فلابد من استخدام رجل إضافية من رجول الميكرو لتوصيل السويتش، وهذه الرجل ستعمل كدخل، فلنفترض مثلا أن هذه الرجل هي RB0 وبالتالي لنجعلها تعمل كدخل يتم إضافة الأمر التالي:

```
TRISB.B0 = 1;
```

ولكن كيف نوصل السويتش على رجل من رجول الميكرو ؟؟ الصور التالية توضح الدائرة المستخدمة في ذلك ...



وهذا هو الشكل النهائي للبرنامج بعد التعديلات:

```
void main()
{
    TRISB.B0 = 1;
    TRISC.B0 = 0;

    while(1)
    {
        if(PORTB.B0 == 1)
        {
            PORTC.B0 = 1;
            delay_ms(1000);

            PORTC.B0 = 0;
            delay_ms(1000);
        }
    }
}
```

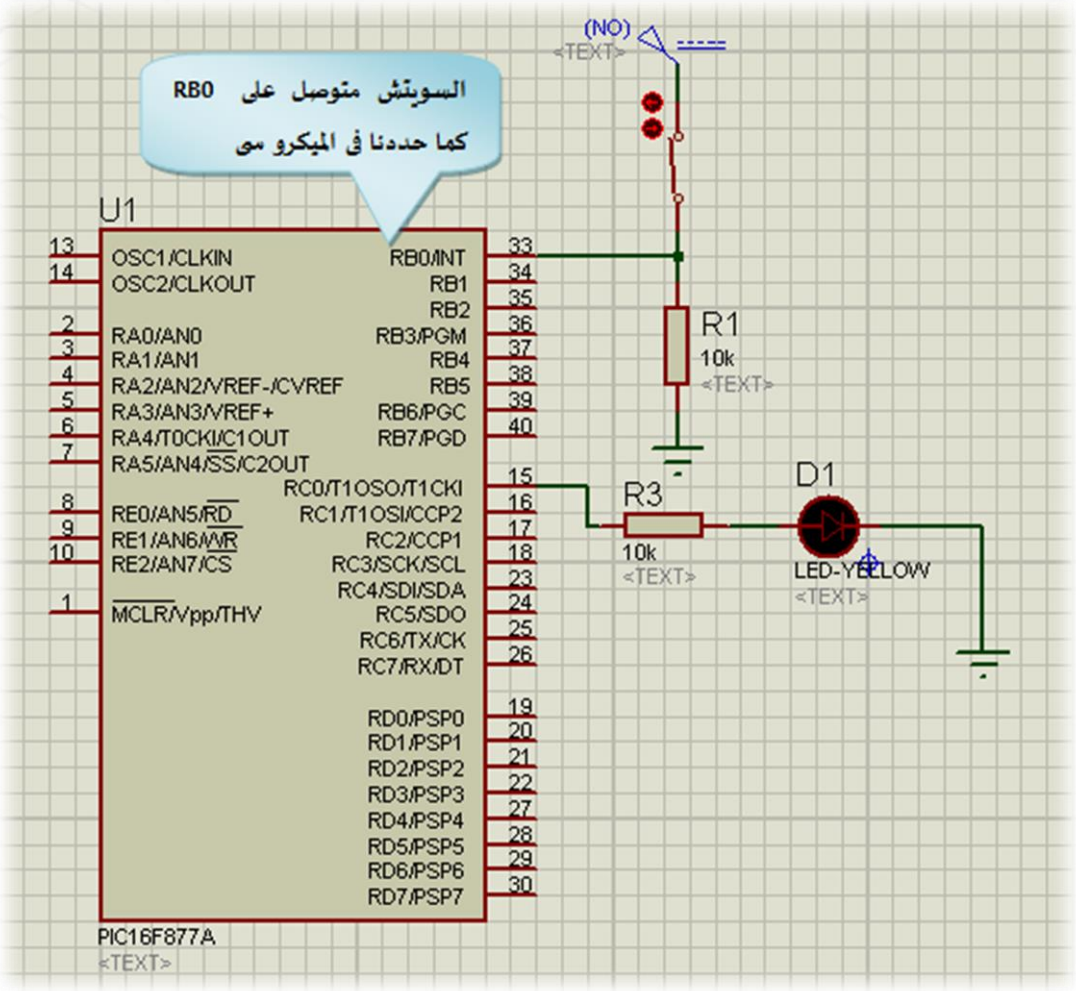
لجعل RB0 يعمل كدخل لان متوصل عليه سويتش

لجعل RB0 يعمل كخرج لان متوصل عليه ليد

لقراءة الرجل اللى متوصل عليها السويتش واختبارها لو كانت بواحد (أى خمسة فولت معناه السويتش مغلق) يبقى الشرط متحقق ينفذ اللى داخل ifبمعنى انه هينفذ اوامر الفلاش ولو لم يكن السويتش مغلق فلن ينفذ أوامر الفلاش

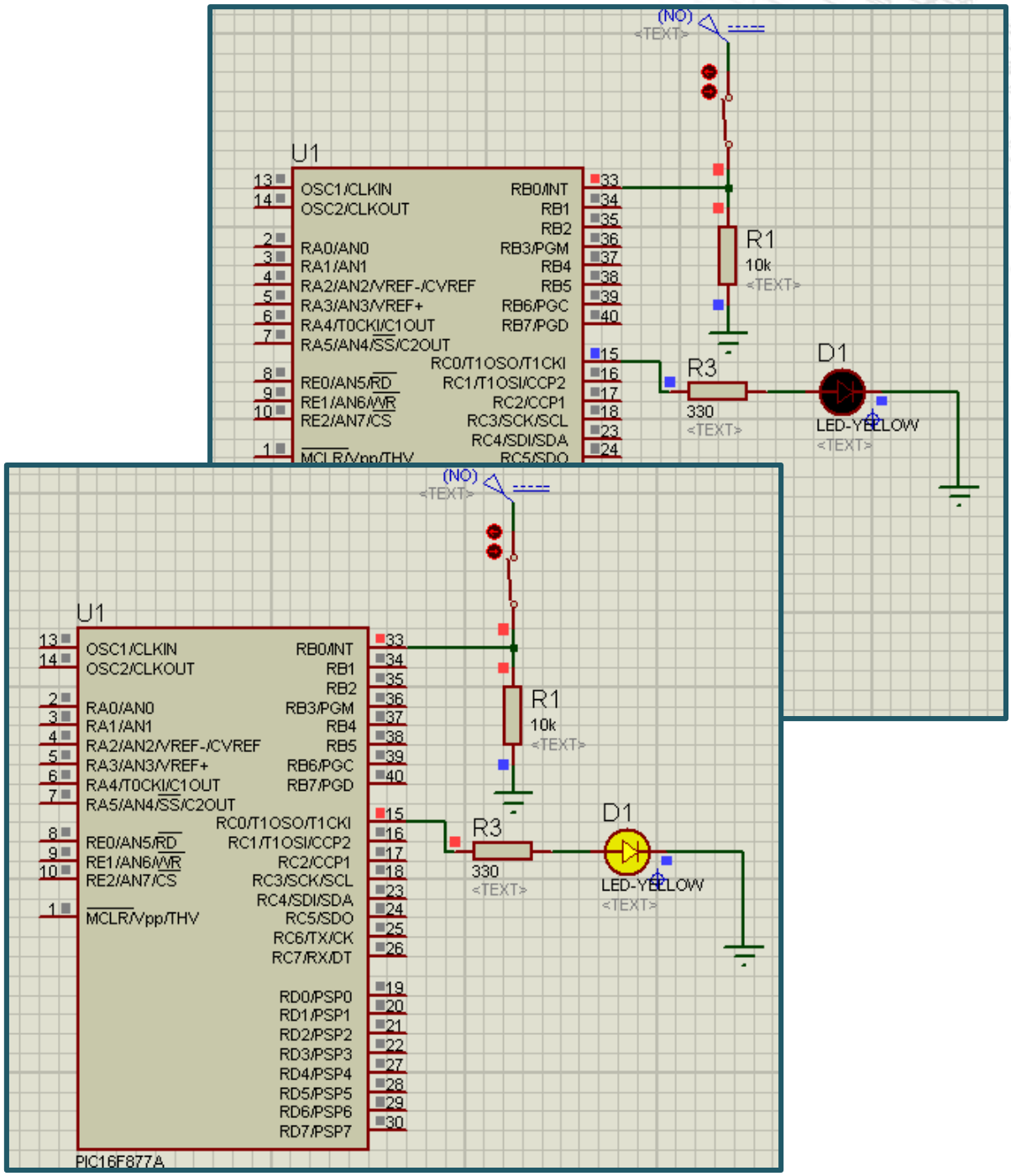
ثم قم بعمل Build للبرنامج ثم ارسم الدائرة على بروتس وحمل البرنامج على الميكرو وابدأ المحاكاة

وكما نوهنا من قبل فإن دائرة إعادة التشغيل يمكن الاستغناء عنها في بروتس أما في الهاردوير فلا يمكن.



ولكني عندما بدأت المحاكاة كما في الصورة لم يبدأ الليد في الإضاءة بالرغم من كون السويتش مغلق ... دعنا نفكر ما هو السبب؟؟ ... لا تتسرع ... انظر في الصورة وتفكر ولاحظ ما هي المشكلة المانعة لبدء عمل المشروع ... لا تيأس سريعا وتنظر للإجابة ... اختبر نفسك وعودها على اكتشاف المشكلات ...

المشكلة في الصورة السابقة تتلخص بكل بساطة في أن المقاومة ما زالت قيمتها ١٠ كيلو أوم ولم نقم بتغييرها إلى ٣٣٠ أوم، فنقوم بتغييرها وتشغيل المحاكاة فيبدأ البرنامج في العمل كما يلي:



تكرار الفلاش عدد محدد من المرات

بمعنى أن الليد يضئ ويطفئ عدد محدد من المرات وليكن عدد ٥ مرات فقط وليس عدد لا نهائي كما في البرنامج الأصلي

وحيث أننا لا نريد عدد لا نهائي فلا حاجة الآن للحلقة while(1) ويمكن استخدام حلقة for ليكون البرنامج على الشكل التالي:

```
int i;
void main()
{

   TRISC.B0 = 0;

   for(i=1;i<=5;i++)
   {
     PORTC.B0 = 1;
     delay_ms(1000);

     PORTC.B0 = 0;
     delay_ms(1000);
   }
}
```

ويمكنك تجربته لترى كم من المرات سينفذ الميكرو أوامر الفلاش ...

مشروع إشارة المرور

هو مشروع لمحاكاة إشارة المرور حيث يوجد ثلاثة ليدات بالألوان الأحمر والأخضر والأصفر والمطلوب:

١) إضاءة الليد الأحمر لمدة ثانية في حين أن الليدين الآخرين غير مضاءين.

٢) إضاءة الليد الأصفر لمدة ثانية في حين أن الليدين الآخرين غير مضاءين.

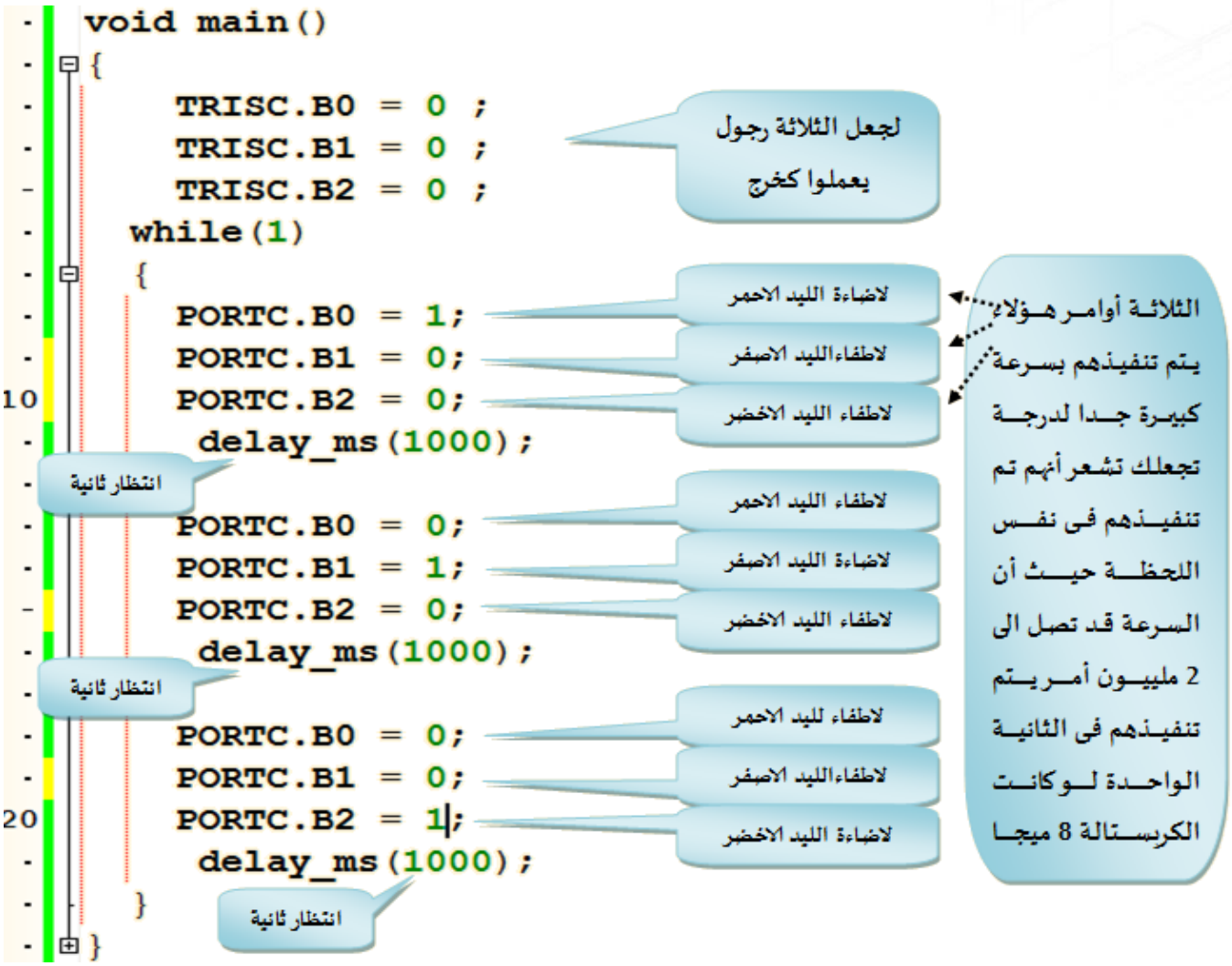
٣) إضاءة الليد الأخضر لمدة ثانية في حين أن الليدين الآخرين غير مضاءين.

نقوم بتحديد الرجول المطلوبة كدخل والمطلوبة كخرج، سنحتاج ثلاثة رجول كخرج من أجل الليدات الثلاثة، ولنفترض أن الليدات الثلاثة سنقوم بتوصيلهم على كل من RC0، RC1، RC2 كما أنه لا يوجد خرج.

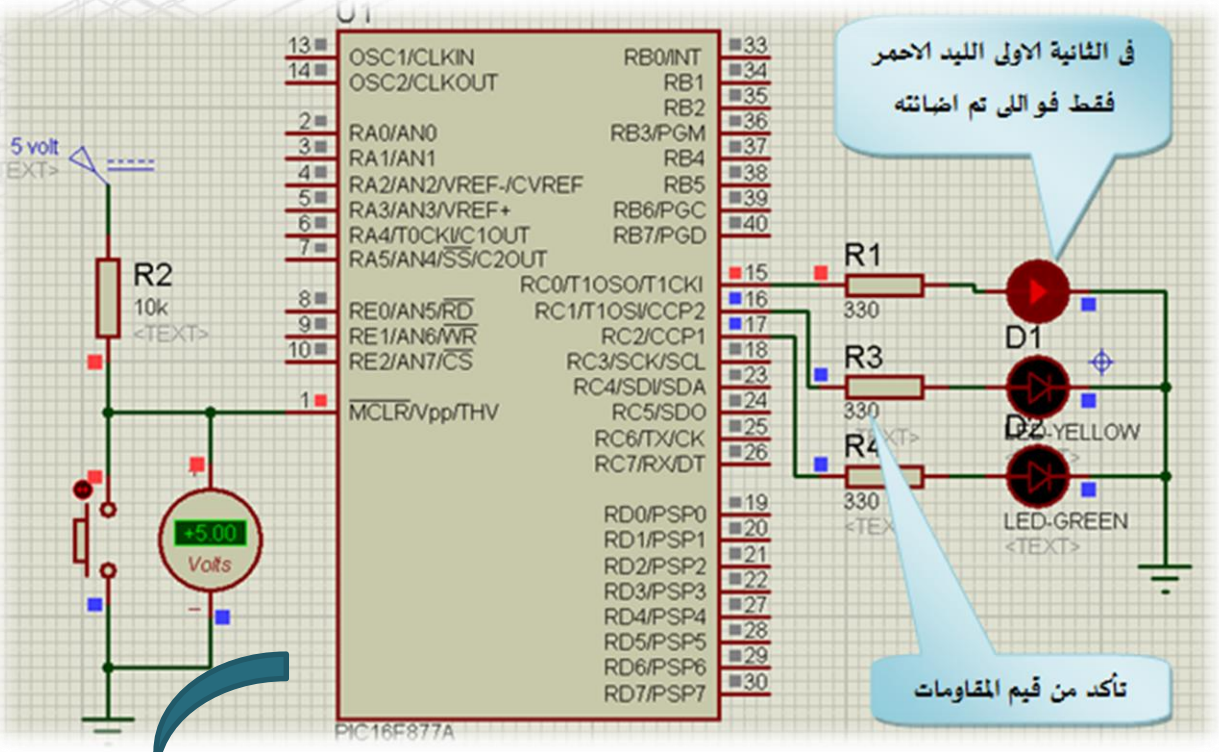
وبالتالي نقوم بكتابة أوامر تحديد الاتجاه بحيث يكون الثلاثة رجول كخرج.

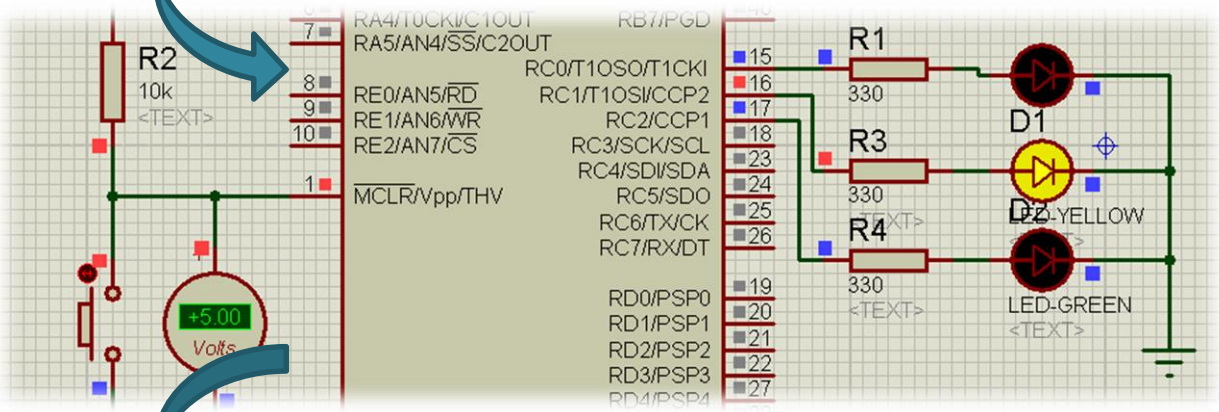
```
TRISC.B0 = 0;
TRISC.B1 = 0;
TRISC.B2 = 0;
```

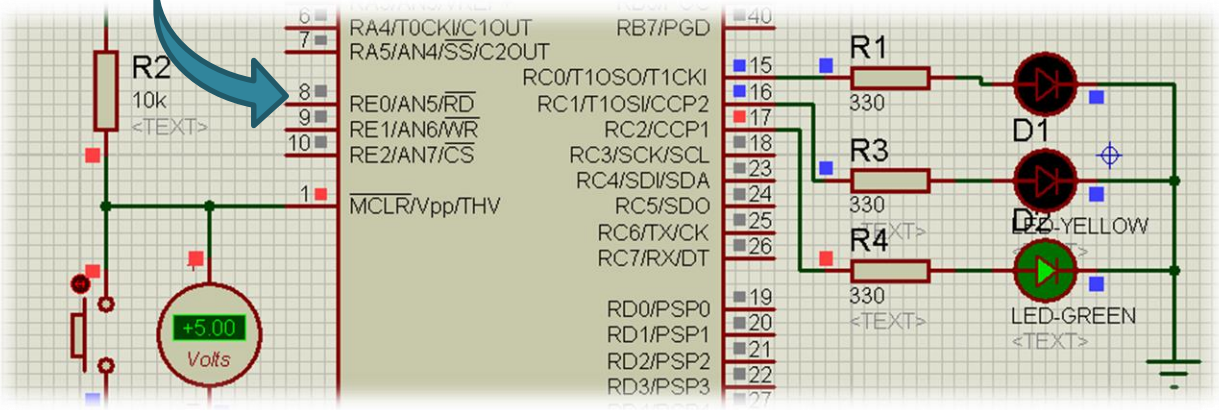
ثم كتابة الأوامر التي تنفذ الخطوات ١ و٢ و٣ المذكورين سابقا داخل حلقة while ليصبح شكل البرنامج كالآتي:



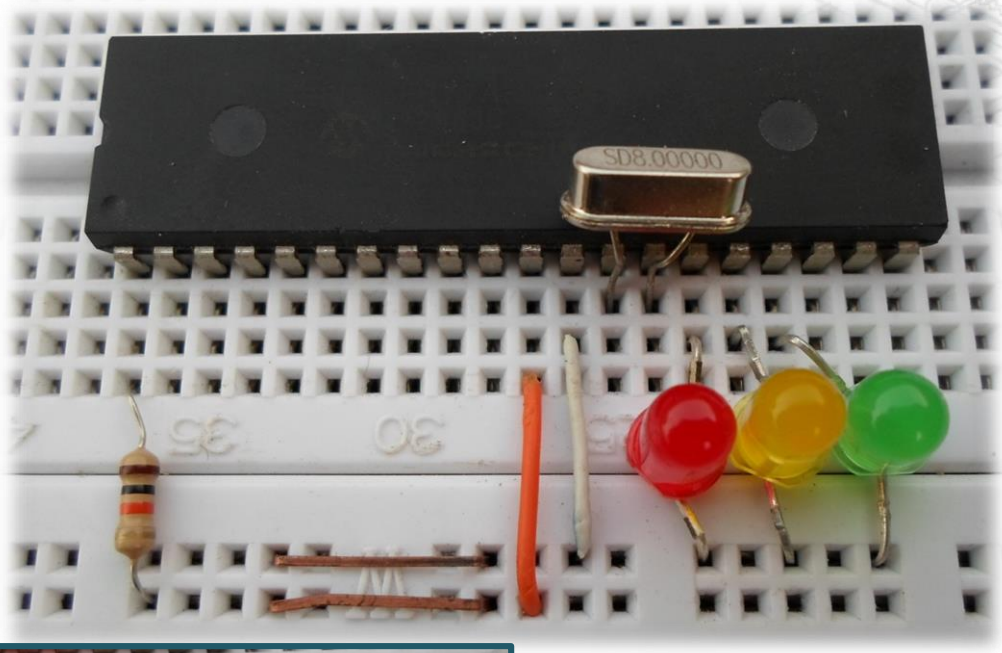
وعند عمل مشروع المحاكاة على البروتس بطريقة مماثلة للمشروع السابق وكما تم شرحه في الفصول السابقة كما في الشكل التالي ثم تشغيل المحاكاة يكون نتيجة المحاكاة هي تغيير الإضاءات كما يلي:

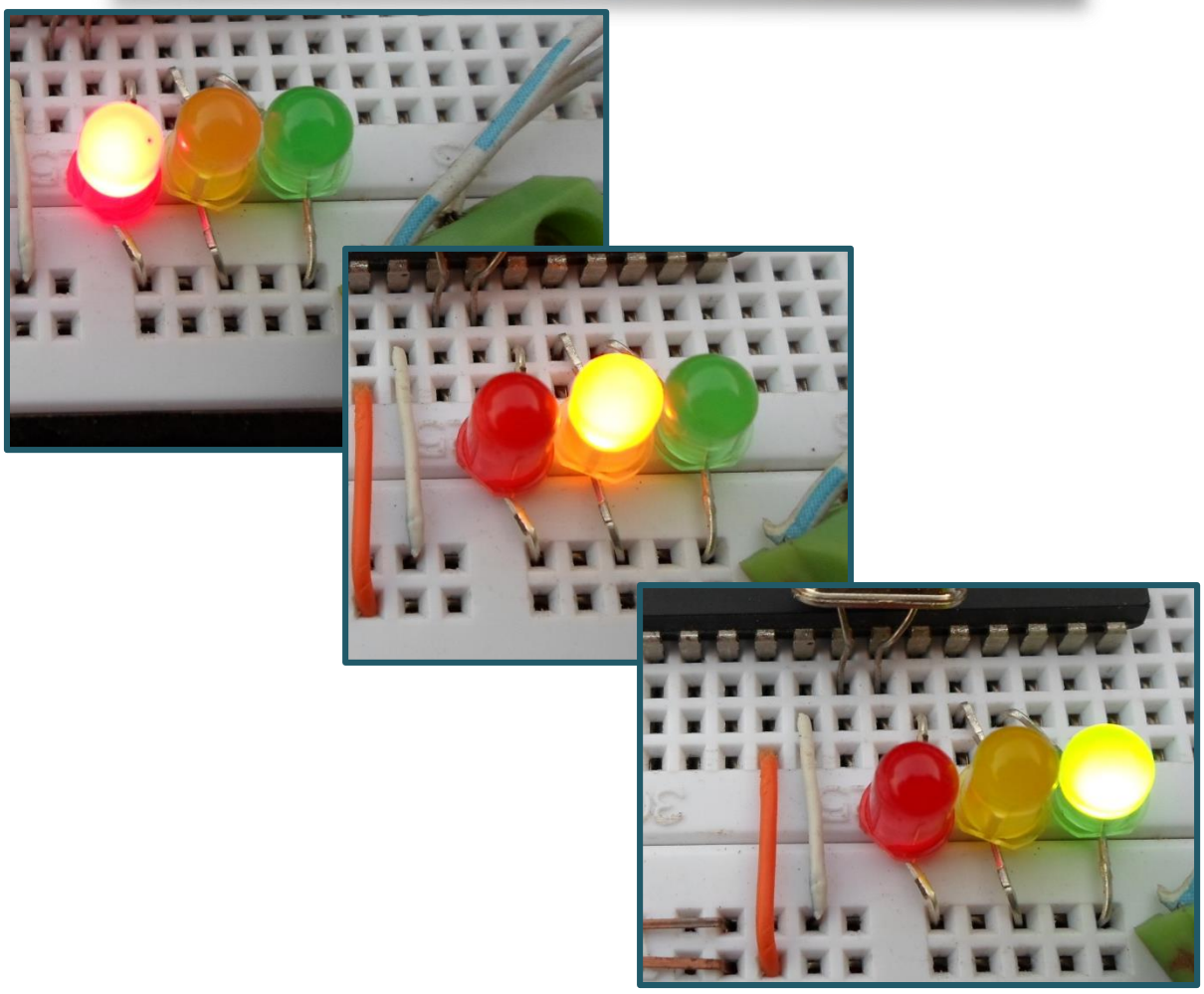






وعلى مستوى الهاردوير:





مشروع العداد الثنائي Binary Counter

وتتلخص فكرة هذا المشروع في استخدام مخرج معين من مخارج الميكرو وليكن مثلا المخرج B في إخراج قيم بالنظام الثنائي، وحيث أن المخرج عبارة عن ثمانية رجول فان مجال القيم المتاحة للخروج عليه بالنظام الثنائي هو من صفر إلى ٢٥٥ بدون الدخول في تفاصيل هذا النظام، ولكن من يعلم القليل منها يعرف لماذا هذه القيم بالذات وكيف تم حسابها.

وهنا سنتعرض لبعض هذه التفاصيل: فإن التمثيل الثنائي للصفر في ٨ خانات هو 00000000 وبالتالي فإننا عند كتابة أمر كهذا PORTB = 0; فان كل رجول هذه المخرج سيخرج عليها صفر فولت لان هذا الأمر يكافئ PORTB = 0B00000000; وعند كتابة الأمر الآتي PORTB = 1; فهذا معناه أن الرجل RB0 هي فقط ما سيخرج عليها خمسة فولت والباقي صفر لان هذا الأمر يكافئ PORTB = 0B00000001; ولو كتبنا الأمر PORTB = 2; فهذا معناه أن الرجل RB1 هي فقط التي سيخرج عليها خمسة فولت وصفر فولت على الباقي لأن هذا الأمر يكافئ بالثنائي PORTB = 0B00000010; ... وهكذا حتى نصل إلى القيمة ٢٥٥ والتي تكافئ بالثنائي 11111111 وعندها يكون خرج كل رجول المخرج B يساوي خمسة فولت ..

لاحظنا في الشرح أنه في كل خطوة نقوم بزيادة واحد على القيمة بداية من الصفر مرورا بالواحد ثم الاثنين والثلاثة ... وهكذا حتى نصل ٢٥٥، فهل هذا يعنى أننا سنكتب ٢٥٥ أمر لكي نخرج هذه القيم واحدا بعد الآخر على المخرج B؟؟؟ بالطبع لا بل هو أمر واحد لكن كيف ذلك ... دعنا ننظر إلى البرنامج التالي:

```
void main()
{
    TRISB = 0B00000000;

    PORTB = 0B00000000;
    delay_ms(2000);

  while(1)
  {
    PORTB = PORTB + 1;

    delay_ms(2000);
  }
}
```

هذا الامر يجعل كل رجول المخرج B تعمل كخرج

هذا الامر يقوم باخراج صفر على كل رجول المخرج B

انتظر ثانيتين لكى نرى الصفر الذة خرج على المخرج B

هذا الامر يقوم بتزويد المخرج B كل مرة بواحد

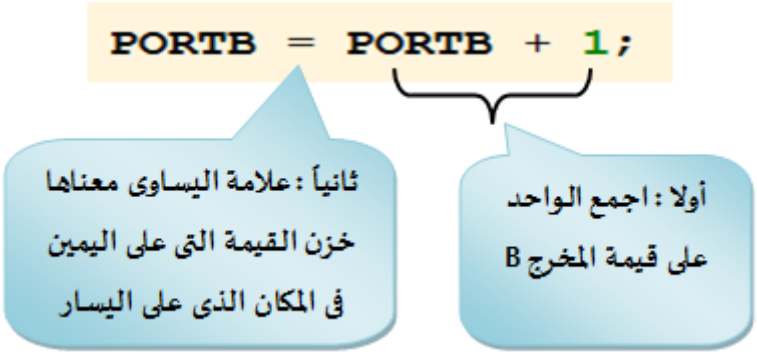
بعد التزويد بواحد فى الامر الماضى يتم الانتظار ثانيتين

عندما يصل الميكرو إلى الأمر:

```
PORTB = 0B00000000;
```

فإنه يقوم بإخراج صفر على كل رجول مخرج الميكروكنترولر ثم ينتظر ثانيتين بناء على الأمر التالي وذلك حتى نتمكن من ملاحظة القيمة صفر اذ أن الأوامر تنفذ بسرعة كبيرة جدا.

ثم يقوم بالانتقال إلى حلقة الـ while ويختبر شرطها فيجد قيمته بواحد فينفذ ما بداخلها وأول الأوامر كالآتي:



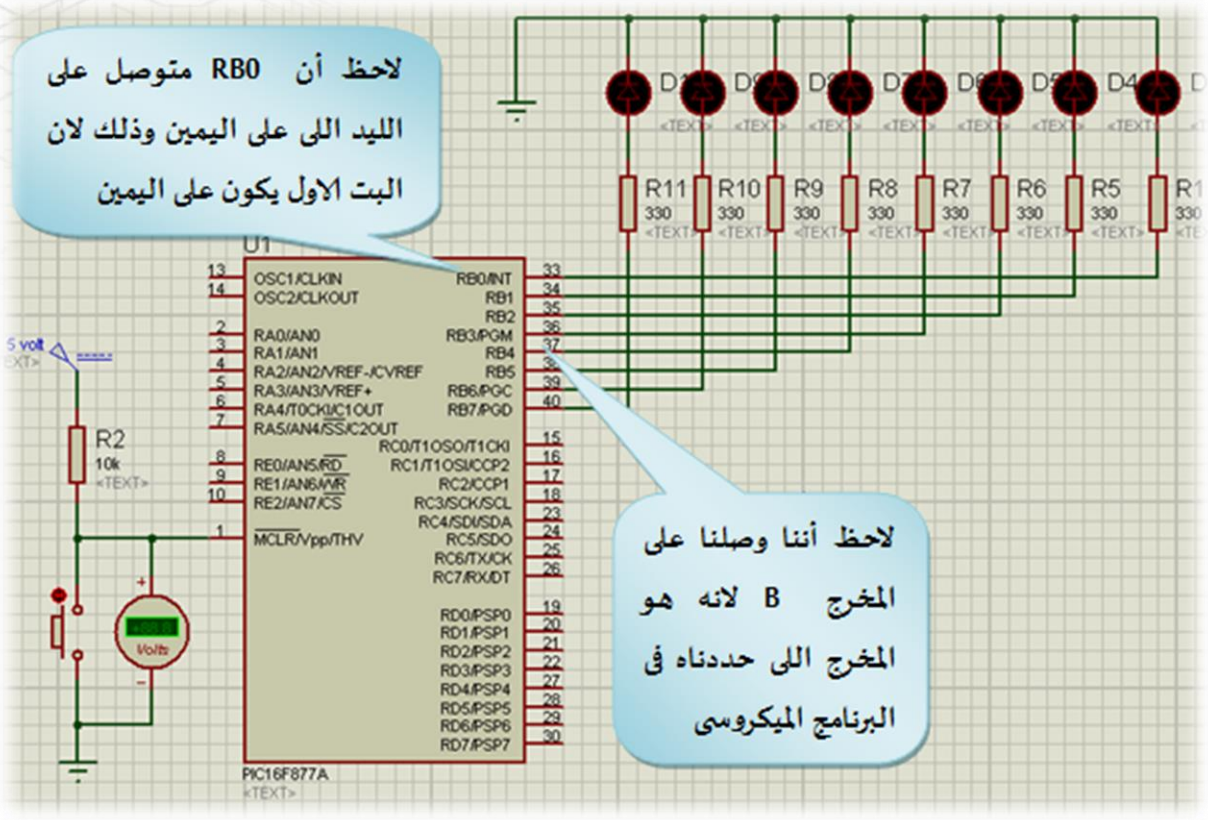
والذي يتم حساب ما بالجهة اليمنى أولا ثم تخصيص الناتج للجهة اليسرى وهذت يعني إضافة واحد إلى قيمة الـ PORTB ثم إخراج ناتج الجمع ثانية على نفس المخرج، وحيث أن قيمة المخرج كانت بصفر فبزيادة واحدة تكون قيمته بواحد التي تنعكس على الخرج، ثم يأتي الأمر التالي وهو الانتظار ثانيتين لكى نستطيع رؤية القيمة واحد أيضا، ثم تنتهى while فيعود للبداية لاختبار الشرط مرة أخرى فيجد قيمته بواحد فيدخل لينفذ ما بداخل الـ while مرة ثانية، وبنفس الطريقة ستخرج القيمة ٢ على المخرج B ثم ينتظر ثانيتين ... وهكذا، وبالتالي إجمالا فان هذا البرنامج يقوم بإخراج القيم الثنائية من صفر إلى ٢٥٥ على المخرج B.

ملحوظة: القيمة ستزداد باستمرار بداية من صفر إلى ٢٥٥ ولكن ماذا سيحدث بعد ذلك؟؟ سيتم إضافة واحد كما هو واضح في الأمر ولكن هذه الإضافة ستجعل القيمة ٢٥٦ وهوا ما لا يمكن عرضة في ٨ خانات فقط وبالتالي سيعود العداد إلى الصفر ثانية حيث أن التمثيل الثنائي للقيمة ٢٥٦ يتضمن ٨ أصفار ثم واحد وهو ما يتم إهماله ولا يظهر سوى الأصفار ... فيبدأ العداد من جديد

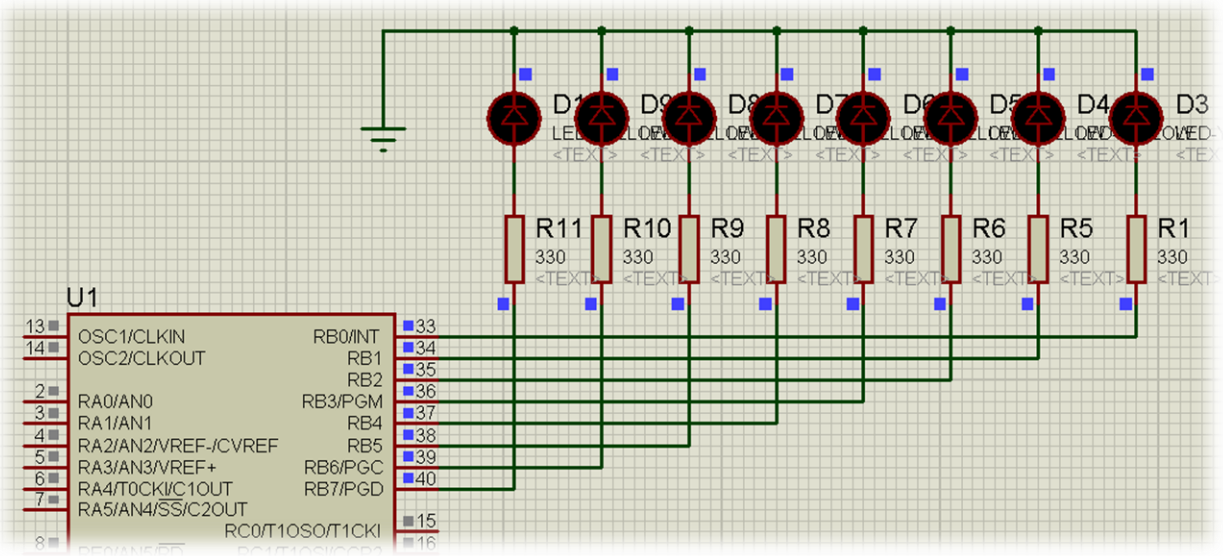
بقيت مشكلة بسيطة وهي كيفية قراءة هذه القيم الثنائية التي ستخرج على المخرج B ؟؟؟ لذلك سنقوم بوضع ليدات على رجول هذا المخرج وإذا كان الليد مضيء فهذا يعني أن الرجل المتصل بها الليد قيمتها تساوى واحد ولو الليد غير مضيء فهذا معناه أن قيمة الرجل تساوى صفر.

المحاكاة

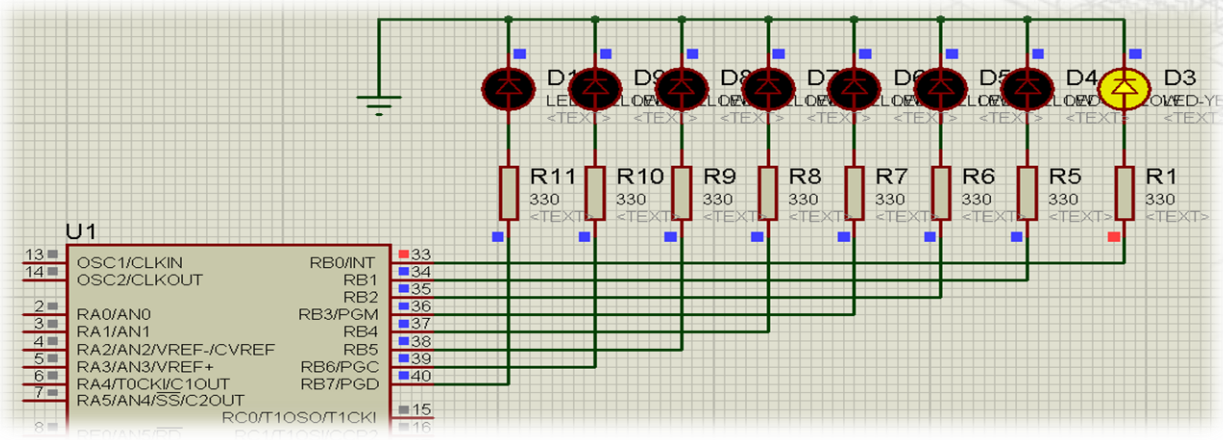
الخطوة التالية المعتادة هي رسم الدائرة على بروتس وتشغيلها كما في الشكل:



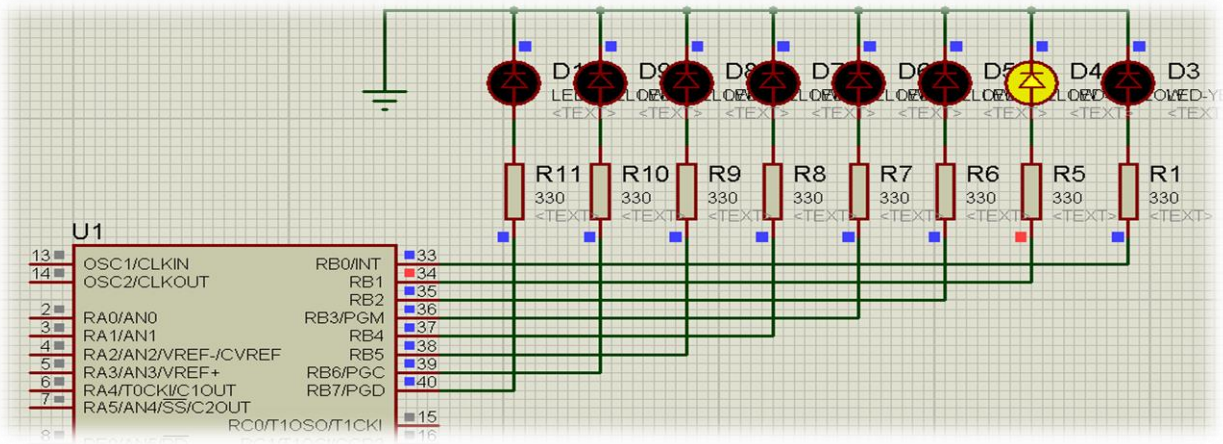
وعند تشغيل الدائرة فالصورة التالية تمثل أول قراءة والذي تشير إلى القيمة صفر:



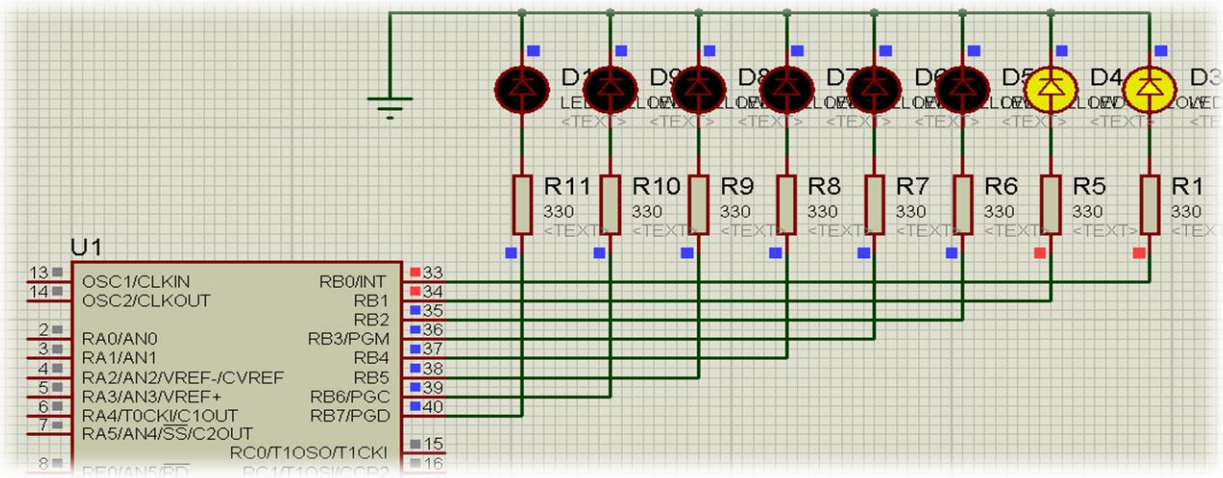
ثم تأتي القراءة التالية كما في الصورة لتشير إلى القيمة واحد بالثنائي:



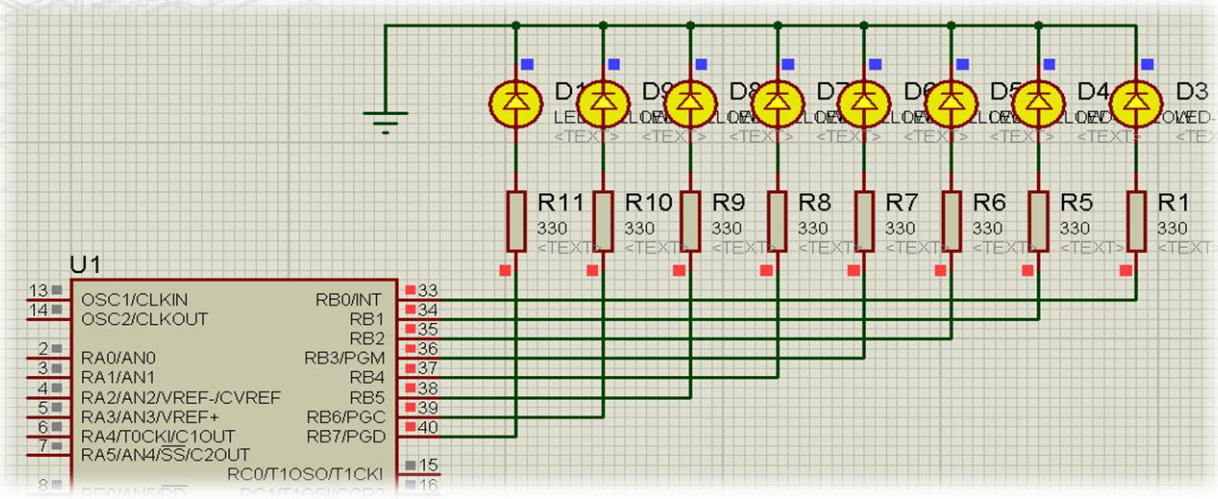
ثم القيمة ٢ بالثنائي أيضا:



وأيضا القيمة ٣ بالثنائي:

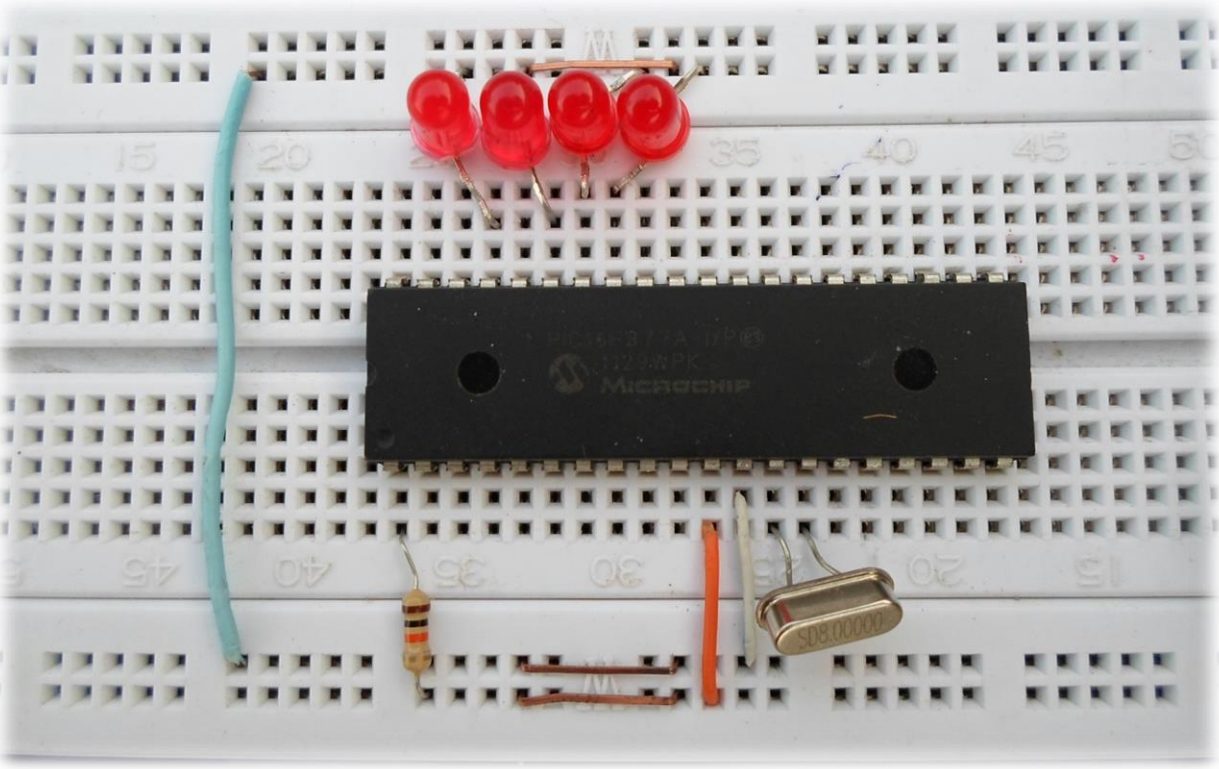


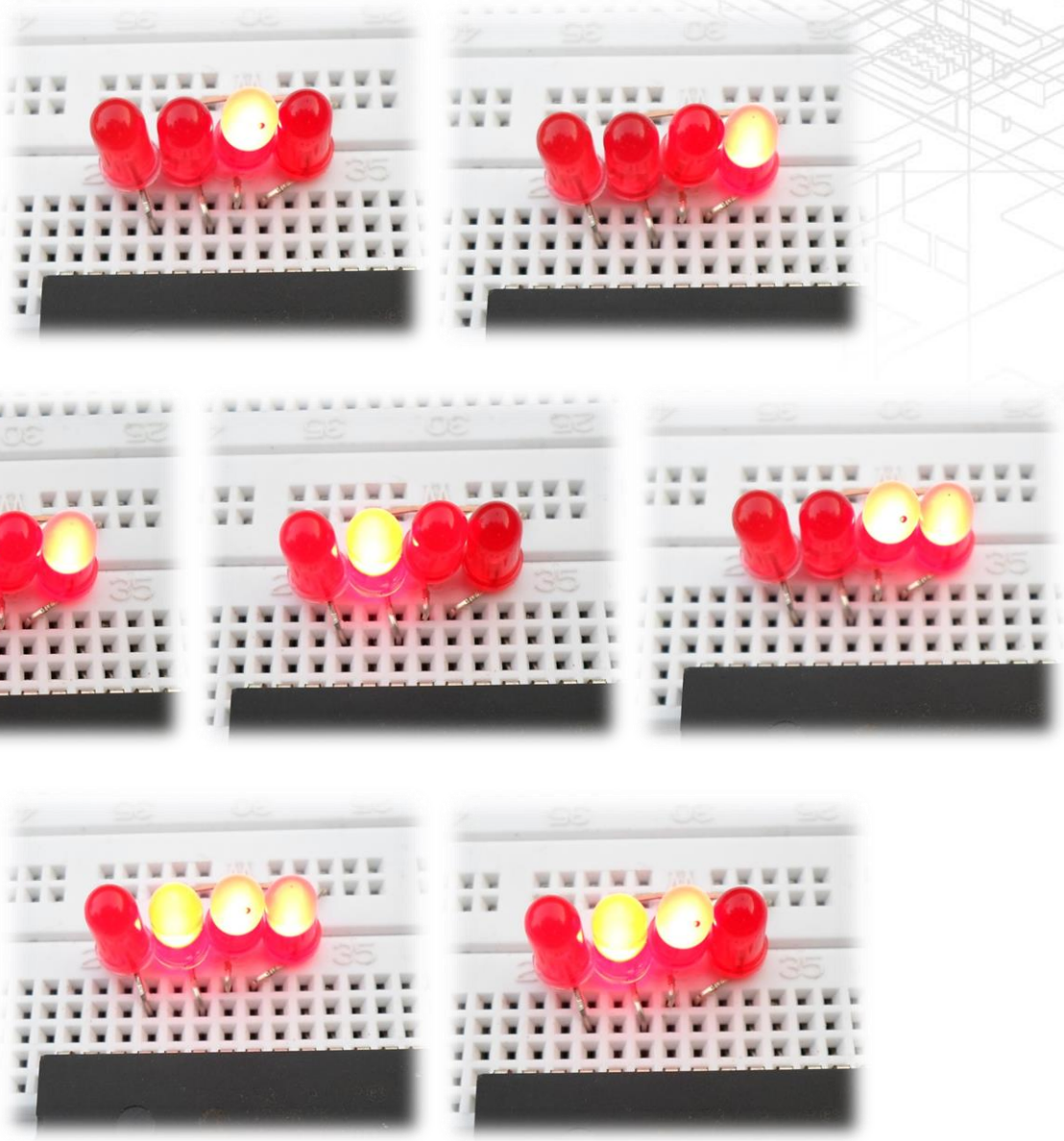
وهكذا حتى نصل إلى الصورة الأتية والتي تشير إلى القيمة ٢٥٥ بالثنائي:



تصميم الهاردوير

وهنا في التصميم الهاردوير وضعنا ٤ ليدات فقط وليس ٨ وبذلك تظهر الأرقام حتى مدى الأربعة بت فقط أي من صفر حتى ١٥





ولكن القيمة التالية لم تظهر كما هو متوقع فالمتوقع أن تضيء الليد اليسرى وينطفئ الباقي معبرين عن القيمة ٨ وهو ما لم يحدث ... ولعلك لاحظت الخطأ البسيط في هذا التصميم وهي أن الليد اليسرى لم يتم تركيبنها على الميكرو على الرجل الخاصة بها فلعلك لاحظت وجود رجل ميكرو فارغة بين الليد اليسرى والليد التي قبلها ... وعند تعديل هذا الخطأ يعمل البرنامج كما ينبغي ...

إضافات ومهارات

أولا: برنامج الفلاش الأول يمكن كتابته بصيغة أخرى كما هو موضح:

```
int i;
void main()
{

    TRISC.B0 = 0;

    while(1)
    {
        PORTC.B0 = 1;
        delay_ms(1000);

        PORTC.B0 = 0;
        delay_ms(1000);

    }
}
```

```
int i;
void main()
{

    TRISC.B0 = 0;

    while(1)
    {
        RB0_BIT = 1;
        delay_ms(1000);

        RB0_BIT = 0;
        delay_ms(1000);

    }
}
```

معناه اجعل قيمة RB0 بواحد

أي أن الأمر `PORTB.B0 = 1;` يكافئ الأمر `RB0_BIT = 1;` وأيضا الأمر `PORTC.B0 = 0;` يكافئ `RC0_BIT = 0;` وهكذا .....

ثانيا: برنامج إشارة المرور الذي كتب أثناء الشرح يمكن أيضا كتابته بصيغة أخرى، ففي الصيغة التي شرحت سابقا كنا نتعامل مع كل رجل على حدة فمثلا كنا نكتب أمر كالتالي `PORTC.B0 = 1;` لكن في هذه الطريقة سنتعامل مع الثلاثة رجول في أمر واحد كالآتي:

```
TRISC.B0 = 0;
TRISC.B1 = 0;
TRISC.B2 = 0;
```

يمكن استبدالهم بأمر واحد ← `TRISC=0b00000000;`

```
portc.B0 = 1;
portc.B1 = 0;
portc.B2 = 0;
```

يمكن استبدالهم بالامر المشار اليه ← `PORTC = 0B00000001;`

```
portc.B0 = 0;
portc.B1 = 1;
portc.B2 = 0;
```

يمكن استبدالهم بالامر المشار اليه ← `PORTC = 0B00000010;`

```
portc.B0 = 0;
portc.B1 = 0;
portc.B2 = 1;
```

يمكن استبدالهم بالامر المشار اليه

```
PORTC = 0B00000100;
```

وبالتالى سيصبح البرنامج الجديد كالاتى

```
void main()
{
    TRISC = 0B00000000;

    while(1)
    {
        PORTC = 0B00000001;
         delay_ms(1000);

        PORTC = 0B00000010;
         delay_ms(1000);

        PORTC = 0B00000100;
         delay_ms(1000);
     }
}
}
```

مشاريع يقوم القارئ بتطبيقها

١) مشروع به سويتش وليد، إذا تم الضغط على السويتش يتم إضاءة الليد وإذا تم الضغط عليه مرة أخرى يتم إطفاء الليد.

٢) مشروع لجعل ٨ ليدات يتم إضاءتهم وإطفائهم ٢٠ مرة فقط.

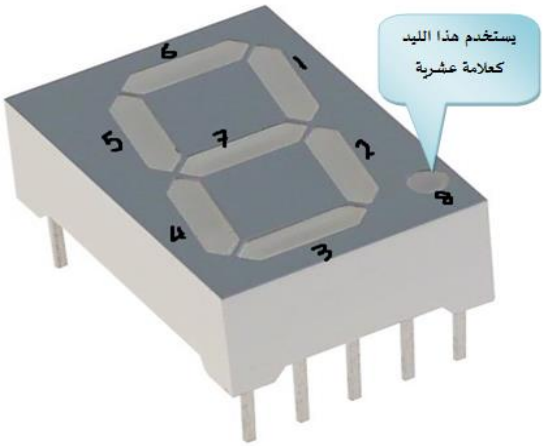
الفصل السادس

التعامل مع السيفين سيجمنت

يمكن للميكروكنترولر القيام بعمليات التحكم المختلفة، لكن توجد وظيفة أخرى لا تقل أهمية عن عملية التحكم نفسها وهي إظهار النتائج للمستخدم، وفى هذا الفصل سنتعرف على كيفية إظهار النتائج باستخدام ما يسمى 7-Segment

عن السيفين سيجمنت

تعريف



يمكن تعريف السيفين سيجمنت (7-Segment) على أنها عبارة عن ٧ ليدات أساسية مرتبة بطريقة تمكن من إظهار الأرقام وبعض الحروف، كما يتم وضع ليد إضافي ليمثل العلامة العشرية (dot) والذي يستخدم عندما نريد إظهار قيم تحتوي على علامة عشرية.

استخدامات

ومن أمثلة استخدامات هذا المكون الإلكتروني عرض قيمة درجة الحرارة التي يقوم الميكروكنترولر بقراءتها مثلا، وقد نرى السيفين سيجمنت في الأسانسير يظهر عليها رقم الدور الموجود فيه الأسانسير الآن، أو قد نجدها في البنوك ليعرض عليها رقم العميل الذي يقف على الشباك الآن، وببساطة تستخدم أيضا في ساعات الحائط وغير ذلك من الاستخدامات الكثير.

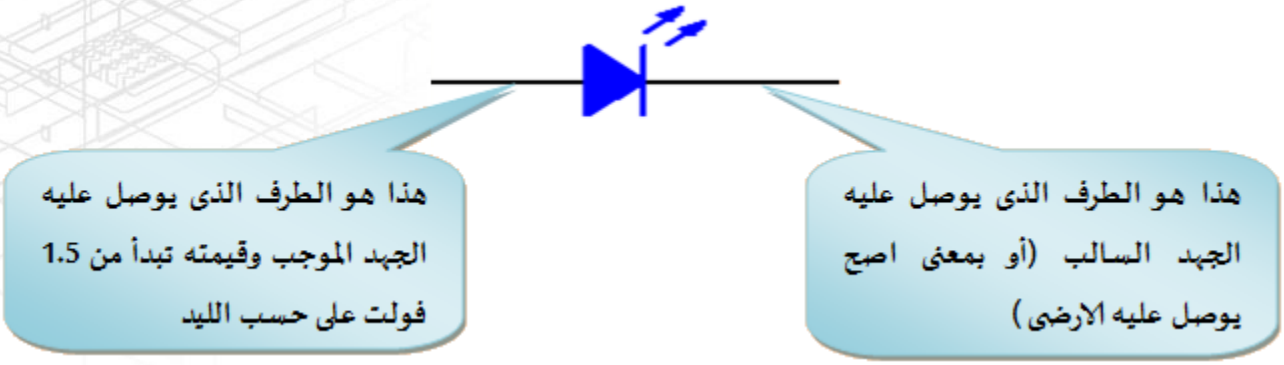

Reference 6

السيفين سيجمنت الواحدة يمكنها أن تعرض خانة الآحاد فقط أي الأرقام من صفر إلى تسعة وبالتالي لو أردنا أن نقوم بعرض قيم تتكون من آحاد وعشرات مثل ١٥ فسنحتاج لقطعتين من السفين سيجمنت وهكذا ...

أنواع السيفين سيجمنت

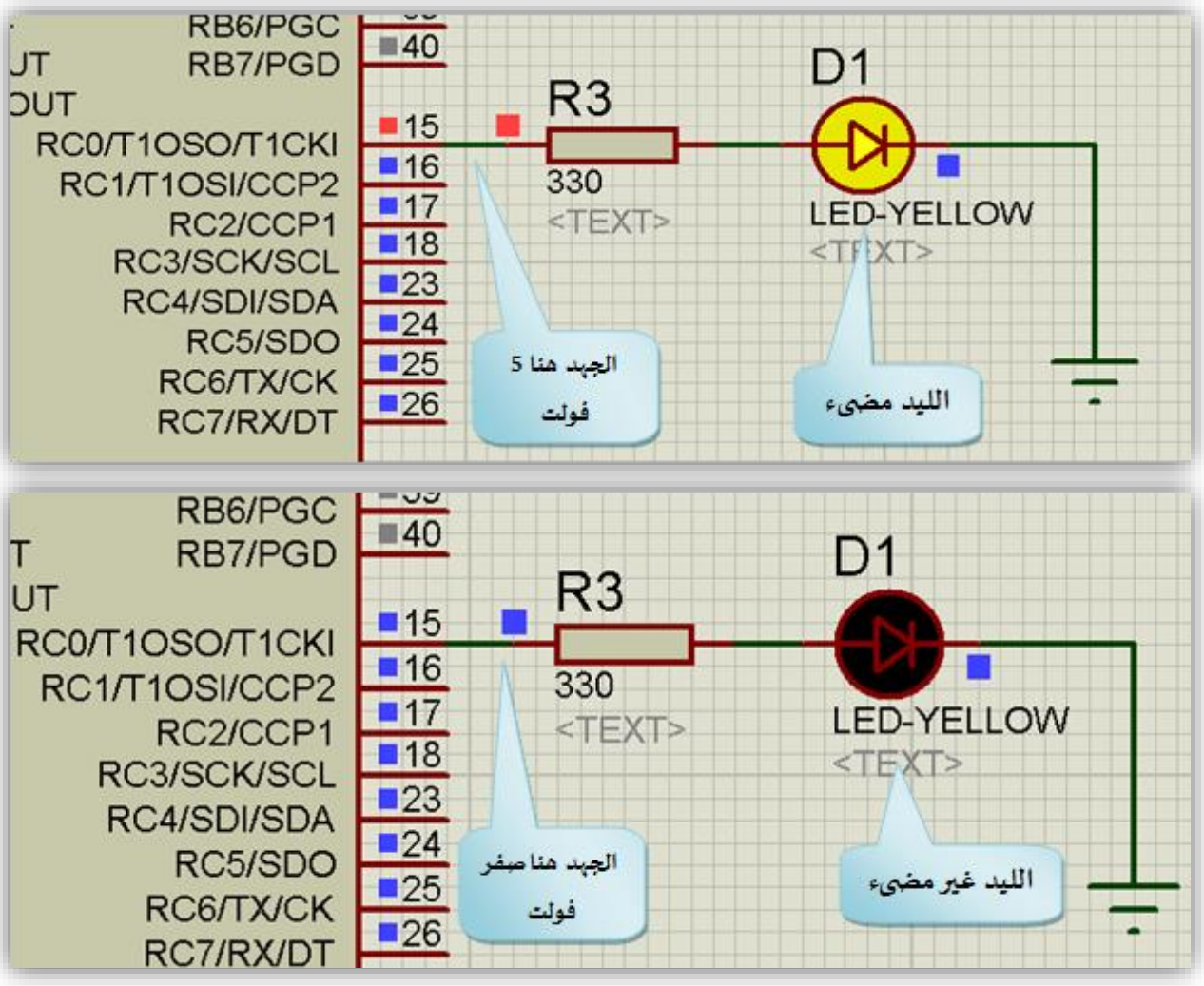
مقدمة عن الليد

قبل شرح أنواع السيفين سيجمنت لابد من التنويه عن معلومة مهمة في طريقة عمل الليد والتي على أساسها يتم تصنيف أنواع السيفين سيجمنت وهي أن الليد له طرفان أحدهما يوصل بالجهد الموجب والآخر يتم توصيله على الأرضي لكي يضئ الليد، وذلك كما هو موضح بالشكل:

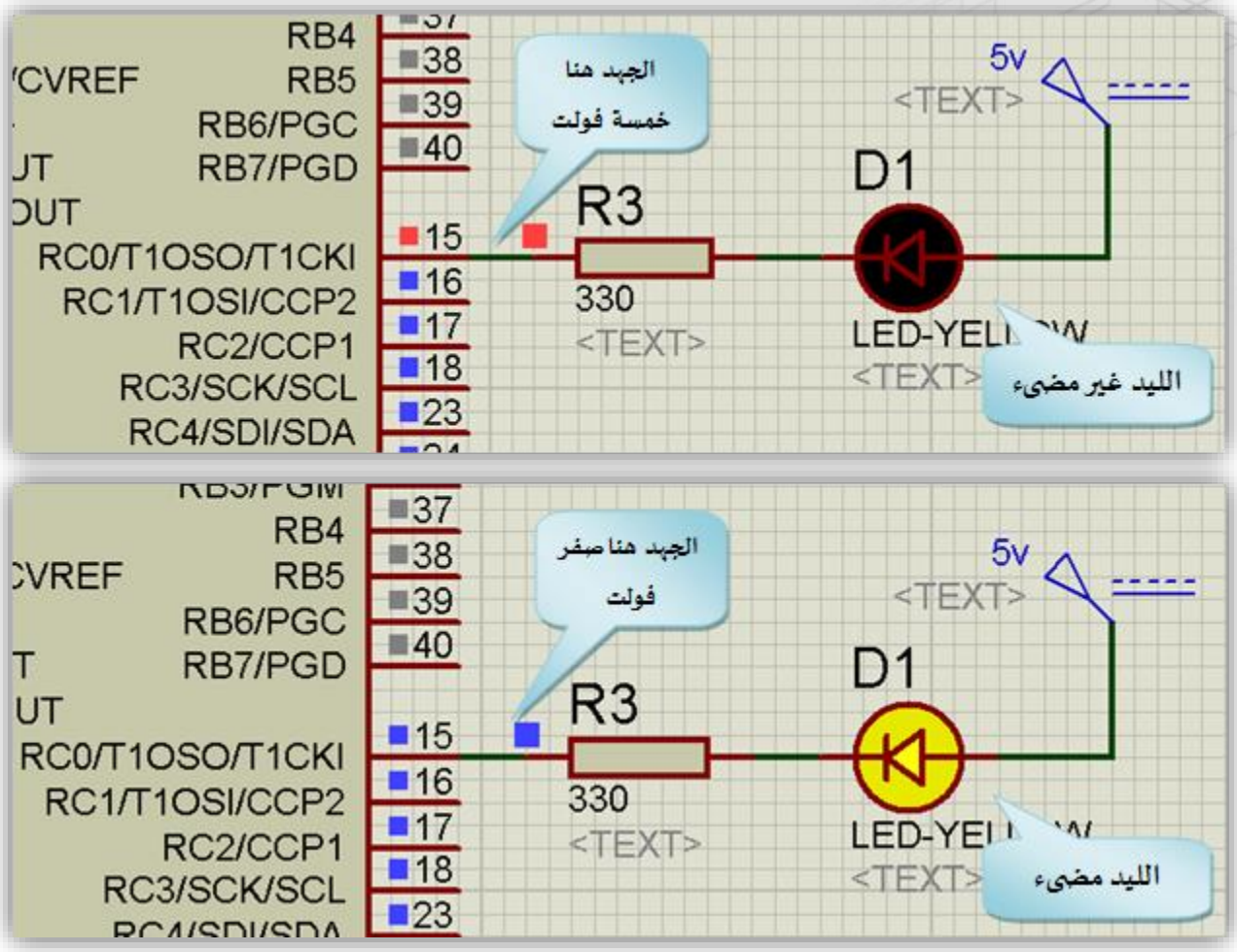


ومن هنا فإنه يوجد طريقتين لتوصيل الليد برجل الميكروكنترولر وإضاءته.

الطريقة الأولى: تكون بتوصيل الطرف الموجب لليد برجل الميكرو (من خلال المقاومة بالطبع)، وتوصيل الطرف الآخر بالأرضي، ثم لكي نقوم بإضاءة الليد لابد من إخراج ٥ فولت على رجل الميكروكنترولر وذلك كما هو بالشكل الآتي:



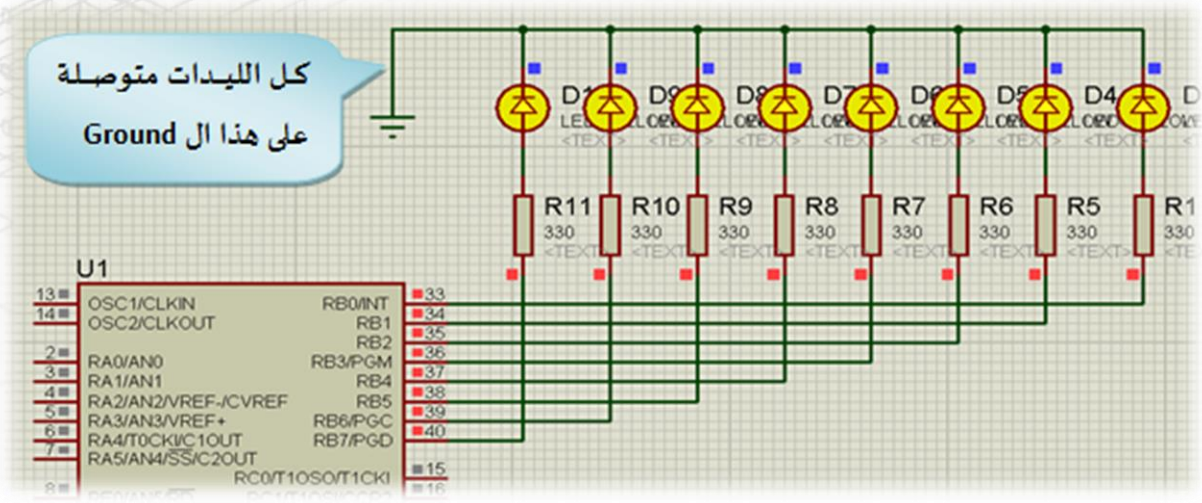
الطريقة الثانية: وهي كما بالشكل حيث يتم توصيل الطرف الموجب لليد بجهد خمسة فولت والطرف الآخر بالميكروكنترولر ولكي يضيء الليد يقوم الميكرو بإخراج جهد صفر فولت على رجله أما لو اخرج خمسة فولت فلن يضئ الليد حيث سيصبح طرفا الليد كل منهما عليه خمسة فولت:



ومن هذه المعلومة يمكنك استنتاج النوعين المختلفين للسيفين سيجمنت كما سيتم توضيح ذلك بعد قليل.

عدد رجول السيفين سيجمنت

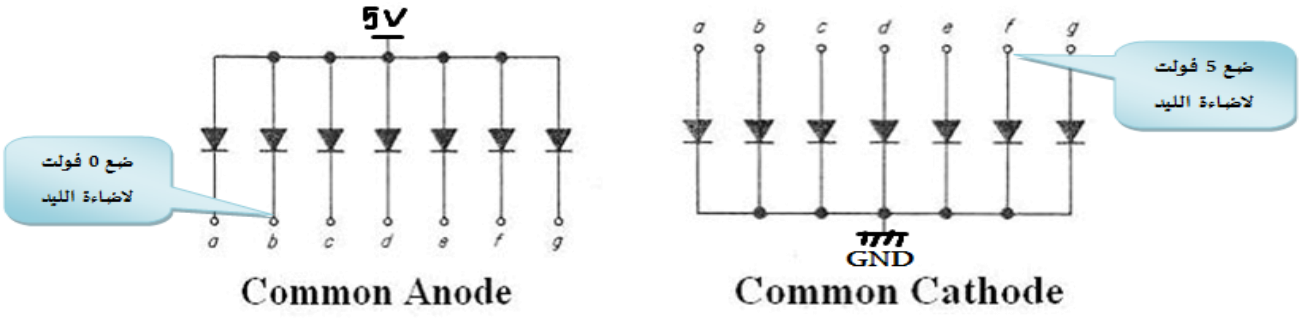
علمنا أن الليد له طرفان وأن السيفين سيجمنت يوجد فيها ٧ ليدات أساسية وليد إضافي وبالتالي إجمالي ٨ ليدات ... وبالتالي ما هو التوقع لعدد رجول قطعة السيفين سيجمنت ؟؟؟ ربما تكون إجابتك هي ١٦ رجل، ولكن عندما تعلم أن هناك رجل مشتركة بين كل الليدات وأن هذه الرجل ربما تكون الطرف الموجب لكل الليدات أو الطرف الأرضي لهم، ولعلك تتذكر هذه الرسمة من الفصل الماضي.

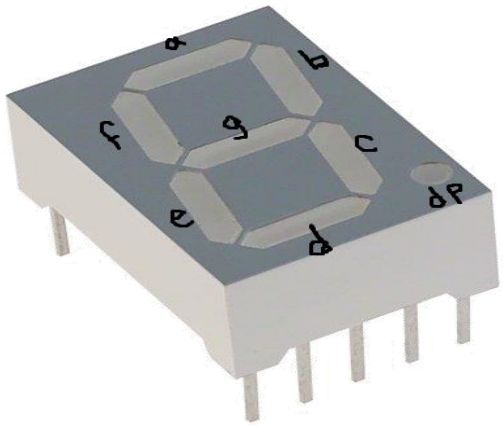


حيث تم توصيل كل الليدات على أرضي واحد ولم نقم بعمل أرضي لكل منهم، وبالمثل فإنه سيكون للسيفين سيجمنت طرف مشترك بين كل الليدات وهذه هي أول رجل ويبقى لكل ليد طرف آخر للتحكم في إضاءته وبالتالي ٨ رجول أخرين، وبناء عليه يكون للقطعة ٩ رجول ولكن في الغالب يكون في السيفين سيجمنت ١٠ رجول وليس ٩ حيث يوجد يتم تكرار الطرف المشترك مرة في الأعلى ومرة أخرى في الأسفل، وعند التوصيل على الهاردوير يمكن توصيل أحدهما فقط لانهما متصلان من الداخل وسيتم تفصيل هذا الاتصال لاحقا بإذن الله.

الأنواع

وبناءً على فهمنا لما سبق نستطيع أن نقسم السيفين سيجمنت إلى نوعين: **النوع الأول:** يسمى Common Cathode وكلمة Common معناها شيء مشترك ما بين مجموعة وكلمة Cathode تطلق دائما على الجزء الذي يحمل الإشارة السالبة (أو مجازا المتصل بالطرف الأرضي) ... وبالتالي فان Common Cathode تعنى أن الرجل المشتركة هنا هي الأرضي ولإضاءة أي ليد فيها نقوم بإخراج ٥ فولت على الرجل المناظرة ليها. **والنوع الثاني:** هو الـ Common Anode وفيه تكون الرجل المشتركة هي الطرف الموجب لليد ويتم توصيلها على ٥ فولت ولإضاءة أي ليد منها نقوم بإخراج صفر فولت على الرجل المناظرة ليها، والصورة التالية توضح الشكل الداخلي لكل منهما:



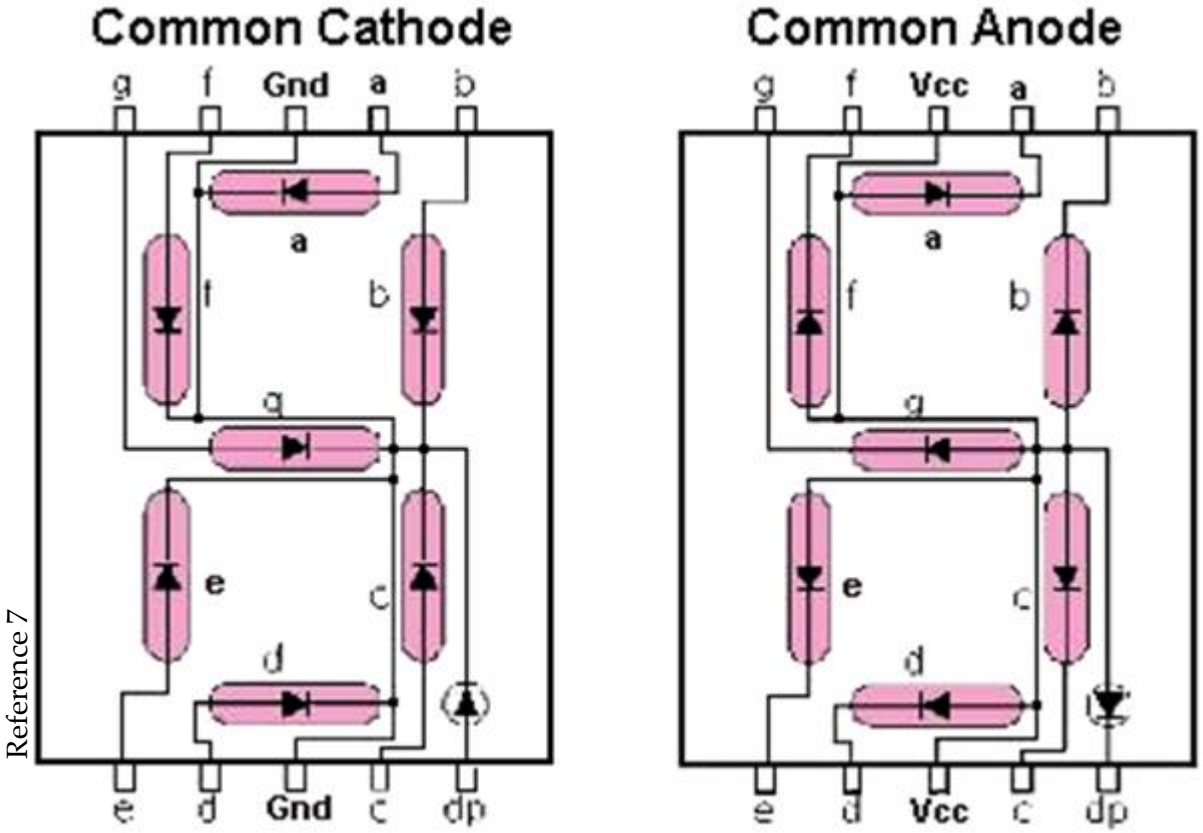


ونلاحظ من الرسمة السابقة أن كل ليد له اسم عبارة عن حرف يعرف به، الشكل المجاور يبين كل ليد وحرفه a، b، c، d، e، f، g، dp.

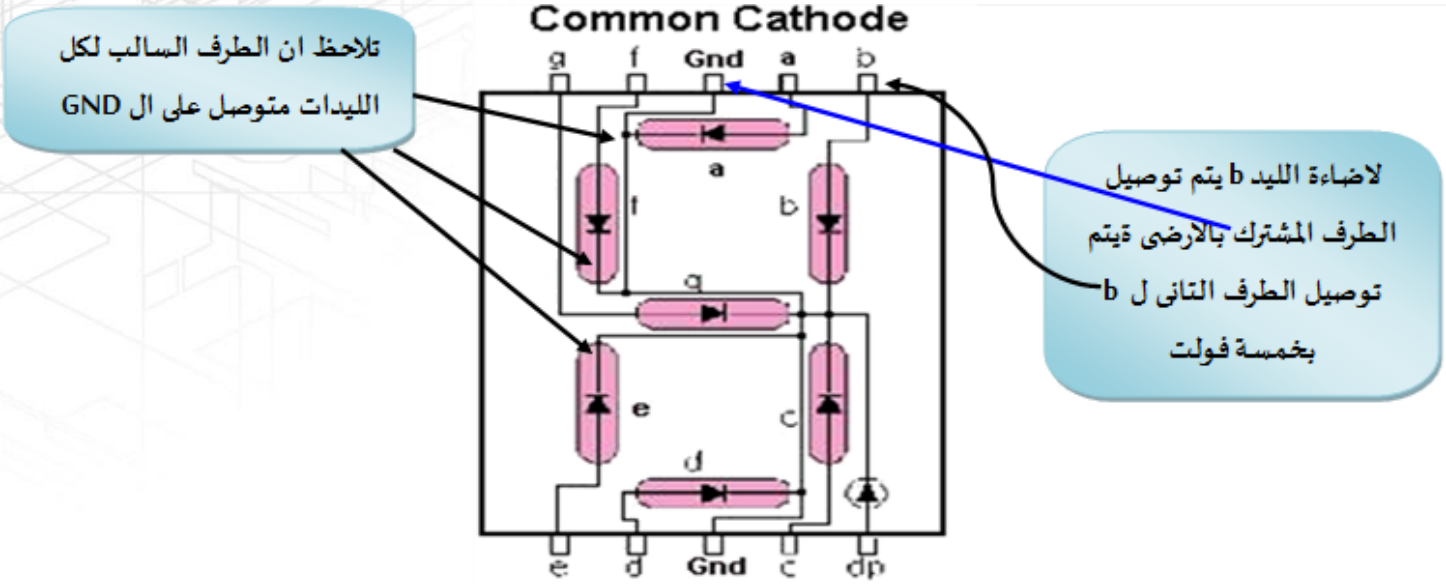
هذه الحروف والمكان الفعلي لليدات المناظرة لها يجب أن تكون معلومة حيث يفيد هذا عند توصيل السيفين سيجمنت بالميكروكنترولر.

التوصيل الداخلي

حتى الآن نكون قد فهمنا ما المقصود بالسيفين سيجمنت وأنواعها وترقيم الليدات داخلها والان تعالوا لنرى كيف هو شكلها الحقيقي وكيفية توصيل الليدات من الداخل لنعلم أي ليد متصلة بأي رجل من رجول القطعة الإلكترونية.



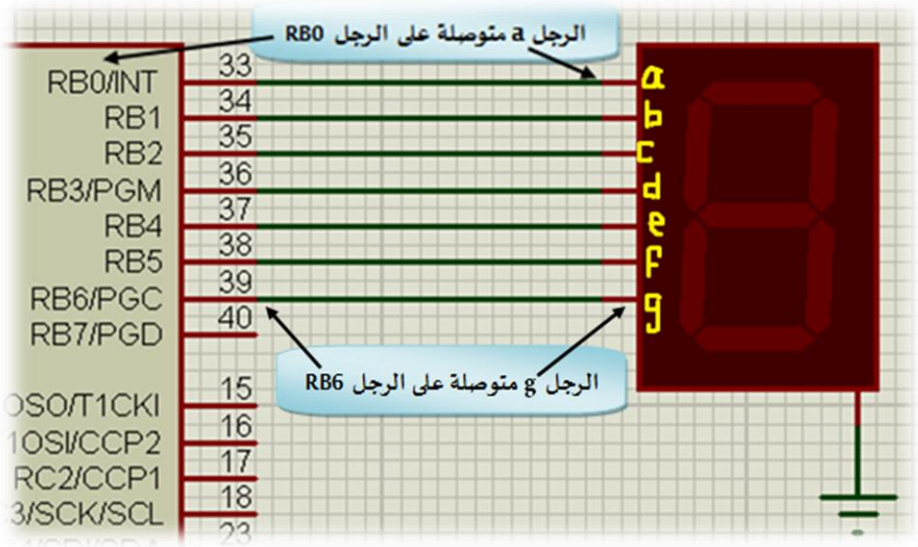
وعند النظر بشيء من التمحيص للصورة اليسرى نلاحظ اشتراك جميع الليدات في طرف الأرضي واتصال الأطراف الموجبة بالرجول الأخرى للقطعة وذلك كما هو موضح في الصورة التالية، وبالطبع يمكن معاملة الصورة اليمنى بالمثل أيضا باختلاف أنهم مشتركين في الرجال الموجبة:



مشروع تطبيقي

لشرح كيفية كتابة البرنامج للتعامل مع السيفين سيجمنت لابد من توضيح القيم التي تظهر على البورت المتصل بها لإظهار الأرقام المطلوبة ...

دعنا نختار سيفين سيجمنت من النوع Common Cathode لننفذ عليها مشاريع هذا الفصل أي أن الرجل المشتركة تتصل بالأرضي ولكي تقوم بإضاءة ليد نضع خمسة فولت على رجل الميكرو المتصلة برجل القطعة المناظرة لها. وسنهمل التعامل مع الليد الصغير الخاص بالعلامة العشرية وسنعتبره غير موجود لأننا لا نحتاجه (رقم وحيد بدون كسور).

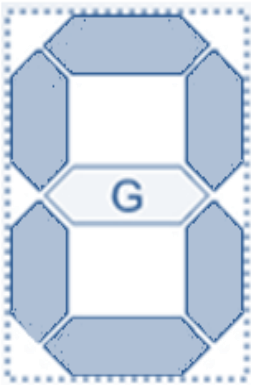


دعنا أيضا نختار المخرج B لنقوم بتوصيل السيفين سيجمنت.

ملحوظة: ترتيب رجول السيفين سيجمنت في بروتس في الصورة ليست كما هي في الهاردوير.

قيم البورت المناظرة للأرقام على السيفين سيجمنت

دعنا نكون جدول يكون فيه كل رقم من صفر إلى تسعة وما يقابله من قيمة يجب أن تخرج على بورت الميكرو لإظهار هذا الرقم، لتكوين هذا الجدول نرسم كل رقم على السيفين سيجمنت و نجد أي الليدات مضيء وأيها مطفئ.



لو أردنا أن نعرض الرقم صفر فما هي حروف الليدات التي يجب إضاءتها لإظهاره كما بالشكل المجاور ؟؟؟ انظر إلى الرسمة واكتب الحروف واذكر الحروف المضيئة فتجد الجميع مضيء عدا الليد g فهو غير مضيء كما بالشكل.

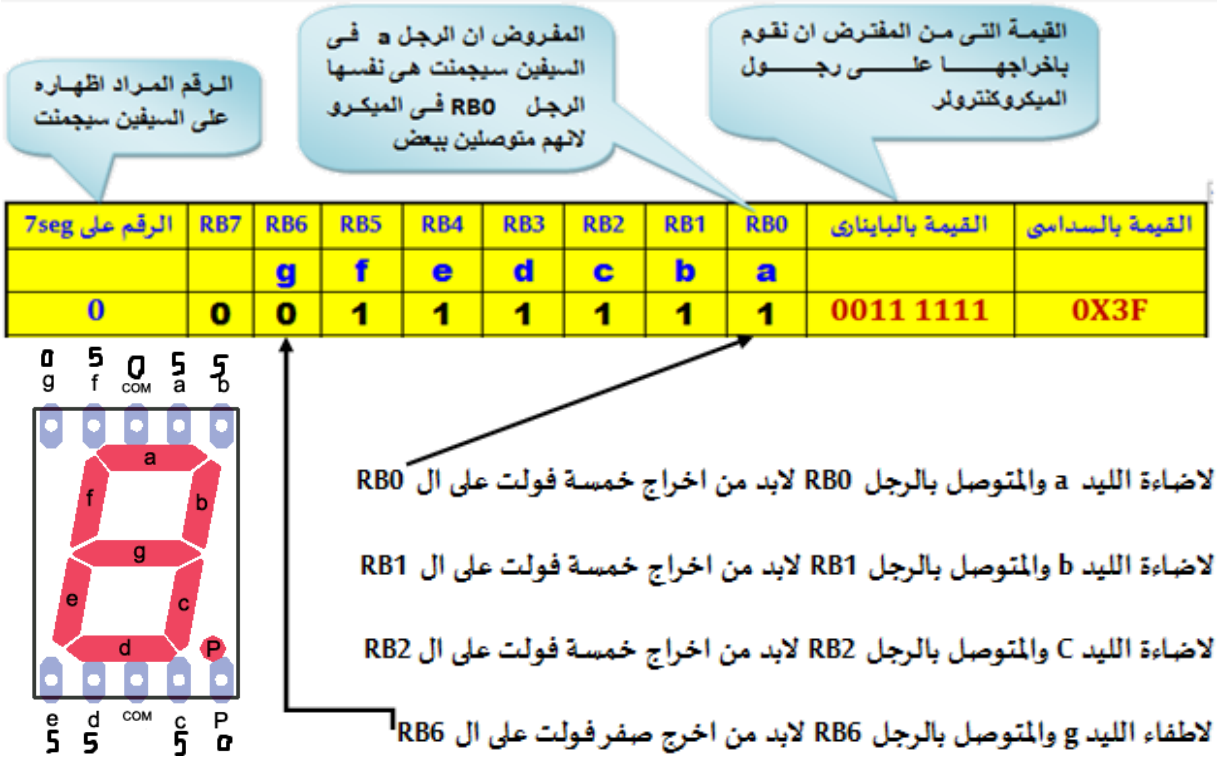
وهذا معناه أننا لابد أن نوصل خمسة فولت على رجول كل الليدات عدا الرجل g، وبالتالي لابد أن تكون القيمة على المخرج PORTB تساوي ٠٠١١١١١١ وذلك عن طريق الأمر التالي:

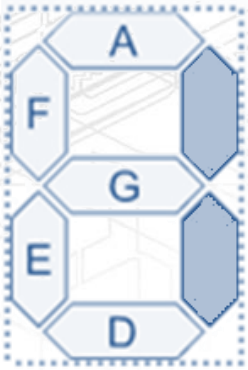
```
PORTB = 0B00111111;
```

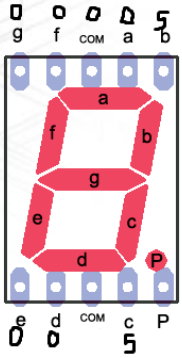
أو بصورته بالتمثيل السداسي عشر التي يمكن الحصول عليها بواسطة المحول الموجود في برنامج الميكرو سي.

```
PORTB = 0x3F;
```

وبالتالي يتكون أول صف في الجدول:







وبالمثل إذا أردنا عرض الرقم واحد فسوف نضئ الليدات b، cفقط وبالتالي فان رجولهم الرجول RB1، RB2 هي ما سيتم إخراج عليها خمسة فولت وباقي رجول المخرج PORTB سيكون عليه صفر فولت، وهذا يتم من خلال الأمر التالي:

```
PORTB = 0B00000110;
```

وهنا يتم إضافة الصف الثاني للجدول ليكون بالشكل التالي:

الرقم على 7seg	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	القيمة بالباينارى	القيمة بالسداسى
		g	f	e	d	c	b	a		
0	0	0	1	1	1	1	1	1	0011 1111	0X3F
1	0	0	0	0	0	1	1	0	0000 0110	0X06

الليد c , b فقط هما اللى هنخرج عليهم واحد (يعنى خمسة فولت) والباقى هيخرج عليه اصفار كما هو موضح

وبفعل المثل بالنسبة لباقي الأرقام يصبح شكل الجدول النهائي كما يلي:

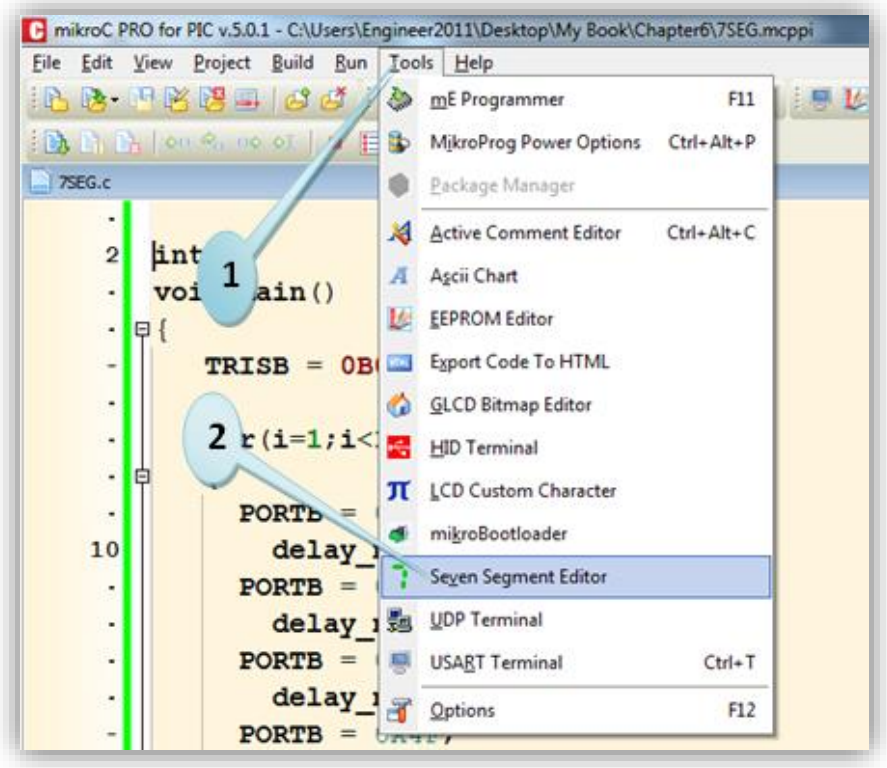
الرقم المراد اظهاره على السيفين سيجمنت

المفروض ان الرجل a فى السيفين سيجمنت هى نفسها الرجل RB0 فى الميكرو لانهم متوصلين ببعض

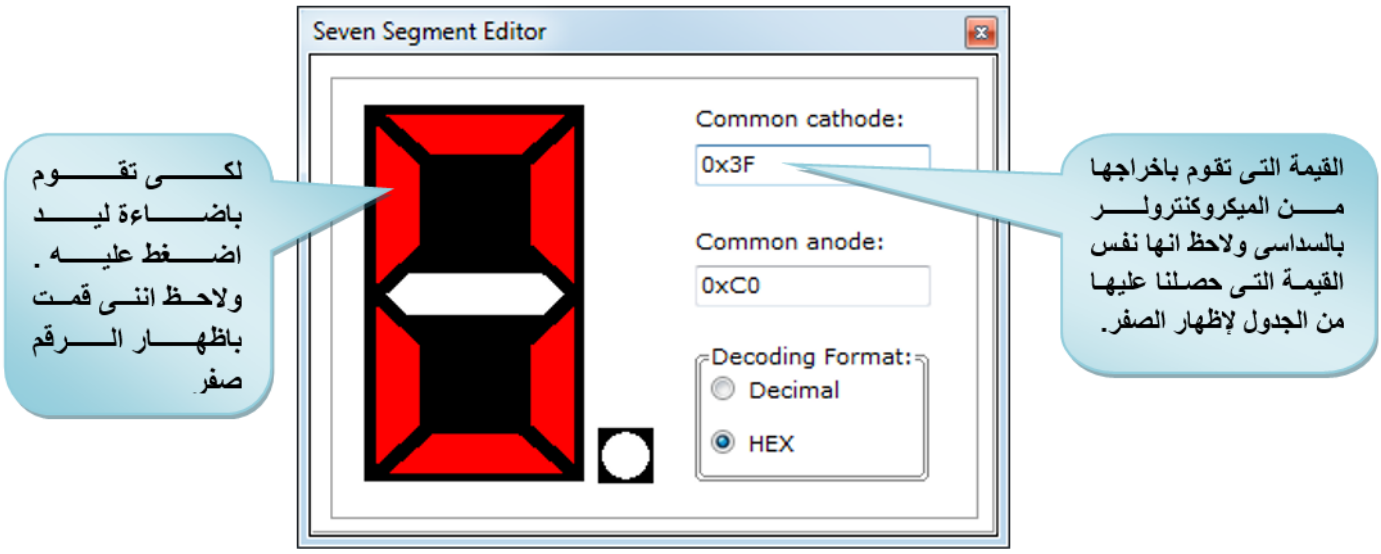
القيمة التى من المفترض ان نقوم باخراجها على رجول الميكروكنترولر

الرقم على 7seg	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	القيمة بالباينارى	القيمة بالسداسى
		g	f	e	d	c	b	a		
0	0	0	1	1	1	1	1	1	0011 1111	0X3F
1	0	0	0	0	0	1	1	0	0000 0110	0X06
2	0	1	0	1	1	0	1	1	0101 1011	0X5B
3	0	1	0	0	1	1	1	1	0100 1111	0X4F
4	0	1	1	0	0	1	1	0	0110 0110	0X66
5	0	1	1	0	1	1	0	1	0110 1101	0X6D

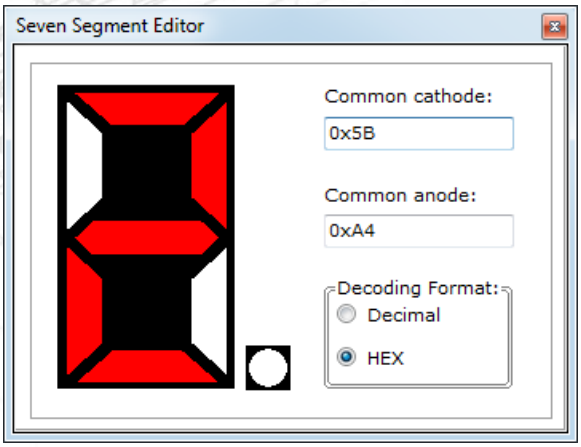
والآن بعد أن فهمنا هذه القيم وكيفية حسابها، هناك طريقة أخرى أسرع وأسهل للحصول على هذه القيم اللازمة وهي كالآتي:

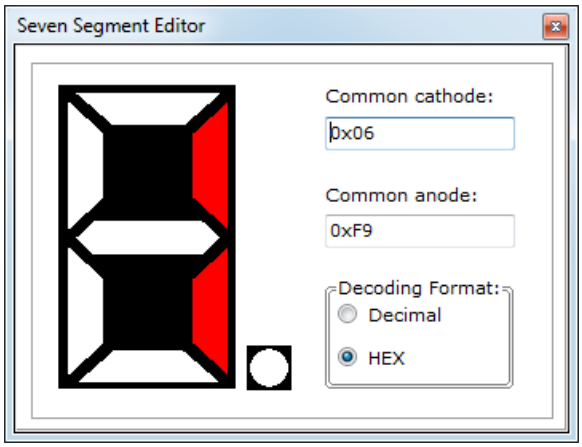


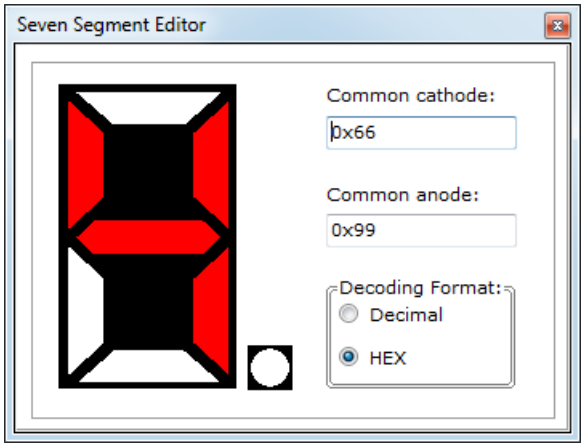
Tools menu
↓
Seven Segment Editor

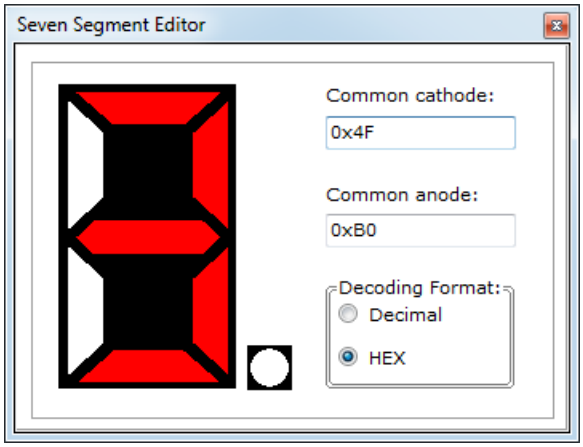


في هذه النافذة نستطيع الحصول على القيمة المماثلة للأرقام سواء كان Common cathode كما في الخانة العلوية أو Common anode كما في الخانة السفلية وأيضا يمكن عرض القيمة بالنظام العشري أو بالنظام السداسي عشر وذلك من المربع السفلي.









البرامج

إظهار الرقم الصفر فقط

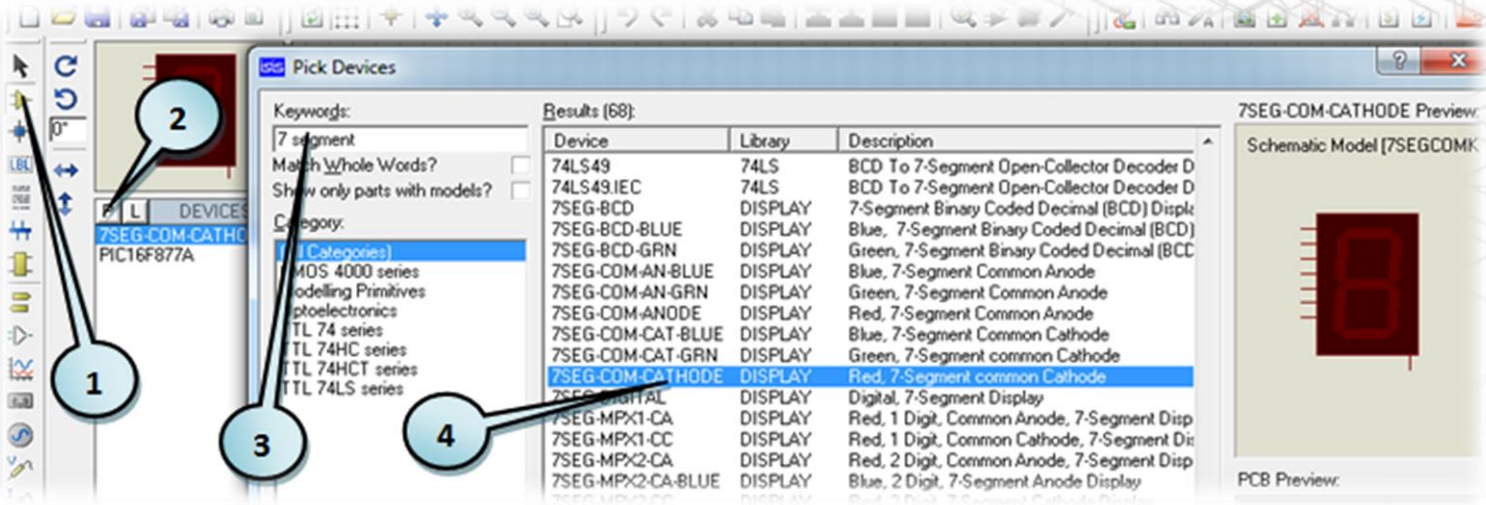
دعنا نبدأ ببرنامج يقوم بإظهار الرقم صفر فقط على السيفين سيجمنت، نقوم بعمل مشروع جديد في الميكرو سي وكتابة البرنامج التالي وعمل Build.

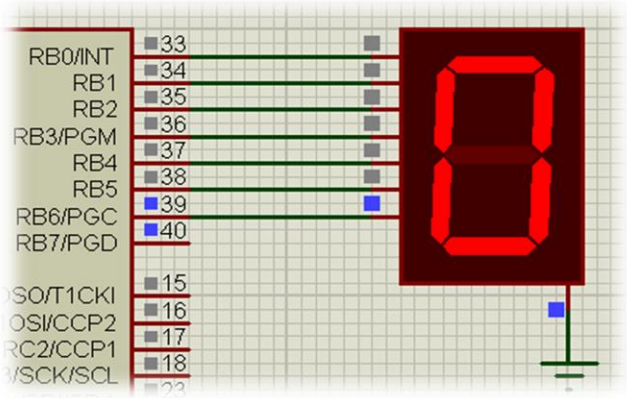
```
void main()
{

    TRISB = 0B00000000;

    PORTB = 0B00111111;

}
```

لجعل PORTB يعمل كخرج

لاخراج القيمة 00111111 على PORTBوالتى تستخدم لاظهار صفر على 7SEG

دعنا ننتقل سريعا إلى محاكاة البرنامج على بروتس، لنقم بتجهيز المكونات المطلوبة وتوصيل الدائرة، وللحصول على السيفين سيجمنت نتبع الشكل الآتي:





ثم قم بتوصيل السيفين سيجمنت بالبورت B على الميكرو ثم قم بتحميل ملف البرنامج الذي سبق أن حصلت من برنامج الميكرو سي على الميكروكنترولر كما سبق وتعلمنا ثم ابدأ المحاكاة لتجد النتيجة كما هي بالشكل المقابل

إظهار الرقم صفر ثم الرقم واحد

نريد عمل برنامج آخر يقوم بعرض الرقم صفر ثم الرقم واحد ... حاول أن تخيل أنت كيف سيكون شكل البرنامج بناء على ما تعلمته من الفصول السابقة واكتبه، أو أنظر إلى البرنامج التالي واختبره: هل هو صحيح ام لا ...

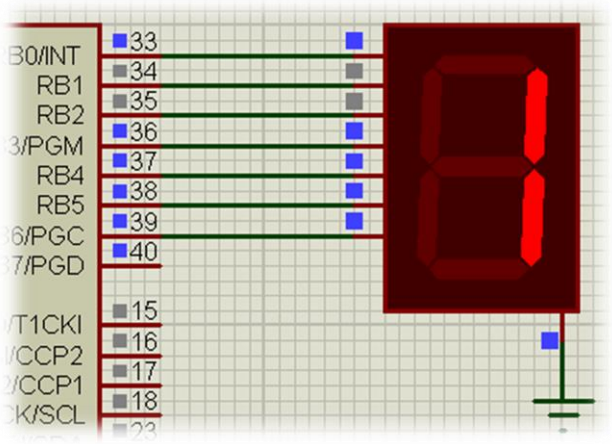
```
void main()
{

    TRISB = 0B00000000;

    PORTB = 0B00111111;
    PORTB = 0B00000110;
}
```

لاظهار صفر على ال 7SEG

لاظهار واحد على ال 7SEG

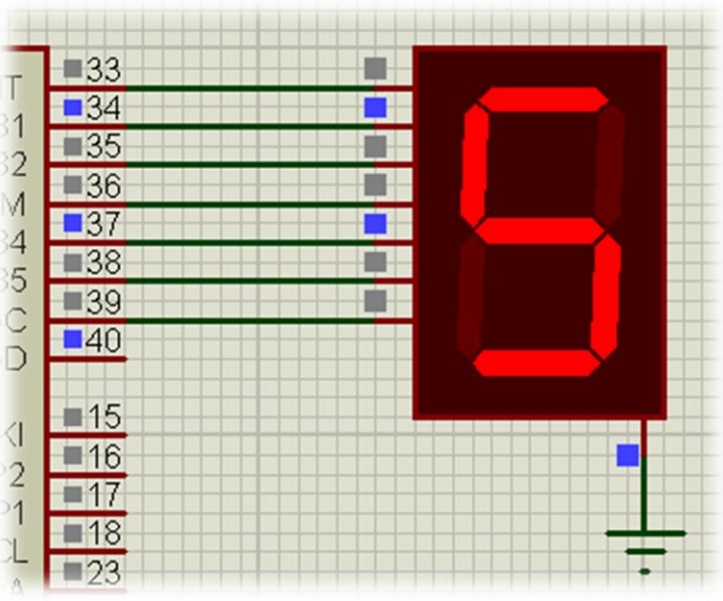
لعلك لاحظت هذا الخطأ البسيط: فمن المفترض أن هذا البرنامج سيعرض أولا صفر على السيفين سيجمنت، ثم يعرض واحد ويقف على ذلك حيث لا توجد حلقة while التي تحدث تكرار ... ولكن السؤال هنا يقول: كم المدة التي سيعرض فيها الصفر والتي بعدها يعرض الواحد؟؟ في هذا البرنامج سيعرض الصفر وبسرعة كبيرة جدا سيعرض الواحد لدرجة أننا لن نلحظ الصفر بأعيننا لأنها ستختفي بسرعة، والحل هو أن نضع أمر delay بين أمر الصفر وأمر الواحد ليصبح البرنامج كالتالي:

```
void main()
{

    TRISB = 0B00000000;

    PORTB = 0B00111111;
      delay_ms(2000);
    PORTB = 0B00000110;
}
```



إظهار الأرقام من صفر إلى خمسة

```
void main()
{
    TRISB = 0B00000000;

    PORTB = 0B00111111;
      delay_ms(1000);
    PORTB = 0B00000110;
      delay_ms(1000);
    PORTB = 0B01011011;
      delay_ms(1000);
    PORTB = 0B01001111;
      delay_ms(1000);
    PORTB = 0B01100110;
      delay_ms(1000);
    PORTB = 0B01101101;
}
```



ملحوظة: يمكن كتابة البرنامج السابق كما يلي طبقا للقيم السداسية عشر:

```
void main()
{
    TRISB = 0B00000000;

    PORTB = 0x3F;
      delay_ms(1000);
    PORTB = 0X06;
      delay_ms(1000);
    PORTB = 0X5B;
      delay_ms(1000);
    PORTB = 0X4F;
      delay_ms(1000);
    PORTB = 0X66;
      delay_ms(1000);
    PORTB = 0X6D;
}
```

حيث يوجد في الجدول عمود يحتوي على القيم الثنائية وعمود آخر يحتوي على القيم بالسداسي عشر ويمكنك الاختيار بين أي من العمودين مع تغيير البادئة قبل القيمة من 0B إلى 0X.

التكرار

البرنامج الماضي يقوم بإظهار الأرقام من صفر إلى خمسة، ثم ينهى التنفيذ، وبهذا تكون آخر قيمة على السيفين سيجمنت هي آخر قيمة في البرنامج أي القيمة خمسة، فاذا أردنا البرنامج أن يقوم بعرض نفس هذه الأرقام ثم تكرارها ... فاذا كنت تريد عدد لا نهائي من مرات التكرار سوف نستخدم حلقة `while (1)` أما إذا كنت تريد عدد معين من التكرار يتم استخدام حلقة `for` وسأكتب لكم المثالين

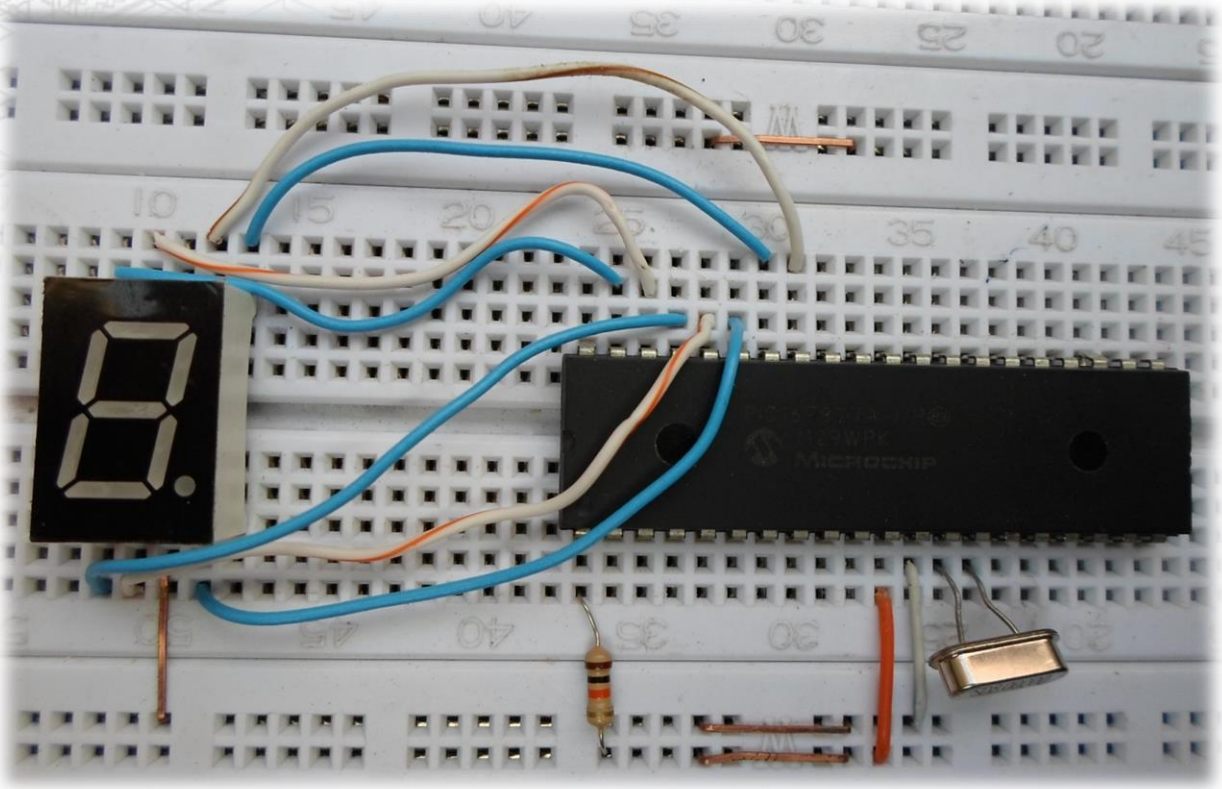
```
void main()
{
    TRISB = 0B00000000;

    while(1)
    {
      PORTB = 0x3F;
        delay_ms(1000);
      PORTB = 0X06;
        delay_ms(1000);
      PORTB = 0X5B;
        delay_ms(1000);
      PORTB = 0X4F;
        delay_ms(1000);
      PORTB = 0X66;
        delay_ms(1000);
      PORTB = 0X6D;
        delay_ms(1000);
    }
}
```

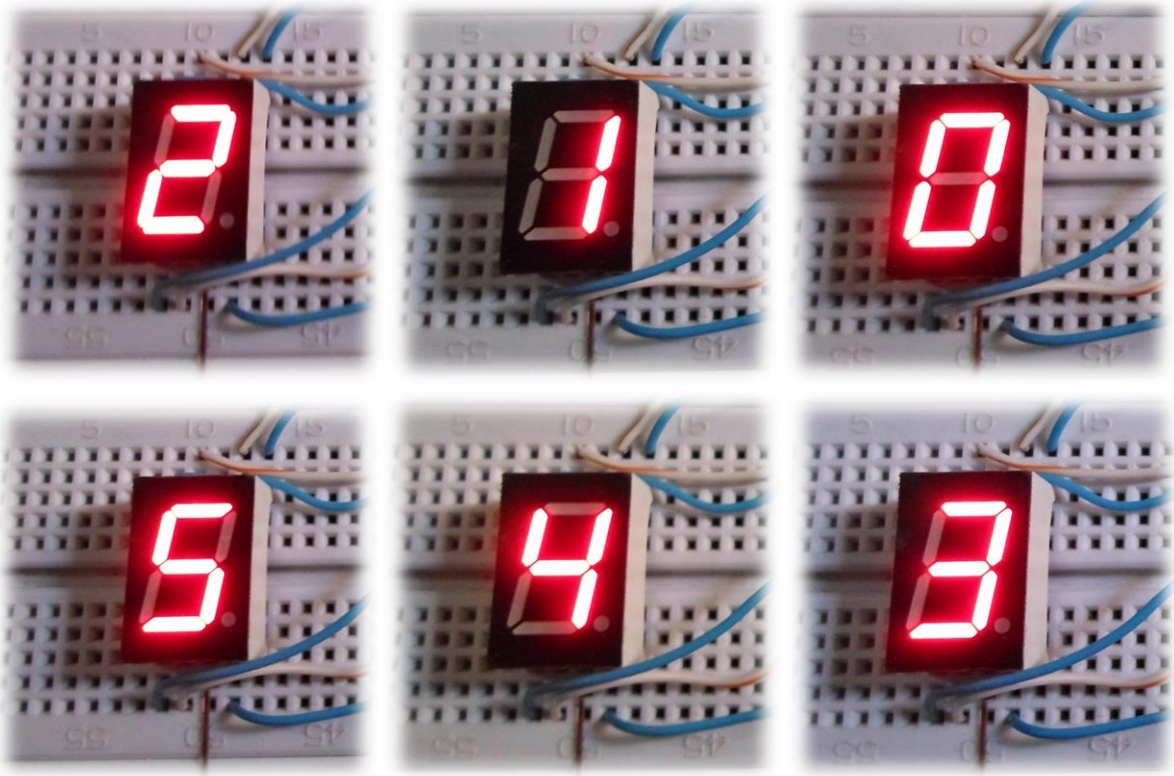
لاحظ وجود delay هنا حتى لا يتم الانتقال الى اول امر فى while دون أن نرى الرقم خمسة يظهر.

```
int i;
void main()
{
    TRISB = 0B00000000;

    for(i=1;i<10;i++)
    {
      PORTB = 0x3F;
        delay_ms(1000);
      PORTB = 0X06;
        delay_ms(1000);
      PORTB = 0X5B;
        delay_ms(1000);
      PORTB = 0X4F;
        delay_ms(1000);
      PORTB = 0X66;
        delay_ms(1000);
      PORTB = 0X6D;
        delay_ms(1000);
    }
}
```

عدد مرات التكرار

الهاردوير

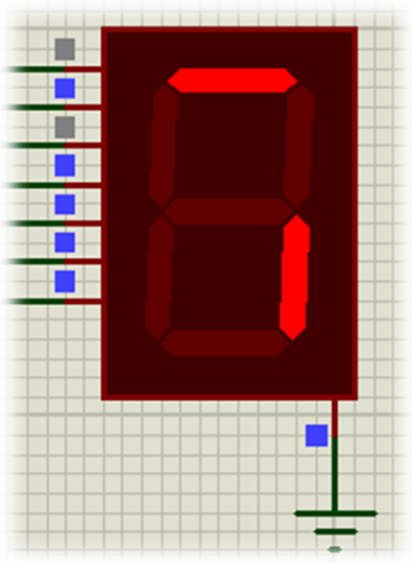




تعديل للتسهيل

في الأجزاء الماضية تعلمنا كيفية إخراج قيم على PORTB التي يتم ترجمتها إلى أرقام على السيفين سيجمنت، وكنا إذا أردنا إظهار الرقم صفر لا نخرج صفر على البورت وإنما نخرج القيمة ٠٠١١١١١١ بالثنائي أو 0X3F بالسداسي عشر، ولكن ألا يوجد ما هو أسهل بحيث إذا أردنا إظهار الصفر على السيفين سيجمنت نخرج صفر على البورت وليس قيمة أخرى معقدة أي ألا يمكن أن يكون شكل البرنامج كالآتي؟؟

```
void main()
{
    TRISB = 0B00000000;

    while(1)
    {
      PORTB = 0;
        delay_ms(1000);
      PORTB = 1;
        delay_ms(1000);
      PORTB = 2;
        delay_ms(1000);
      PORTB = 3;
        delay_ms(1000);
      PORTB = 4;
        delay_ms(1000);
      PORTB = 5;
        delay_ms(1000);
    }
}
```



إذا قمنا بكتابة البرنامج بهذا الشكل وكان توصيل الميكرو بالسيفين سيجمنت كما هو عليه في المشاريع السابقة فلن تعرض القيم ٠ و١ و٢ و٣ و٤ و٥ وذلك لأننا لم نخرج قيم مناسبة بل ستظهر أشكال غير مفهومة كما بالصورة المقابلة.

ولكن إذا أردنا حل لكتابة البرنامج بهذه السهولة المعهودة فإن هذا الحل يكمن في تركيب IC معين بين الميكرو والسيفين سيجمنت يقوم بتحويل القيم الخارجة من الميكرو إلي قيمها المناظرة اللي تعرض الرقم المناظر لها على السيفين سيجمنت، وبالتالي لكي نظهر الرقم صفر مثلا لا نكتب القيمة ٠٠١١١١١١ بل ببساطة نكتب الأمر التالي:

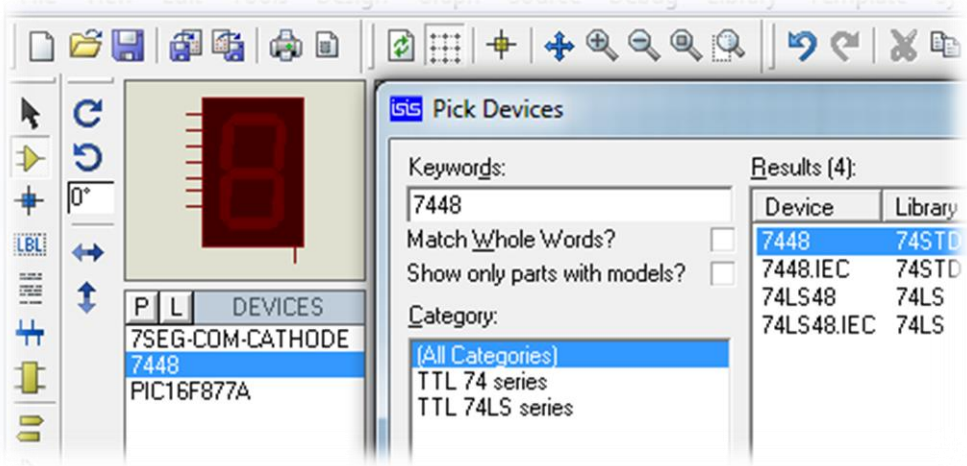
```
PORTB = 0;
```

ثم يقوم هذا الـIC باستلام القيمة صفر من الميكرو و يحولها إلى ٠٠١١١١١١ ويسلمها بدوره إلى السيفين سيجمنت، وهكذا لإظهار واحد نجعل الميكرو يخرج القيمة واحد بالأمر:

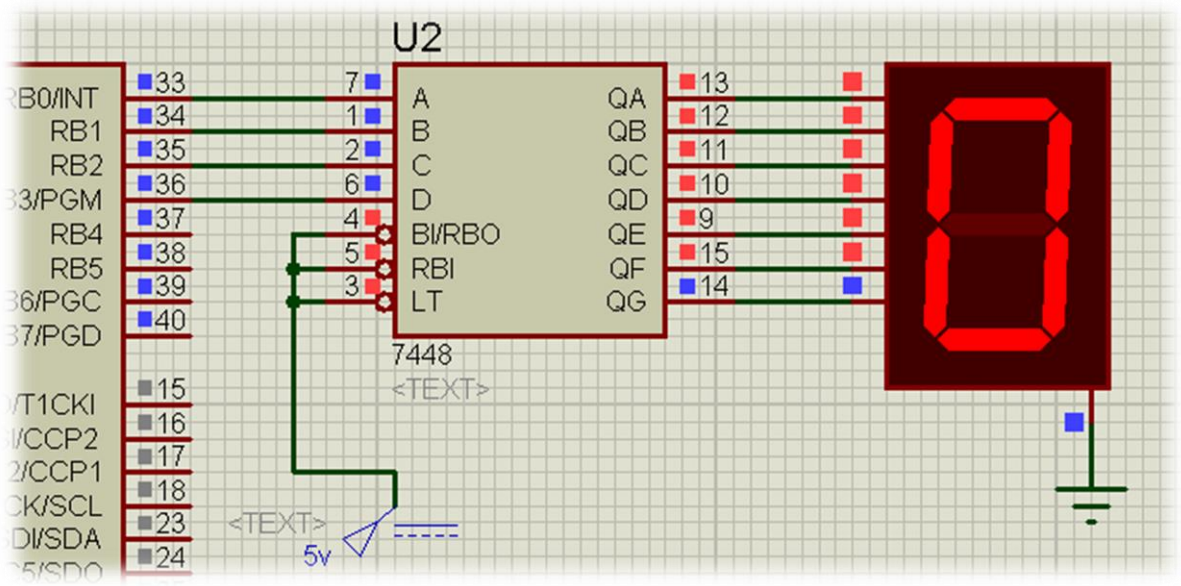
```
PORTB = 1;
```

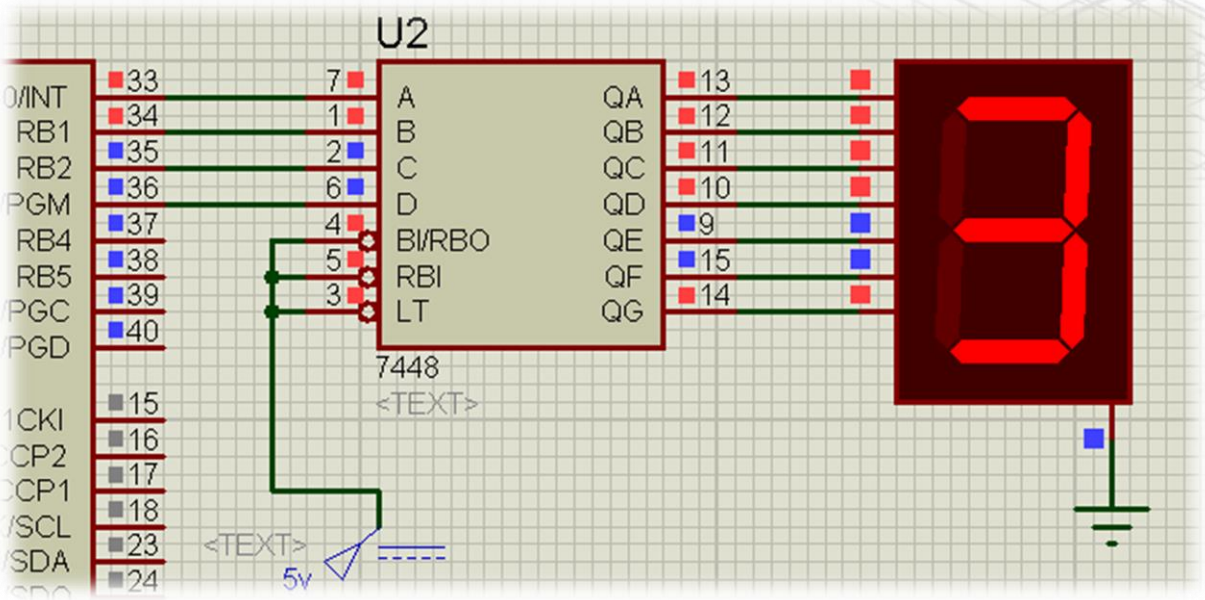
ويقوم الـ IC بتحويل القيمة واحد الواصلة له من الميكرو إلى القيمة 00000110 التي يحتاجها السيفين سيجمنت لكي تظهر الرقم واحد ... وهكذا بالنسبة لباقي الأرقام.

هذا الـ IC يحمل الرقم 7448 في حالة الـ Common Cathode أو الرقم 7447 في حالة الـ Common Anode وللحصول عليه في بروتس انظر الصورة الآتية:



وبعد عمل البرنامج السابق على الميكرو سي وعمل build له وتحميله على الميكرو قم بتوصيل الدائرة كما بالشكل الآتي وشغل الدائرة ليظهر الخرج كاللقطة التالية:





ميزة الطريقة الثانية عن الأولى انه تم استخدام عدد قليل من رجول الميكروكنترولر حيث تم استخدام أربعة رجول فقط في حين انه في الحالة الأولى تم استخدام سبعة، وهذا نلجأ الحل نضطر إليه غالبا إذا كان المشروع كبير و نحتاج لتوصيل أشياء كثيرة بالميكرو(*).

تسجيل الجدول في البرنامج

عادة لا نستخدم السيفين سيجمنت في إظهار الأرقام بهذه الطريقة وإنما على سبيل المثال يقوم الميكرو بقراءة درجة الحرارة مثلا ويقوم بتخزينها في متغير ... فما الأوامر التي يمكن كتابتها لتحويل هذه القيمة من المتغير إلى القيمة المركبة المستخدمة في المشاريع الأولية في هذا الفصل ؟؟؟

فخطوة قراءة درجة الحرارة سنتعرض لها في الفصول القادمة إن شاء الله، أما كيفية التعامل مع هذه المشكلة فهيا بنا نتدارس بشأنها ... بداية دعنا نستعرض بعض الأمور البرمجية في لغة السي والمتعلقة بحل هذه المشكلة ...

المتغيرات

من المعروف أننا نخزن أنواع مختلفة من البيانات في الذاكرة فقد نخزن قيم صحيحة مثلا ٢٥ أو قيم تحتوى على علامة عشرية مثلا ٢٥,٣ أو نخزن حروف أو نخزن جملة ... أو إلخ، وكل من هذه الأنواع يحتاج مساحات تخزين مختلفة، فمثلا لنقوم بتخزين حرف يتم حجز واحد بايت له في الذاكرة ولتخزين جملة لابد من حجز مساحة تتناسب مع عدد الحروف في الجملة ولتخزين قيمة صحيحة نحتاج مثلا ٢ بايت ولكى نخزن قيم قد تصل إلى اكتر من ٣٢ الف قد نحتاج إلى اكتر من ٢ بايت لتخزين

(*) أحمد سمير فايد

هذه القيم الكبيرة وهكذا ... والسؤال هنا هو كيف سيعرف المعالج أن يحجز ١ بايت أو ٢ بايت أو ١٠ بايت أو ... لهذا المتغير ؟؟ وذلك عن طريق سطر في الكود يسمى الإعلان عن المتغير أو تعريفه حيث تذكر في هذا السطر نوع المتغير ومنه يعرف المتغير الحجم اللازم للتخزين حيث أن كل نوع له مساحة معروفة لدي المعالج ... وبالطبع يتم إعطاء المتغير اسم للمتغير في هذا الإعلان وهو الذي يستخدم في باقي البرنامج:

لتعريف متغير صحيح:

```
int x;
```

حيث أن كلمة int هي الكلمة التي تمثل نوع المتغير وهي التي عن طريقها يعرف المعالج أن هذا المتغير من النوعية الصحيحة وبالتالي يحجز له مثلا ٢ بايت في الذاكرة، وأما الـ X فهي اسم المتغير الذي تم حجز له المكان في الذاكرة، فمثلا لو أردنا أن نقوم بتخزين القيمة ١٥٠ في هذه المتغير نكتبه كالآتي:

```
x = 150;
```

ولتعريف متغير آخر من النوع المعوم أي النوع الذي يحتوي على علامة عشرية وتخزين القيمة ١٥٠,٣ به نكتب الكود التالي:

```
float y;
y = 150.3;
```

ولتعريف متغير البت وهو متغير يستخدم لتخزين واحد بت فقط نكتب الآتي:

```
bit x;
x = 0;
```

وهذه القيمة في المتغير البت إما تكون صفر أو تكون واحد ولا يمكن أن تكون خمسة أو عشرة لأنها تخزن في بت واحد فقط.

لتعريف متغير من النوع الحرفي أي النوع الذي يستخدم لتخزين حرف وتخزين A فيه مثلا:

```
char z;
z = 'A';
```

الـ z يمثل اسم المتغير والحرف A هو عبارة عن الحرف المراد تخزينه في المتغير z، ويجب ملاحظة أنه عند تخزين حرف يتم وضعه بين single quotation كما هو موضح بالأمر وهما العلامتين ' '.

ملحوظة: يستخدم هذا النوع أيضا لتخزين قيم صحيحة ولكن لأنه يتم حجز ١ بايت فقط له فانه يخزن القيم من ٠ إلى ٢٥٥ فقط وهذه ملحوظة مهمة قد نحتاجها فيما بعد ...

لتعريف متغير سلسلة حرفية أي متغير لتخزين جملة وتخزين الجملة Welcome In Egypt.

```
char *str;
str = "Welcome In Egypt";
```

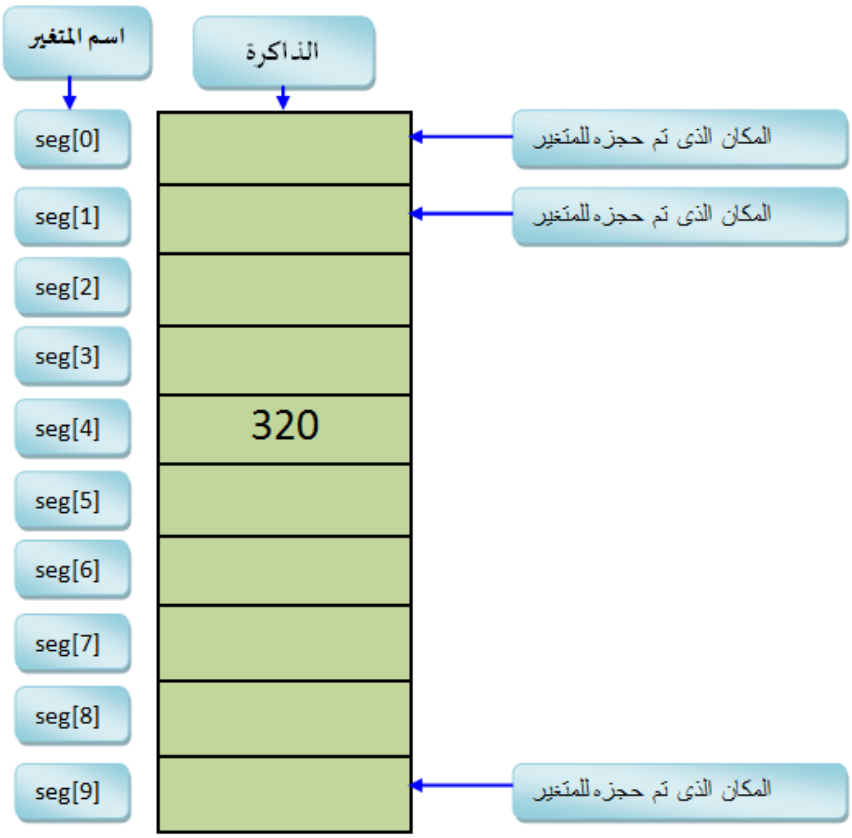
ويلاحظ في الجملة المطلوب تخزينها أنها وضعت بين Double Quotation أي بين العلامتين " ".

المصفوفات

لو افترضنا أننا نريد أن نخزن ١٠ قيم صحيحة فسوف نقوم بالإعلان عن ١٠ متغيرات من النوع int وذلك عن طريقة كتابة عشرة أوامر ... ولكن هناك طريقة أفضل وأسهل من ذلك وهي المصفوفات حيث يتم تعريف العشرة متغيرات في أمر واحد وهو كالآتي:

```
int seg[10];
```

ومن هذا الأمر يقوم المعالج بحجز ١٠ أماكن متتالية في الذاكرة (RAM) وكل مكان حجمه ٢ بايت ... ويتبقى معنا أن نعلم كيف لي أن أقوم بتخزين قيمة في المكان الخامس مثلا أو في المكان الأخير أو أي مكان آخر؟؟



وبالنظر للصورة المقابلة ونلاحظ أن ترقيم أسماء المتغيرات يبدأ من صفر وليس واحد وبالتالي فان العشرة متغيرات يأخذوا الأرقام من صفر إلى تسعة في أسمائهم فلو مثلا قلت لك اذكر اسم المتغير الثالث فان اسمه هو `seg[2]` ذلك لان المتغير الأول اسمه `seg[0]`.

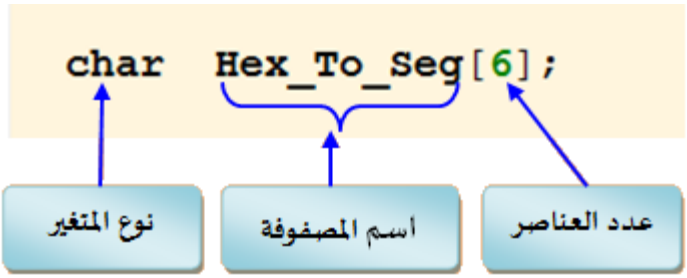
وبالتالي لو أردنا تخزين القيمة ٣٢٠ مثلا في المتغير الخامس سيكون الأمر كالتالي:

```
seg[4] = 320;
```

كيف سنوظف كل هذا فيما يخص التعامل مع السيفين سيجمنت ؟؟

فمثلا نريد كتابة برنامج يقوم بقراءة قيمة جهد معينة تتراوح ما بين صفر إلى خمسة فولت وتخزينها في متغير وعرضها على السيفين سيجمنت.

مبدئيا كم قيمة سنحتاجها؟ ستة قيم وهم الصفر والواحد والاثنين والثلاثة والأربعة والخمسة، وبالتالي سنقوم بكتابة أمر نعرف فيه مصفوفة من ستة عناصر وتكون من النوع char إذ أنه كما ذكر سابقا أن هذا النوع يستخدم في تخزين القيم الصحيحة من ٠ إلى ٢٥٥ ولن نحتاج لأكثر من هذا وبالتالي نوفر في الذاكرة وهذا الأمر بالشكر التالي:



والآن ما هي القيم التي سنقوم بتخزينها في هذه المصفوفة؟ تذكر معي جدول السيفين سيجمنت: لإظهار صفر نخرج 0x3F على البورت ولإظهار واحد نخرج 0x06 ولإظهار اثنين نخرج 0x5B ولإظهار ثلاثة نخرج 0x4F ولإظهار أربعة نخرج 0x66 ولإظهار خمسة نخرج 0x6D.

نقوم بتخزين القيم السابقة في المصفوفة وهي القيم التي سنقوم بإخراجها على PORTB وذلك عن طريق الأوامر الآتية:

```
Hex_To_Seg[0] = 0x3F;
Hex_To_Seg[1] = 0x06;
Hex_To_Seg[2] = 0x5B;
Hex_To_Seg[3] = 0x4F;
Hex_To_Seg[4] = 0x66;
Hex_To_Seg[4] = 0x6D;
```

ويمكن اختصار الأوامر الستة السابقة بالإضافة إلى خطوة الإعلان عن المصفوفة في سطر واحد كالتالي:

```
char Hex_To_Seg[6] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D};
```

بافتراض أن قيمة الجهد يتم قراءتها بطريقة ما لن نتطرق إليها الآن ثم تخزينها في متغير من النوع char أو int ... فنقوم بتعريف المتغير وليكن:

```
char x;
```

وسنخزن فيه القيمة ٣ بفرض أن الميكرو قرأها كقيمة للجهد وخزنها في المتغير:

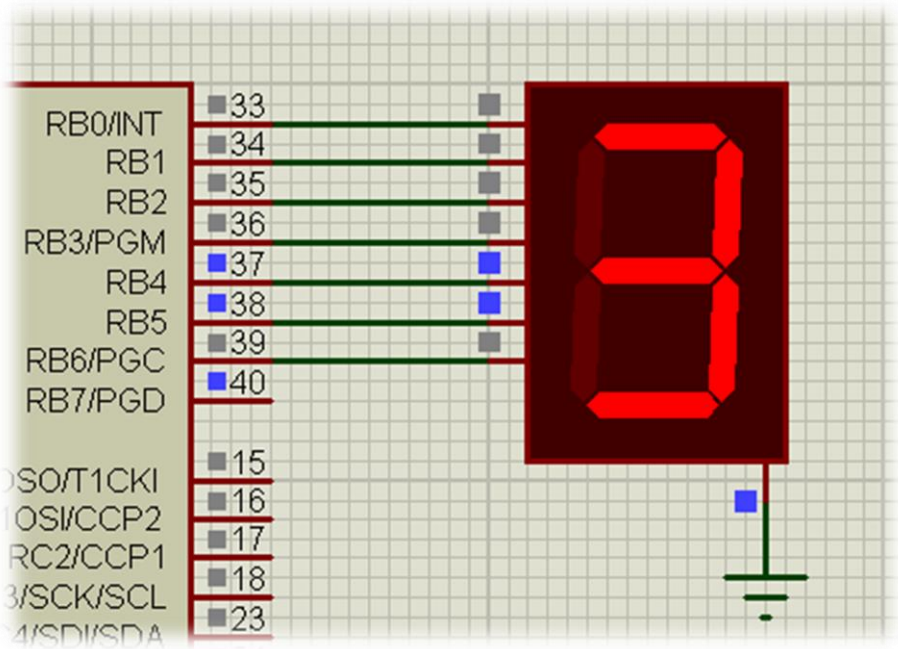
```
x = 3;
```

ويمكن بالتالي كتابة البرنامج كالتالي:

```
char  Hex_To_Seg[6] = {0X3F , 0X06 , 0X5B , 0X4F , 0X66 , 0X6D};

char x = 3;

void main()
{
   TRISB = 0B00000000;

   while(1)
   {
     PORTB = Hex_To_Seg[x] ;
   }
}
```

لتعريف المتغير وتخزين القيمة فيه بدل من كتاب أمرين

هذا الامر بيحذف قيمة x ويضع قيمتها والتى تساوى 3 ليصبح الامر كالاتى PORTB = Hex_To_seg[3]; والذى معناه خذ القيمة المخزنة فى العنصر Hex_To_seg[3] وأخرجها على PORTB وهذا العنصر يعتبر العنصر الرابع ويحتوى على القيمة 0X4F والتى تستخدم لاظهر القيمة 3



يمكنك تجربة البرنامج وتشغيله على برو تس بدون طبعا 7448IC أي بتوصيل السيفين سيجمنت بالميكرو مباشرة.

يمكنك أيضا تخزين قيمة أخرى في المتغير x وسوف ترى أن البرنامج يظهر نفس القيمة التي خزنتها.

مشاريع إضافية

وهناك أيضا طريقة أخرى فبدلا من استخدام المصفوفات يمكنك استخدام الدول ولكنى سأترك لكم هذه الجزئية تبحثوا عنها ... وأيضا أريدك أن تقوم بعمل مشروع تستخدم فيه اثنين سيفين سيجمنت أو ثلاثة ... ومشروع آخر لاستخدام اثنين سيفين سيجمنت على نفس مخرج الميكرو وإظهار عليهم قيم مختلفة وهو جزء هام لم أتطرق إليه هنا وذلك لان هذا يتطلب شرح التايمر والذى لم يدرج شرحه في هذا الكتاب لكنه سيكون إن شاء الله مدرجا في الجزء الثاني من هذا الكتاب ... وأيضا لقد قمت بالعمل على الأرقام من صفر إلى خمسة فقط في البرامج التي قمت بشرحها أما أنت فمن المفترض أن تعمل من صفر لتسعة.

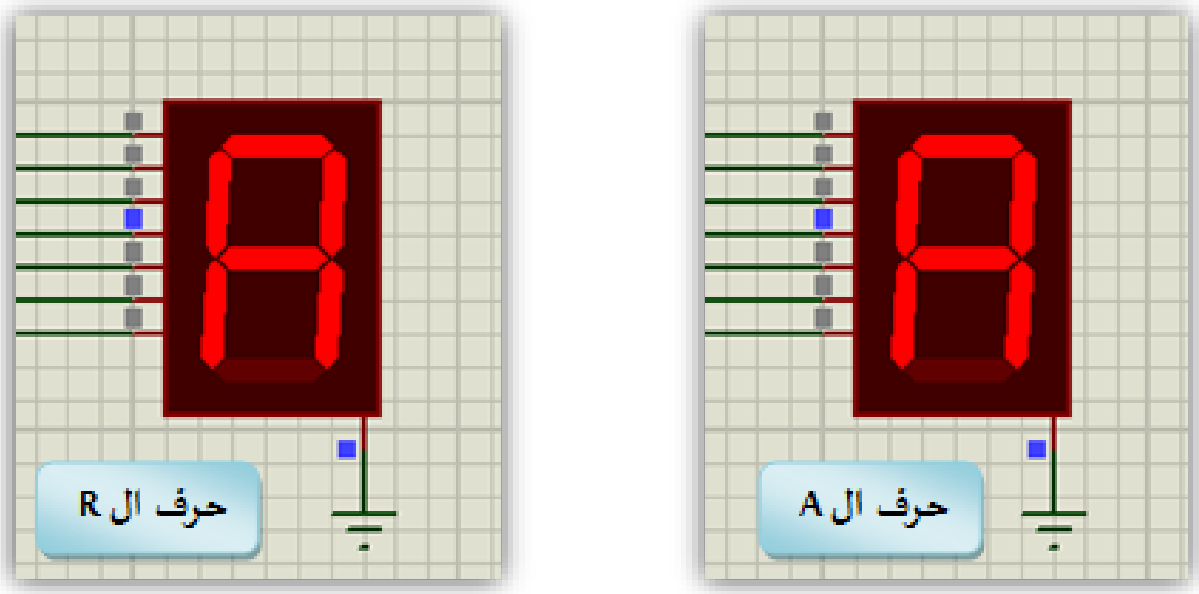
الفصل السابع

التعامل مع شاشات الـ LCD

في الفصل الماضي تعلمنا كيفية إظهار البيانات باستخدام السيفين سيجمنت والتي تحتوي على مجموعة من المشاكل، حيث يمكن التغلب عليها في هذا الفصل باستخدام الشاشات

درسنا في الفصل الماضي السيفين سيجمنت وتعلمنا كيفية التعامل معها وتوصيلها وفائدتها ولكن هناك مشاكل في السيفين سيجمنت ينبغي علينا معرفتها وخير مكان لمعرفتها هو بداية هذا الفصل فلنبدأ ...

١- لا تمكنني السيفين سيجمنت من عرض كل الحروف، فعلى سبيل المثال إذا حاولنا إضاءة ليدات معينة بحيث يظهر الحرف W فلن نستطيع ذلك، وأيضا هناك من الحروف التي تبدو متماثلة عند عرضها على السيفين سيجمنت فمثلا الحرف A والحرف R إذا أردنا عرضهم فسيظهرون بصورة طبق الأصل من بعضهما البعض كما بالشكل:

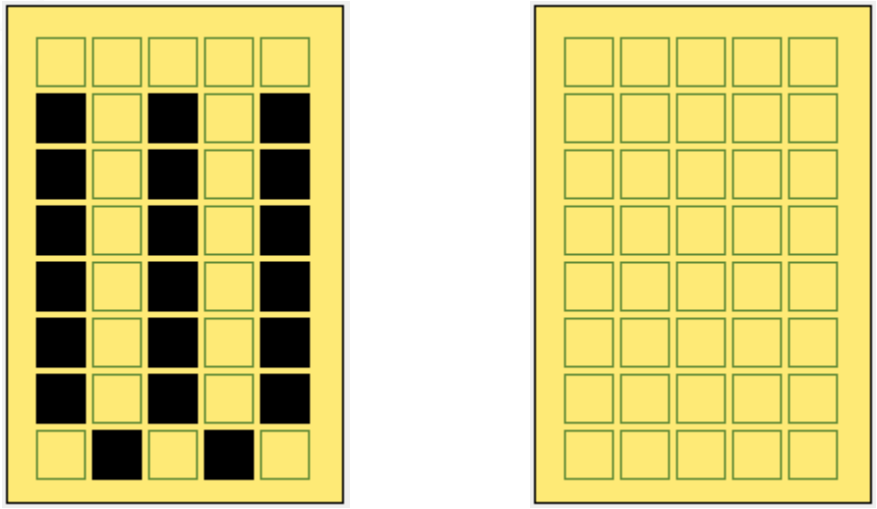


٢- إذا أردنا أن نقوم بعرض كلمة أو جملة بالسيفين سيجمنت فهذا يتطلب عدد من السيفين سيجمنت يساوى عدد الحروف الموجودة في الجملة، فمثلا لو أردنا أن نقوم بعرض كلمة Display وبافتراض عدم وجود المشكلة الأولى وأننا نستطيع عرض أي حرف بالسيفين سيجمنت فإننا سنحتاج إلى سبعة منها، كل واحدة لعرض حرف من الكلمة وهذا ينتج عنه صعوبة في عمل البرنامج وصعوبة في عمل الهاردوير.

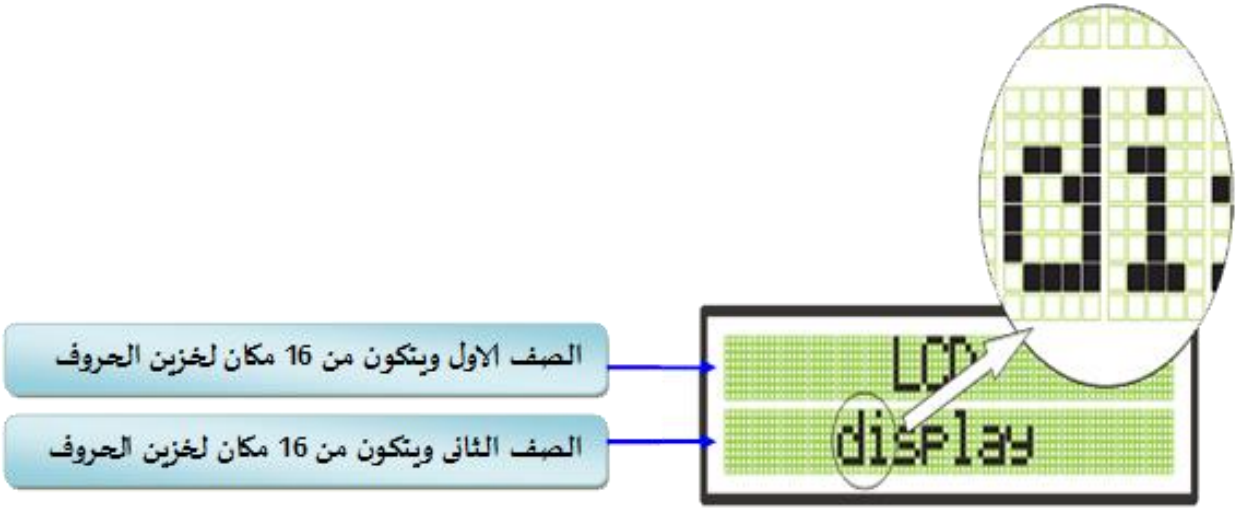
عن الـ LCD

ولذلك كان لابد من وضع حلول لهذه المشاكل وهي ما تمثلت في الـ LCD، كيف ذلك ؟؟

١- بالنسبة للمشكلة الأولى فتم التغلب عليها بزيادة عدد الليدات وتصغير حجمها وترتيبها بطريقة منتظمة كما بالشكل التالي بحيث تسمح بعرض أي حرف لم يكن متاح في السيفين سيجمنت، وبالتالي لنقوم بعرض نفس الحرف W كمثال فلن يكون ذلك صعبا، ويمكنك أيضا تجربة أي حرف آخر وتخيل شكله ومدى إتاحته من عدمه ...



٢- بالنسبة للمشكلة الثانية المتمثلة في عرض كلمة أو جملة فهذه المشكلة تم حلها أيضا عن طريق وضع عدد كبير من الجزء مصفوفة الليدات الموجودة في الصورة السابقة والمستخدمة لعرض حرف واحد وبذلك يمكن عرض مجموعة حروف متجاورة لنكون جملة أو كلمة داخل الـ LCD.

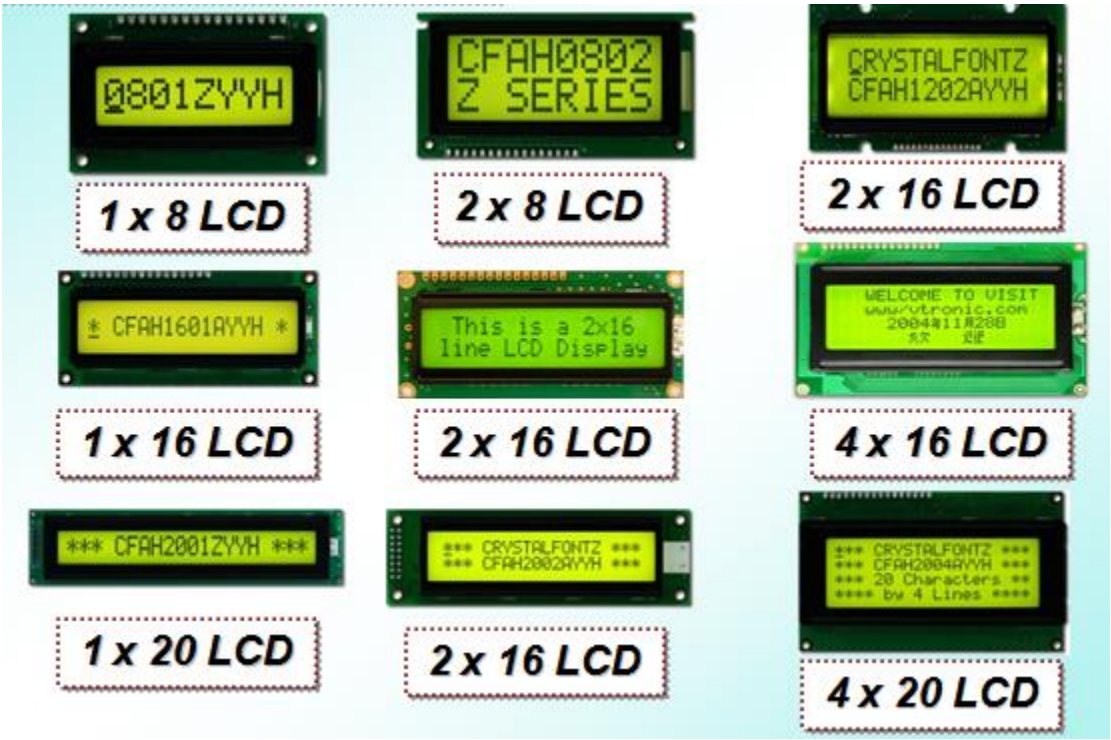


أغلب الـ LCDs تحتوي على أكثر من صف وكل صف يتكون من أكثر من مكان لعرض الحروف ...

أنواع الـ LCD

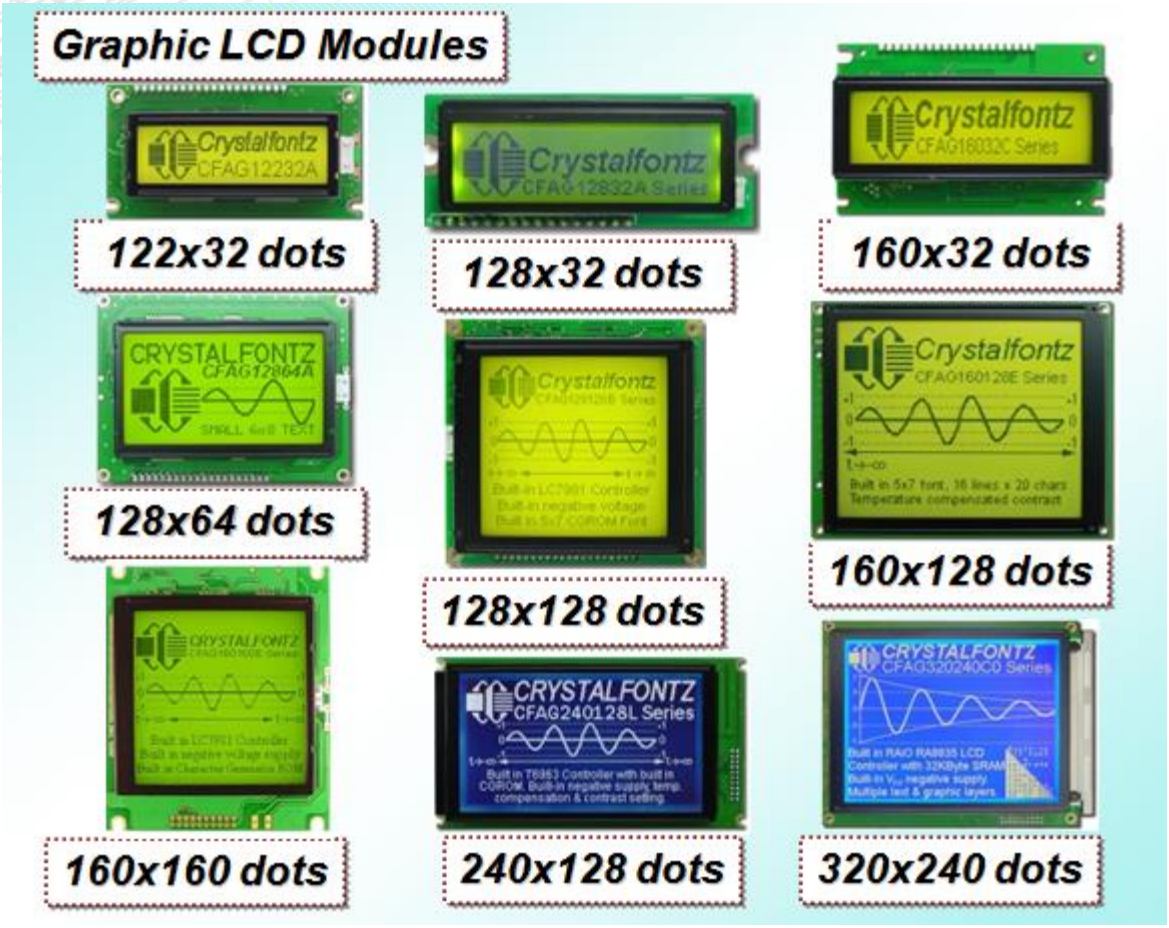
Character LCD

النوع الذي تحدثنا عنه منذ قليل يسمى Character LCD حيث يستخدم لعرض الحروف، وله أشكال مختلفة يختلف كل شكل عن الآخر في عدد السطور وعدد الحروف داخل كل سطر، فمثلا توجد أنواع بها صف واحد وهذا الصف يستوعب ثمانية أحرف ويوجد أنواع تحتوي على صفين وكل صف يستوعب ثمانية أحرف وغير ذلك كما هو مبين



Graphical LCD

نوع آخر من أنواع الـ LCD وهو الـ Graphical LCD الذي يستخدم لعرض الرسومات البيانية، وهو لا ينقسم إلى أجزاء كالنوع السابق وإنما يشبه في طريقة عمله شاشة الكمبيوتر حيث تكون عبارة عن وحدة واحدة بها عدد كبير من الـ Pixels مرتبة في صفوف وأعمدة ويمكنك إظهار أي شكل تريده عن طريقة التحكم في إضاءة أو إطفاء أي بيكسل وبترتيب البيكسل المضاءة والبيكسل المطفأة نحصل على الشكل المطلوب رسمه، ويوجد منها أحجام أيضا ولكن لا يتم التعبير عنها بعدد السطور وعدد الحروف في السطر وإنما بعدد البيكسل الأفقي مضروبا في عدد البيكسل الرأسي كما في الشكل التالي:



سوف يكون شغلنا في هذا الفصل على الـ Character LCD فقط واخترنا شاشة بسيطة تحتوي على صفين وكل صف فيه ١٦ حرف أي وتكتب هكذا 2*16.

توصيل الـ LCD بالميكروكنترولر

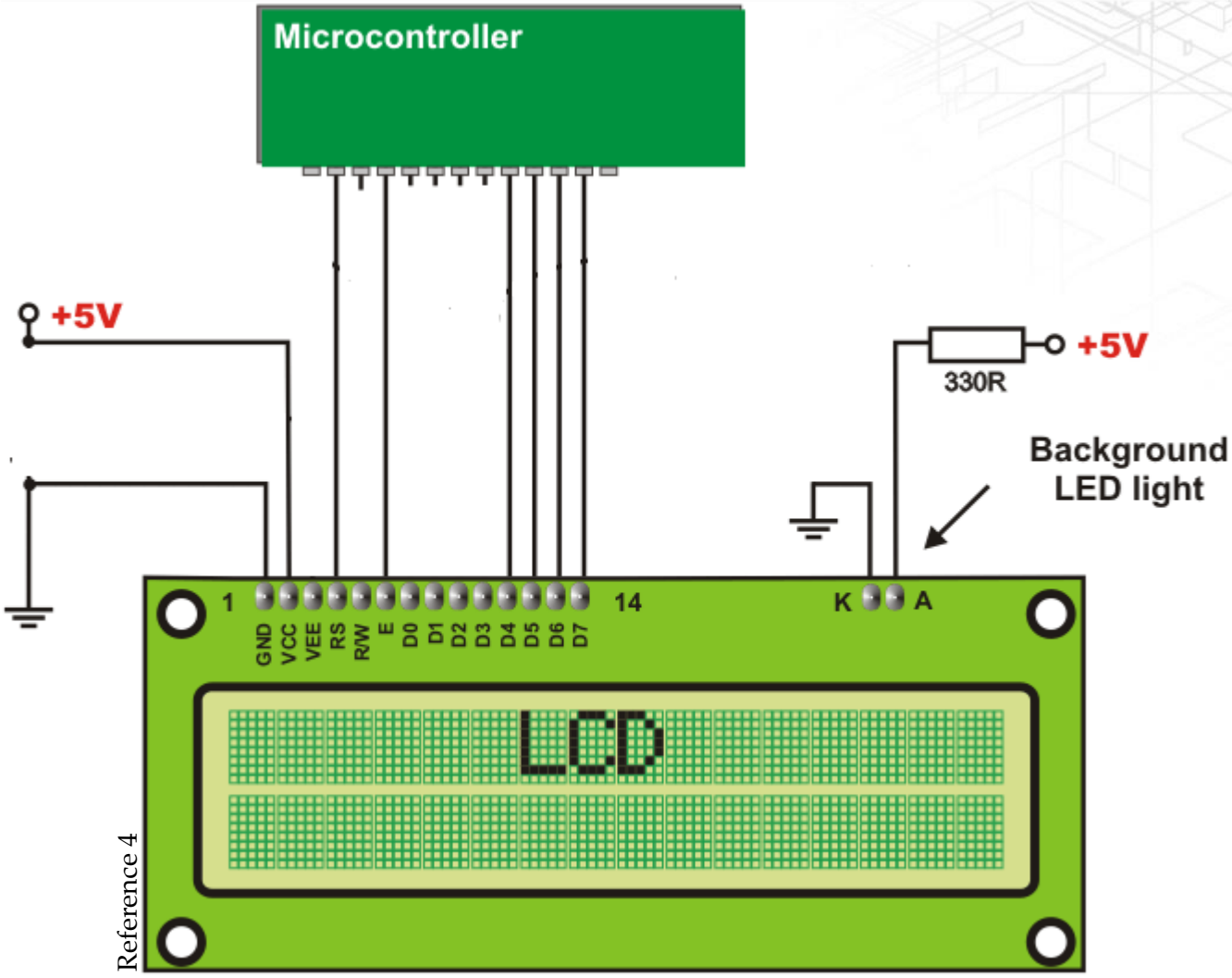
توجد في غالبية الشاشات ١٦ طرف (رجل) مرقمين من واحد إلى ١٦ والذي نحتاجه فقط منها هو الآتي:

١) يوجد أربعة رجول لنقل بيانات العرض والأوامر من الميكرو للـ LCD وأرقامهم هي ١١ و ١٢ و١٣ و ١٤ ويكون مكتوب عليهم في هاردوير الـ LCD الآتي: D4 , D5 , D6 , D7.

٢) طرفين أخرين:

أ- الطرف RS: ويستخدم لكي يخبر المايكرو الشاشة عن طريقه ما إذا كان سيتم نقل أمر أم سيتم نقل بيانات الآن، والأوامر مثل (مسح الشاشة) والداتا المقصود بها الكلام المرسل للشاشة لتعرضه.

بـ الطرف EN: ويستخدم لجعل الـ LCD مستعدة لاستقبال البيانات وإظهارها.



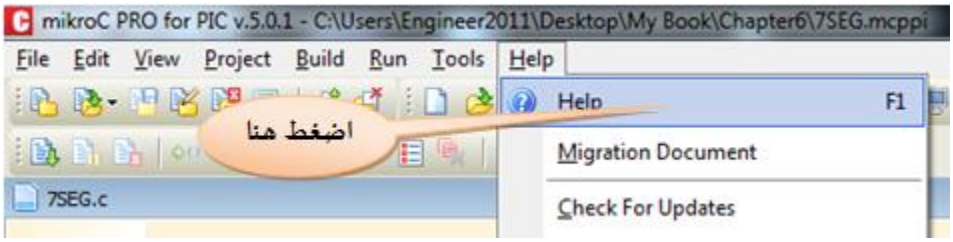
ما سبق من رجول هو ما يخص التوصيل بين الميكرو والـ LCD، ولكن هناك أطراف أخرى للـ LCD لابد من توصيلها ولكن ليس بالميكرو وهم:

- الطرف رقم ٢: ويوصل بالخمسة فولت.
- الأطراف ١ و ٣ و ٥: ويوصلوا جميعا بالأرضي.
- الطرفين ١٥ و ١٦: ويستخدمان لإضاءة ليد موجود في الشاشة، حيث تفيد هذه الليد حتى نتمكن من رؤية البيانات المعروضة جيدا، يتم توصيل الطرف ١٥ على الخمسة فولت والطرف ١٦ على الأرضي.

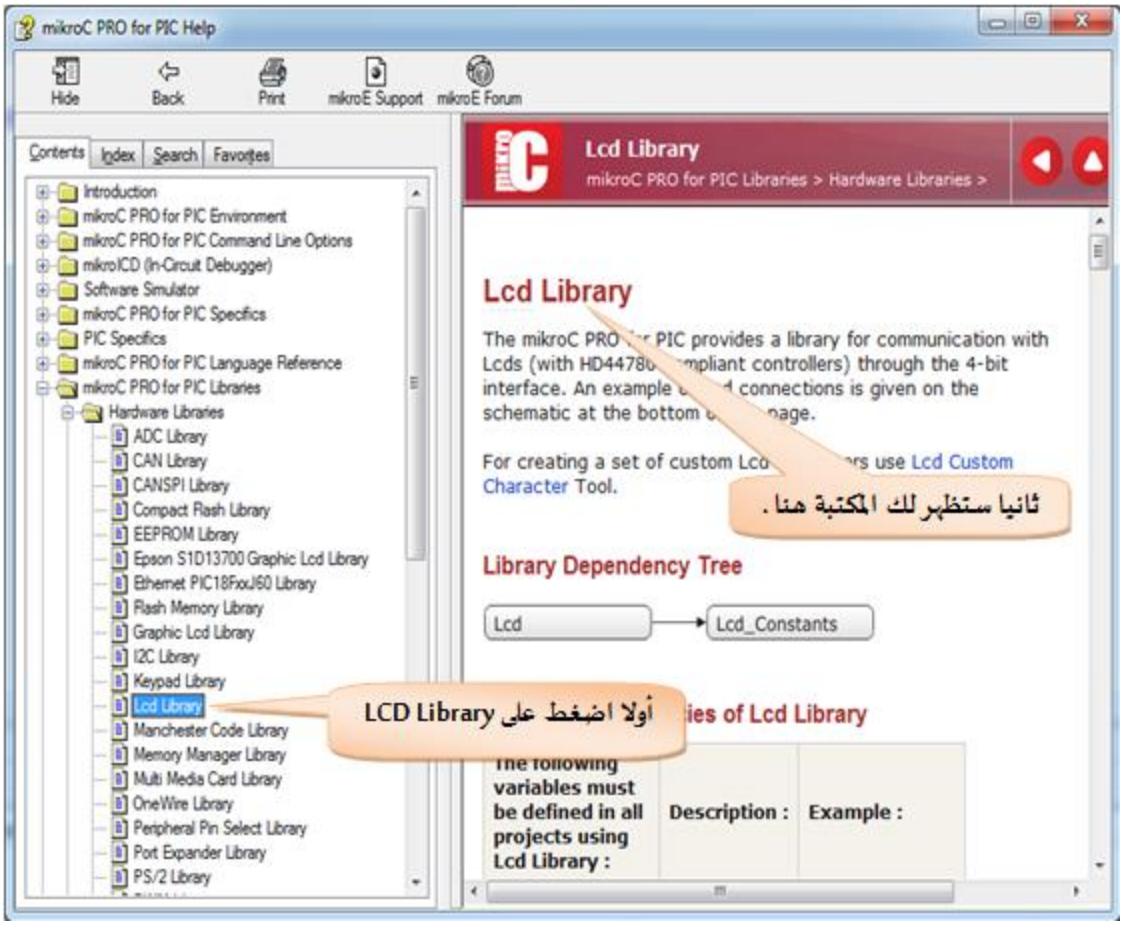
الدوال المستخدمة مع الشاشات

في هذا الفصل سنبدأ بتعلم طريقة جديدة في التعامل مع دوال الميكرو سي المستخدمة في المواضيع المختلفة وهي نافذة المساعدة (Help) في برنامج الميكرو سي وهي طريقة هامة جدا جدا وسيتبين ذلك من خلال الشرح في هذا الفصل وفى الفصول القادمة ...

للدخول على نافذة المساعدة اتبع الصورة التالية:



فتظهر نافذة المساعدة ... اتبع خطوات الصورة التالية لتدخل مكتبة الدوال الخاصة بالتعامل مع شاشات الـ LCD:



دعونا نتناول نافذة المساعدة الخاصة بالـ LCD بشيء من التفصيل وذلك على سبيل شرح كيفية استخدام نافذة المساعدة لأي شيء آخر:

الجزء الثاني في الصورة يشير إلى الحاجات الخارجية التي تعتمد عليها هذه المكتبة ... بالطبع كلام مش مفهوم ... أنا عن نفسي مش فاهمه ☺ ... تعالوا نفسر هذا الكلام.

تتعامل هذه المكتبة مع الشاشات وبالتالي سيكون فيها دوال تقوم بإظهار حروف أو قيم على الشاشة كدرجة الحرارة مثلا، ودالة أخرى تقوم بمسح الشاشة، ودالة أخرى تقوم بتحريك الكلام الموجود على الشاشة لليمين أو لليسار ... إلخ، تستخدم هذه الدوال داخل برنامج الميكرو بالطبع، ولكن السؤال يقول: الدالة التي تقوم بإرسال القيمة لتظهر على LCD مثلا ستقوم بإرسالهم على أي بورت؟؟؟ هل البورت PORTB أم البورت PORTC أم غيرهما؟؟ بالطبع على البورت المتصل بالشاشة، ولكن كيف تعرف هذه الدوال هذا البورت ؟؟؟ هذا ما نود الإجابة عليه في هذه الجزء ...

المجموعة الأولى من المتغيرات

لنتذكر معا كم طرف يوصل بين الشاشة والميكرو؟ ستة أطراف منهم أربعة فقط لنقل الأوامر والداتا والاثنين الآخرين لن نتطرق إلى تفصيلهم ... وبالتالي كان الاقتراح الذي تم تنفيذه في برنامج السي هو وجود ستة متغيرات كل متغير يقابل طرف معين من أطراف الشاشة وهم:

الطرف المختص به في الشاشة	اسم المتغير
مختص بالطرف RS	LCD_RS
مختص بالطرف EN	LCD_EN
مختص بالطرف D4	LCD_D4
مختص بالطرف D5	LCD_D5
مختص بالطرف D6	LCD_D6
مختص بالطرف D7	LCD_D7

ويمكن استنباط الطرف من اسم المتغير بكل سهولة والعكس صحيح ...

ولإعلام الميكرو بالرجول المتصلة بالشاشة نقوم بوضع أسمائها في هذه المتغيرات عن طريق مجموعة أوامر مهمة جدا كالآتي:

```
sbit LCD_RS at RC2_bit;
```

ومن هذا الأمر تعرف الدوال أن الطرف RS الموجود في الشاشة متصل بالرجل رقم ٢ في المخرج C، وماذا عن باقي الأطراف؟ نفس الطريقة:

```
sbit LCD_EN at RC3_bit;
```

هذا الأمر تعرف منه الدوال أن الطرف EN الموجود في الشاشة متصل بالرجل رقم ٣ في المخرج C، وماذا عن باقي الأطراف التي تستخدم في نقل البيانات والأوامر؟

```
sbit LCD_D4 at RC4_bit;
```

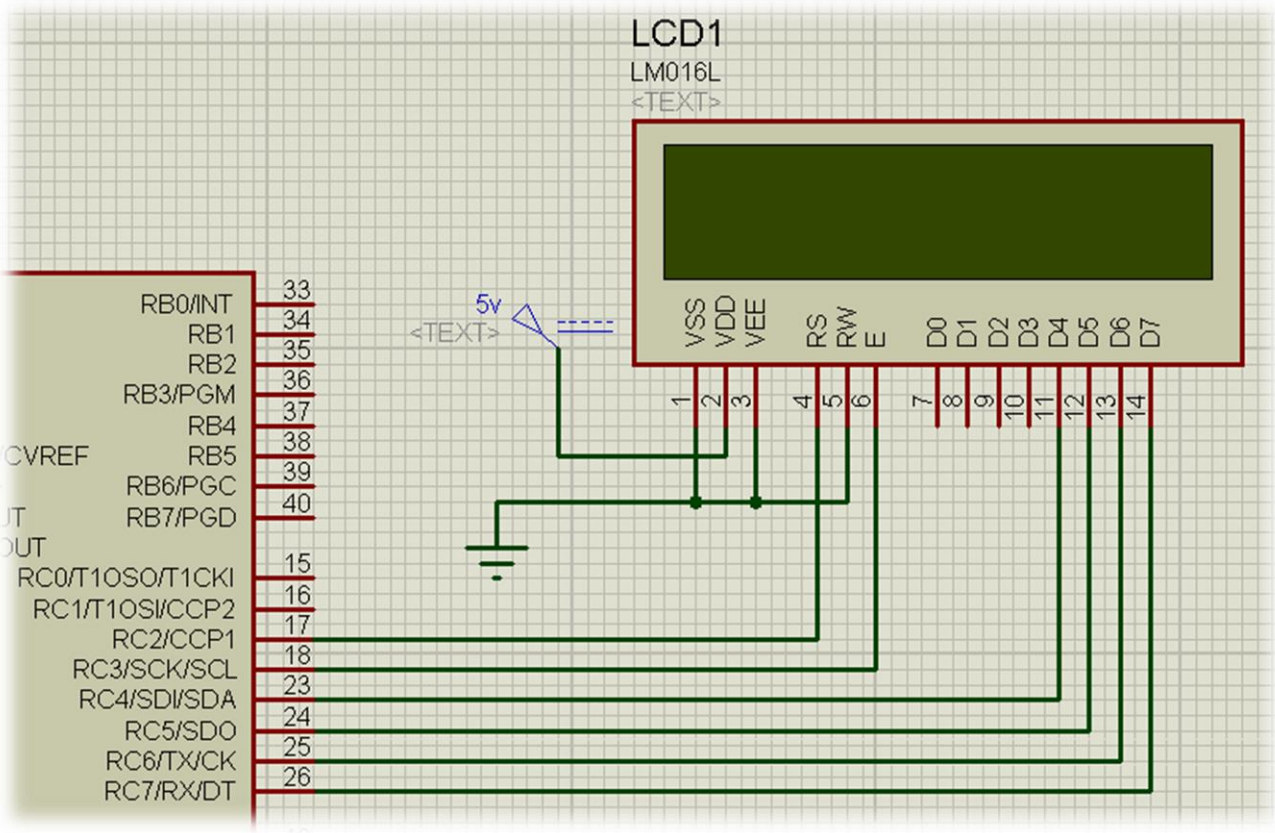
وهذا معناه أن الطرف المسمى D4 في الشاشة وهو الطرف رقم ١١ فيها متصل بالرجل رقم ٤ في المخرج C، وهكذا باقي الأطراف كما هو موضح ... وبالتالي تكون الستة أوامر كالآتي:

```
sbit LCD_RS at RC2_bit;
sbit LCD_EN at RC3_bit;
sbit LCD_D4 at RC4_bit;
sbit LCD_D5 at RC5_bit;
sbit LCD_D6 at RC6_bit;
sbit LCD_D7 at RC7_bit;
```

ملحوظة: هذه الأوامر يتم كتابتها في بداية كل برنامج يتعامل مع شاشة وقبل الدالة الرئيسة له كما سنرى إن شاء الله.

أيضا: في بعض الإصدارات الأخرى للميكرو سي قد لا تجد هذه الأوامر وتجد دالة من خلالها يتم تحديد أطراف التوصيل بين الشاشة والميكرو ... لذا يجب استخدام نفس الإصدار المستخدم في الكتاب.

بعد عمل البرنامج على الميكرو سي وعمل Build له ثم الانتقال إلى المحاكاة على بروتس وتوصيل الشاشة بالميكرو فلابد من الرجوع إلى هذه الستة أوامر التي قمت بكتابتها من قبل والتوصيل على أساسها، وبالتالي يكون شكل الدائرة في بروتس بناء على الستة أوامر السابقة كالتالي:



ستلاحظ في الصورة أن الطرف D7 في الشاشة متصل بالطرف C7 في الميكرو لان الأمر المكتوب ينص على ذلك، وستجد أيضا أن الطرف RS وهو الرجل رقم ٦ في الشاشة متصل بالطرف C3 في الميكرو وهذا فعلا ما قمنا بتحديده في الأوامر.

وبكدة نكون انتهينا من أول مجموعة أوامر نحتاجهم عند التعامل مع الشاشات ...

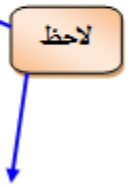
المجموعة الثانية من المتغيرات

لو تتذكر أننا كنا سابقا قبل أن نقوم بإخراج قيم على رجل من رجول الميكرو لابد أن نحدد اتجاه الداتا على هذه الرجل باستخدام الأمر TRIS ... وبالتالي لابد أيضا من تحديد اتجاه الداتا على الأرجل المستخدمة مع الشاشة ... هذا أيضا سيكون بستة أوامر مشابهة للأوامر السابقة وهي كالاتي:

```
sbit LCD_RS_Direction at TRISC2_bit;
sbit LCD_EN_Direction at TRISC3_bit;
sbit LCD_D4_Direction at TRISC4_bit;
sbit LCD_D5_Direction at TRISC5_bit;
sbit LCD_D6_Direction at TRISC6_bit;
sbit LCD_D7_Direction at TRISC7_bit;
```

نلاحظ في هذه الأوامر أننا نتعامل مع نفس الرجول التي تعاملنا معها في الستة أوامر السابقة وهذا مهم جدا ... وعندئذ تصبح الأوامر كلها بالشكل الآتي:

```
sbit LCD_RS at RC2_bit;
sbit LCD_EN at RC3_bit;
sbit LCD_D4 at RC4_bit;
sbit LCD_D5 at RC5_bit;
sbit LCD_D6 at RC6_bit;
sbit LCD_D7 at RC7_bit;

sbit LCD_RS_Direction at TRISC2_bit;
sbit LCD_EN_Direction at TRISC3_bit;
sbit LCD_D4_Direction at TRISC4_bit;
sbit LCD_D5_Direction at TRISC5_bit;
sbit LCD_D6_Direction at TRISC6_bit;
sbit LCD_D7_Direction at TRISC7_bit;
```



الدوال

الآن حان الوقت لنتعرف على الدوال التي تستخدم مع الشاشة:

الدالة الأولى

```
Lcd_Init();
```

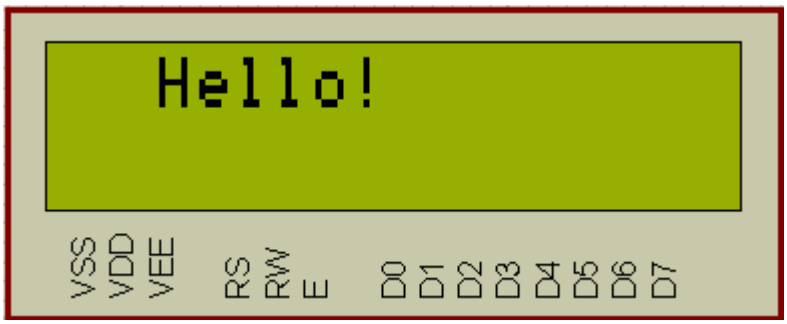
ويتم كتابتها داخل الدالة الرئيسية وقبل الـ while وفائدتها تجهيز الموديول الذي يتعامل مع الشاشة داخل الميكرو ...

الدالة الثانية

وهي دالة تستخدم لإخراج سلسلة حرفية أو جملة على الشاشة وهي كالآتي:

```
Lcd_Out(1, 3, "Hello!");
```

رقم الصف — رقم العمود — الكلمة التى ستظهر



فمثلا عند استخدام هذا الأمر على الصورة السابقة ستظهر الكلمة كما بالشكل:

ولو استخدمناه بالشكل التالي مثلا:

```
Lcd_Out(1,8,"Hello !");
```

فسيكون الخرج كالآتي:

ولو استخدمنا نفس الأمر لكن اخترنا الصف الثاني كما بالشكل:

```
Lcd_Out(2,8,"Hello !");
```

فسيكون الخرج كالآتي:

أتمنى أن تكون هذه الدالة مفهومة ...

الدالة الثالثة

لكن قبل كتابتها اذكر لكم مثال بسيط تفهمونها من خلاله ... في برنامج الورد (word) عندما كنا تكتب مجموعة كلمات تجد انه بعد آخر حرف يوجد Cursor يظهر و يختفى ... فاذا أردت أن تكمل الكتابة فسيظهر ما ستكتبه بعده، لكن ماذا إذا كنت تريد أن تكتب في مكان آخر غير هذا السطر الذي تقف عنده لابد أن تذهب بالماوس وتضغط في المكان الذي تريد أن تكتب به وعندها يظهر الـ Cursor في هذا المكان ومن ثم تبدأ الكتابة ...

وبالمثل فإن هذه الدالة تستخدم للكتابة عند آخر حرف انتهينا منه وبالتالي فلن نحتاج أن نقوم بتحديد الصف والعمود لها:

```
Lcd_Out_Cp("Here!");
```

وبالتالي لو كتبنا الأمرين الآتيين:

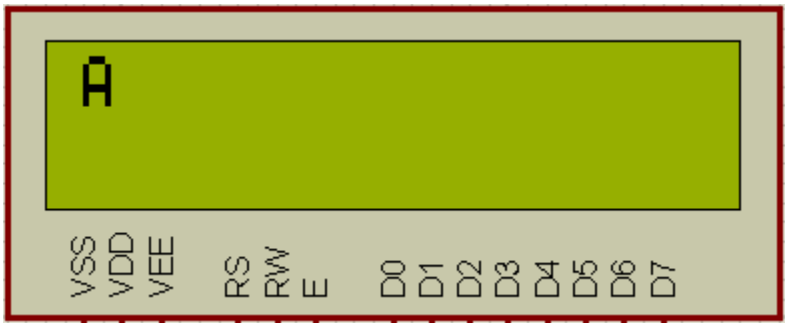
```
Lcd_Out(1,1,"Hello!   ");
Lcd_Out_Cp("Here!");
```

سيكون الخرج كالآتي:

الدالة الرابعة

وهي دالة تستخدم لإظهار حرف واحد على الشاشة، وتأخذ نفس صيغة الدالة التي تخرج السلسلة كالآتي:

```
Lcd_Chr(1,1,"A");
```



الخرج سيكون كما بالشكل:

وبالطبع لو غيرنا الصف أو العمود في هذه الدالة سيتغير مكان ظهور الحرف.

الدالة الخامسة

وهذه هي الدالة الأخيرة – التي سنتعرض لها – وهي الدالة التي تستخدم لنقل أوامر للشاشة لكي تقوم بتنفيذها ... وتكون بالشكل الآتي:

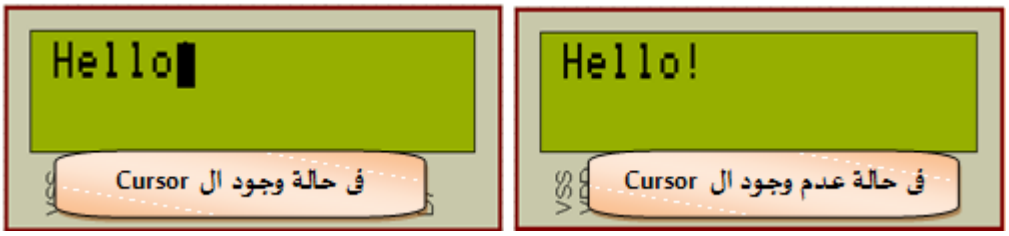
```
Lcd_Cmd(      يكتب الأمر هنا      );
```

حيث نقوم بكتابة الأوامر بين أقواس هذه الدالة، فمثلا لمسح الشاشة نكتب الآتي:

```
Lcd_Cmd( _LCD_CLEAR );
```

أيضا الأمر الذي يستخدم في إلغاء الـ Cursor:

```
Lcd_Cmd( _LCD_CURSOR_OFF );
```



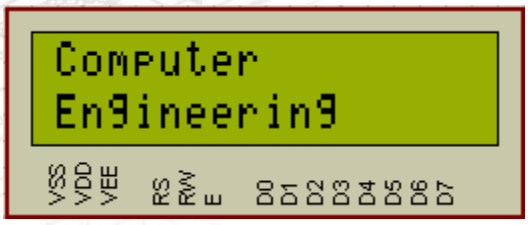
والآن كيف لنا أن نعرف الأوامر المتاح تنفيذها على الشاشة ؟؟

الأوامر موجودة في نافذة المساعدة وكذلك كل الدوال أيضا ويمكنك الاطلاع عليها، وهذا الجدول من نافذة المساعدة يحتوي على كل الأوامر التي تعطى للشاشة ...

Available Lcd Commands

Lcd Command	Purpose
_LCD_FIRST_ROW	Move cursor to the 1st row
_LCD_SECOND_ROW	Move cursor to the 2nd row
_LCD_THIRD_ROW	Move cursor to the 3rd row
_LCD_FOURTH_ROW	Move cursor to the 4th row
_LCD_CLEAR	Clear display (الأمر المستخدم في مسح الشاشة)
_LCD_RETURN_HOME	Return cursor to home position, returns a shifted display to its
_LCD_CURSOR_OFF	Turn off cursor
_LCD_UNDERLINE_ON	Underline cursor on
_LCD_BLINK_CURSOR_ON	Blink cursor on
_LCD_MOVE_CURSOR_LEFT	Move cursor left without changing display data RAM
_LCD_MOVE_CURSOR_RIGHT	Move cursor right without changing display data RAM
_LCD_TURN_ON	Turn Lcd display on
_LCD_TURN_OFF	Turn Lcd display off
_LCD_SHIFT_LEFT	Shift display left without changing display data RAM
_LCD_SHIFT_RIGHT	Shift display right without changing display data RAM

المثال التطبيقي الأول



نريد أن نقوم بتنفيذ مشروع يعرض كلمة Computer على السطر الأول من الشاشة ويعرض كلمة Engineering على السطر الثاني كما يظهر في هذه الصورة.

بعد شرح الأوامر سابقا سنكتب شكل أي برنامج يتعامل مع الشاشة:

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

void main()
{
  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);

    while(1)
     {

     }
}
```

حيث يبدأ يوجد بالاثنتي عشر أمرا المستخدمين في توضيح أطراف التوصيل بالإضافة لدالة الـ initialization والتي تكتب داخل الدالة الرئيسية، ودالة أخرى لإلغاء الـ Cursor ثم بعد ذلك تكتب الدوال التي تستخدم لإظهار ما تريده على الشاشة ... وفى مثالنا نريد اظهرا كلمة كمبيوتر على أول سطر فيكون الأمر كالتالي:

```
Lcd_Out(1,1,"Computer");
```

ونريد إظهار الكلمة الأخرى في السطر الثاني فيكون الأمر كالآتي:

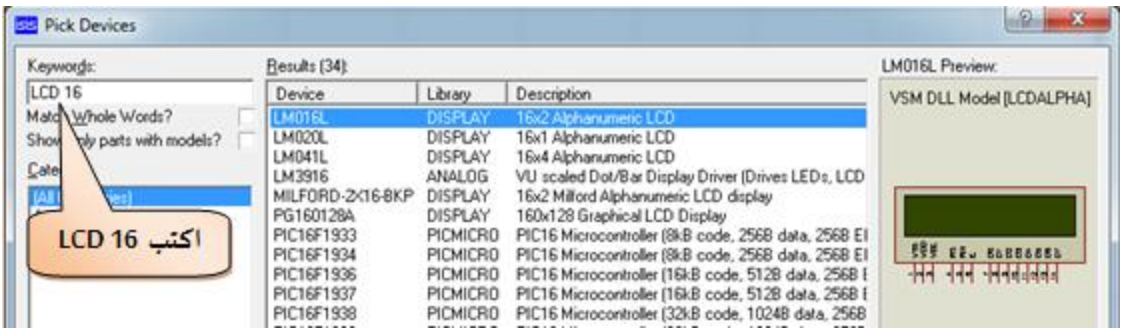
```
Lcd_Out(2,1,"Engineering");
```

ليصبح الشكل النهائي للبرنامج كالتالي:

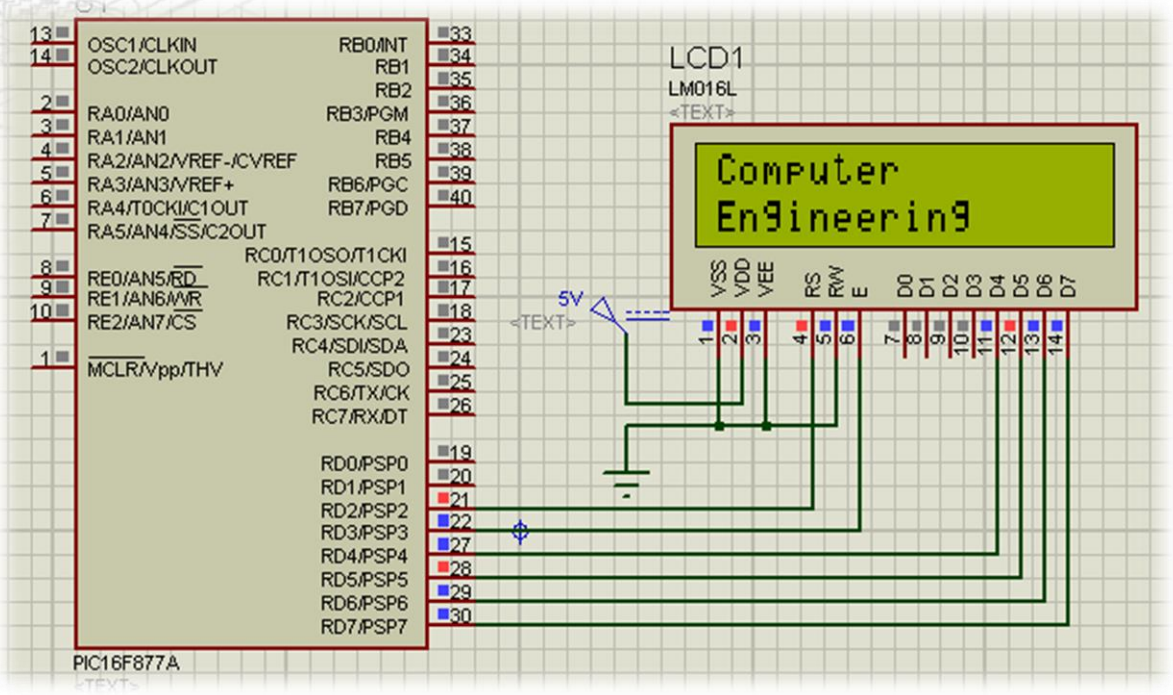
```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

void main()
{

  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);

    while(1)
     {
       Lcd_Out(1,1,"Computer ");
       Lcd_Out(2,1,"Engineering");
     }
}
```

لاحظ أنى قد اخترت PORTD فى الميكرو لتوصيل

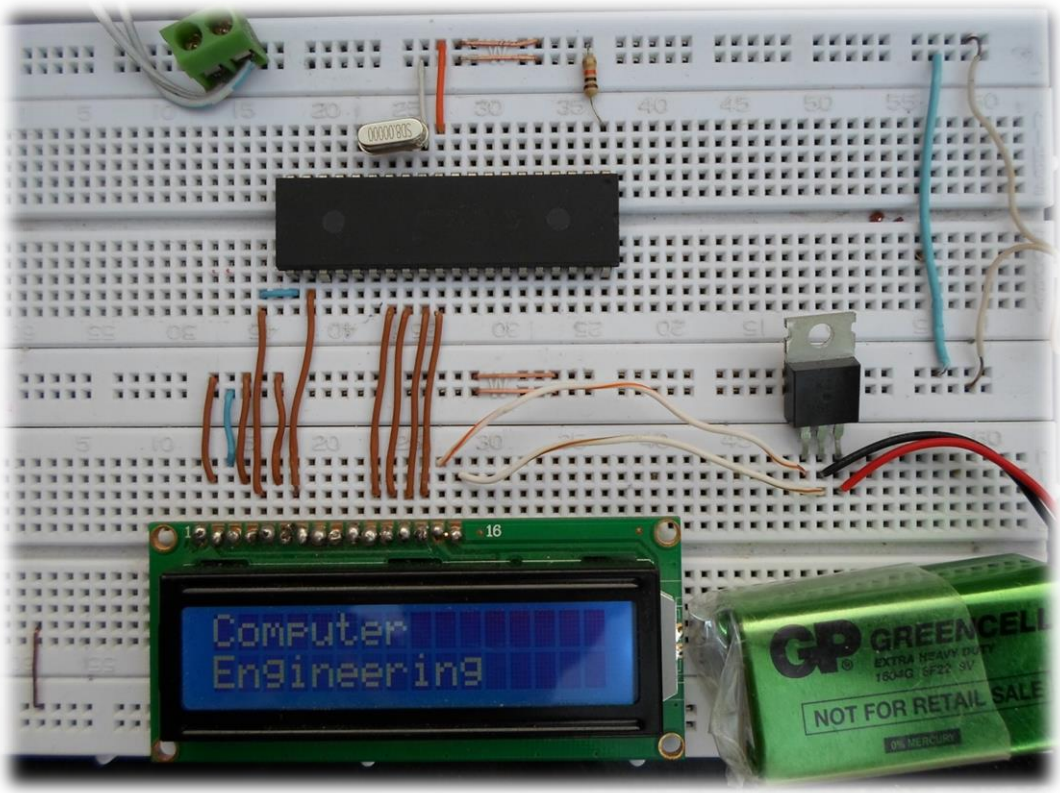
يمكنك تنزيل الشاشة في بروتس كما بالشكل الآتي:

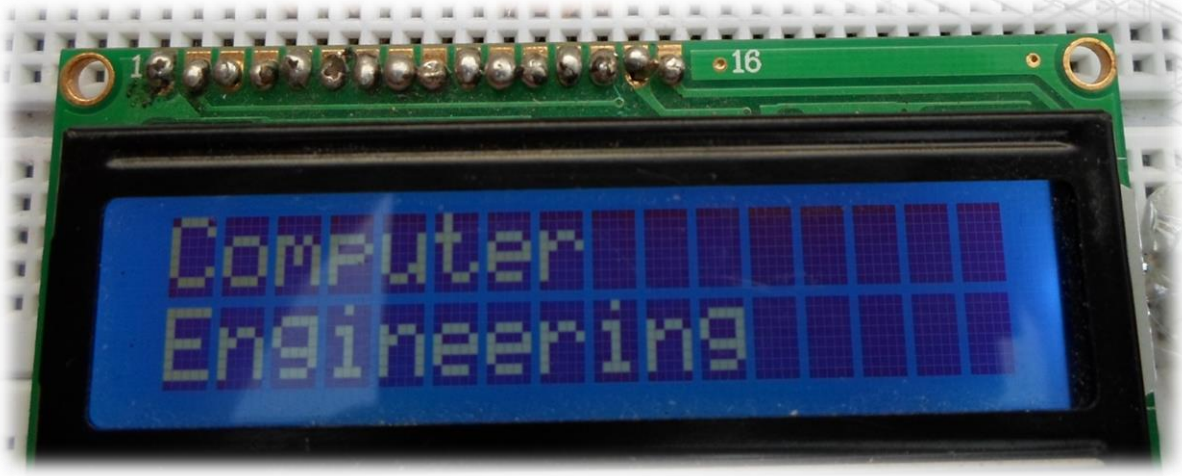


ويمكنك الآن توصيل الدائرة وتشغيلها كما بالشكل:



الهاردوير





المثال التطبيقي الثاني

المطلوب: عرض كلمة DISPLAY على الشاشة وجعلها تسير من اليسار إلى اليمين.

فكرة المشروع: هو أن نقوم بعرض الكلمة في أول السطر ثم نقوم بمسحها وعرضها بداية من الموضع الثاني ثم مسحها بعد زمن صغير وعرضها بداية من الموضع الثالث وهكذا حتى نصل إلى الموضع ١٦ فتظهر الكلمة وكأنها تتحرك ... ويتوقف مدى سرعة تحركها على الزمن الذي تحدده أنت عندما تعرض وتمسح الكلمة ...

لعرض الكلمة بداية من أول الصف نكتب الأمر كالآتي:

```
Lcd_Out(1,1,"DISPLAY");
```

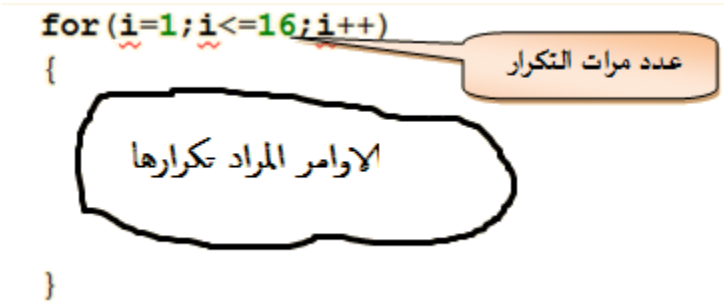
ولعرضها بداية من الموضع الثاني نكتب الأمر الآتي:

```
Lcd_Out(1,2,"DISPLAY");
```

ولعرضها بداية من الموضع الثالث نكتب الأمر الآتي:

```
Lcd_Out(1,3,"DISPLAY");
```

وهكذا حتى نصل إلى الموضع السادس عشر ... وطبعا لن نكتب الأمر ١٦ مرة فلابد من استخدام FOR كما تعلمناها من قبل.



وليصبح البرنامج هكذا بدون الـ ١٢ أمر الخاصين بالتوصيل والذين ينبغي عليك كتابتهم بالتأكيد:

```
int i;

void main()
{
  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);

    while(1)
     {
       for(i=1;i<=16;i++)
        {
           Lcd_Out(1,i,"DISPLAY");
              delay_ms(50);
           Lcd_Cmd(_LCD_CLEAR);
              delay_ms(50);

        }
     }
}
```

هذا متغير صحيح نحتاجه للحلقة التكرارية FOR

لاظهار الكلمة على الشاشة كل مرة فى مكان مختلف لان رقم العمود هو i وهو يتغير فى كل عملية تكرار من واحد الى 16

لاظهار الكلمة لمدة 50 ميللى ثانية

لمسح الشاشة و الانتظار هكذا لمدة 50 ميللى ثانية

المثال التطبيقي الثالث

قبل الخوض في هذا المثال هناك ملحوظة هامة لابد من ذكرها، بافتراض وجود متغير سلسلة حرفية ومخزن فيه كلمة وليكن كالآتي:

```
Char* str1 = "Hamdy";
```

فإذا أردنا إظهار هذا المتغير على الشاشة يكون شكل الأمر كالتالي:

```
Lcd_Out(1,1,str1);
```

لاحظ في الأمر السابق أنه لم يكتب هكذا

```
Lcd_Out(1,1,"str1");
```

فبهذه الطريقة ستظهر على الشاشة الكلمة str1 لكننا نريد إظهار ما بداخل المتغير الذي يسمى str1 فنضع هذا الاسم بدون علامات التنصيص " " هذه، وهذه نقطة مهمة جدا يمكنكم تجربتها في المشاريع السابقة مع العلم أنكم ستحاجون إلى بتعريف المتغير str1 قبل الدالة الرئيسية.

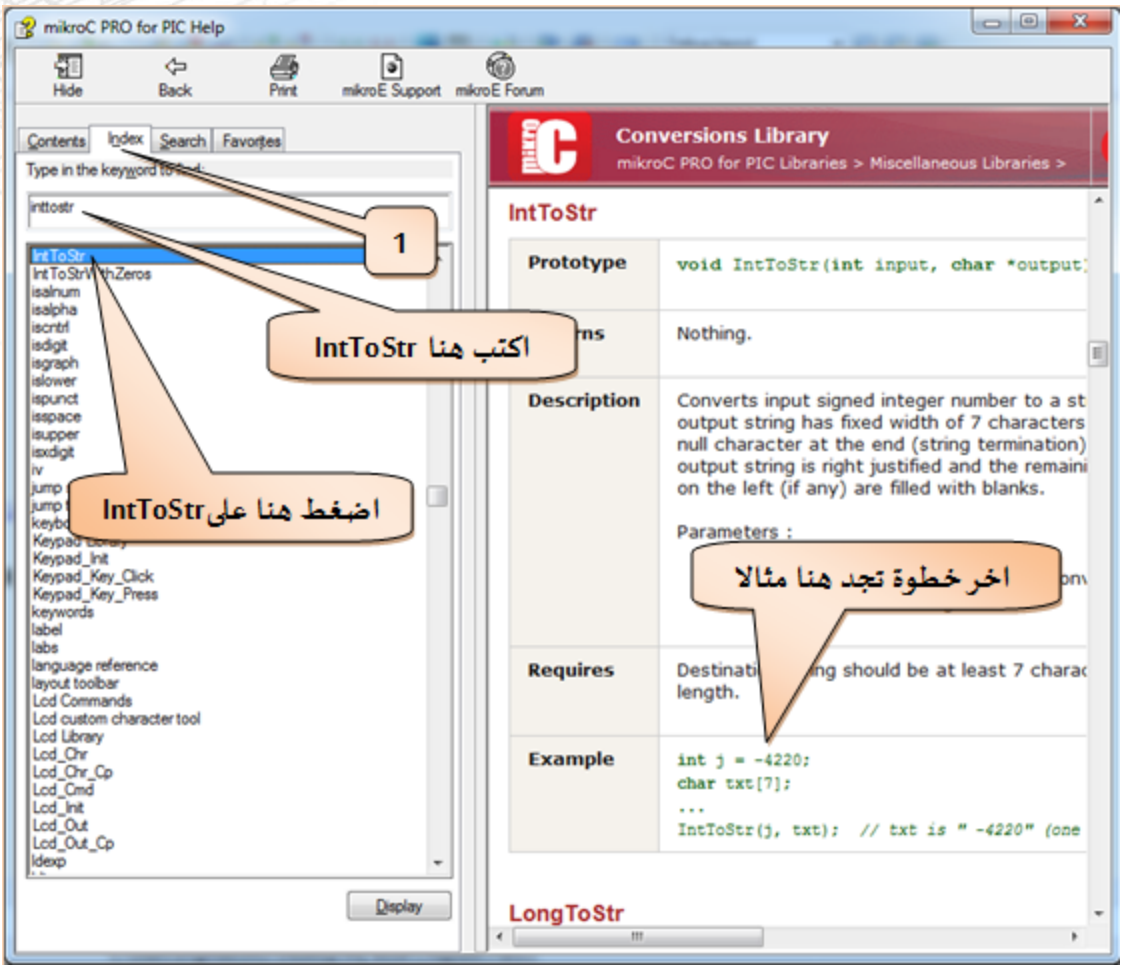
في المشروع التالي نريد أن نقوم بعرض القيم من واحد إلى عشرة على الشاشة ... أول ما قد يتبادر إلى الذهن هو استخدام الحلقة التكرارية for وبالتالي سنحتاج إلى متغير صحيح نقوم بزيادته في كل تكرار ونقوم بإرساله إلى الشاشة باستخدام الطريقة السابقة، ولكن تظهر مشكلة بسيطة هنا وهي أن الدوال السابقة الخاصة بالشاشة لا تتعامل إلا مع السلاسل الحرفية أما المتغيرات الصحيحة فلا تتعامل معها، وبالتالي إذا قمنا بتعريف متغير صحيح وليكن

```
int  x = 5;
```

ثم قمنا بكتابة الدالة الآتية:

```
Lcd_Out(1,1, x );
```

فانه لن يطبع القيمة خمسة، وعليكم تجربة ذلك، ولحل هذه المشكلة لكي نتمكن من طباعة القيمة خمسة الموجودة في المتغير x لابد من تحويله من شكل المتغير الصحيح إلى شكل السلسة الحرفية (ولن تتغير قيمته فنحن نريد ان نقوم بتخزينه بصيغة السلسة الحرفية ولكن الاختلاف أننا لن نتمكن من عمل العمليات الحسابية عليه) والذي يفعل ذلك هو الدالة IntToStr ويمكنك الذهاب لنافذة المساعدة لمعرفة كيفية التعامل مع هذه الدالة كالتالي:



من هذه النافذة مذكور أنه يجب أن يكون المتغير السلسلة الحرفية الذي سنحول فيه المتغير لا يقل طوله عن ٧ حروف، وهذا مذكور في نافذة المساعدة في هذه المنطقة:

Requires	Destination string should be at least 7 characters in length.

وبالتالي لابد من يتم تعريف متغير كالآتي:

```
Char str[7];
```

يتبقى الخطوات التي نريد بها تنفيذ المشروع كالآتي:

- **أولا**: سنستخدم الحلقة التكرارية for لأننا نريد إظهار الأرقام من واحد إلى ١٠ ولا نريد أن نكتب نفس الأوامر أكثر من مرة.
- **ثانيا**: داخل الحلقة for نقوم أولا بمسح الشاشة، وهذا أمر هام لابد منه لأن الشاشة سيتكتب عليها في كل مرة من مرات التكرار، وبالتالي قبل الكتابة لابد من مسحها أولا.

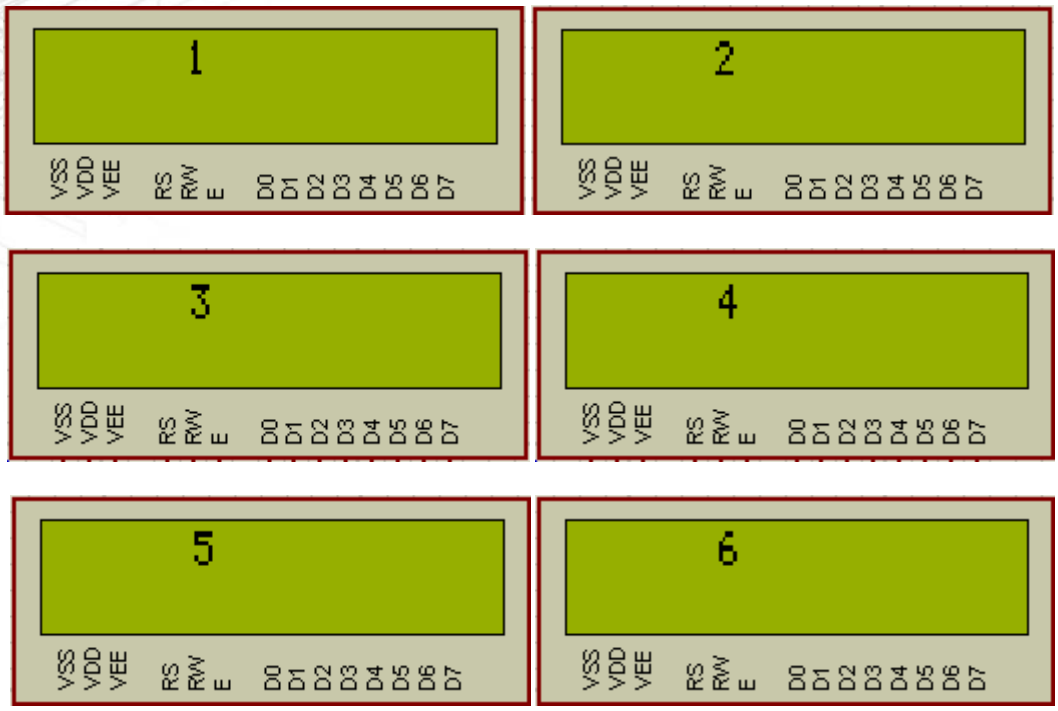
- **ثالثا:** نقوم بتحويل الرقم i الذي تستخدمه حلقة for إلى صيغة السلسة الحرفية باستخدام الدالة التي أشرنا إليها سابقا.
- **رابعا:** نقوم بإظهار الرقم.
- **خامسا:** نقوم بعمل تأخير لمدة واحد ثانية حتى نستطيع رؤية الأرقام.
- **سادسا:** لا ننسى كتابة الـ ١٢ أمر الخاصين بتوصيل الأطراف.

وعندئذ يصبح البرنامج كالآتي:

```
int i;
char str[7];
void main()
{
  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);

    while(1)
     {
       for(i=1;i<=10;i++)
        {
           Lcd_Cmd(_LCD_CLEAR);
           IntToStr(i,str);
           Lcd_Out(1,1,str);
              delay_ms(250);
        }
     }
}
```

نلاحظ أن ما بداخل الحلقة for سيتم تنفيذه ١٠ مرات، في كل مرة سيكون فيها المتغير i بقيمة متزايدة، ففي المرة الأولى سيكون المتغير يساوي بواحد وفى المرة الثانية سيكون المتغير يساوي ٢ ... وهكذا تستمر حلقة التكرار حتى يصل إلى القيمة ١٠ ... لاحظ أيضا أن المتغير السلسلة حرفية تم تحديد حجمه بسبع حروف ...

وبالتالي ليصبح الخرج على بروتس كما في الشكل الموضح:



وهكذا إلى نهاية باقي الأرقام ...

مهارة برمجية

أريد أن أعلمكم شيء جديد آخر خاص بالبرمجة وهي كالآتي: فيما يخص الـ ١٢ أمر الخاصين بالمجموعة الأولى والثانية من المتغيرات الذين يكتبوا في بداية كل برنامج لتحديد أطراف الميكرو المتصلة بأطراف الشاشة ...

لتجنب تكرار هذه الأوامر في بداية كل برنامج ولتجنب زيادة حجم البرنامج لا نريد أن نكتبها في كل مشروع سنستخدم الشاشة فيه ... لتنفيذ ذلك نقوم بحفظهم جميعا ككتلة واحدة في ملف واحد نعطى لهم اسما و نحفظه في مكان ما وعندما نريد أن ننفذ مشروع جديد تستخدم فيه الشاشة نكتب فقط اسم هذا الملف و نخبر المشروع أن يستعمله من المكان المخزن فيه وهذا يكتب في سطر واحد بسيط فقط بدلا من الـ ١٢ امر الكاملين ... لنرى كيف ذلك؟

أولا: نقوم بفتح برنامج المحرر المدمج في الويندوز Notepad من قائمة البداية Start:

Start >> All programs >> Accessories >> Notepad

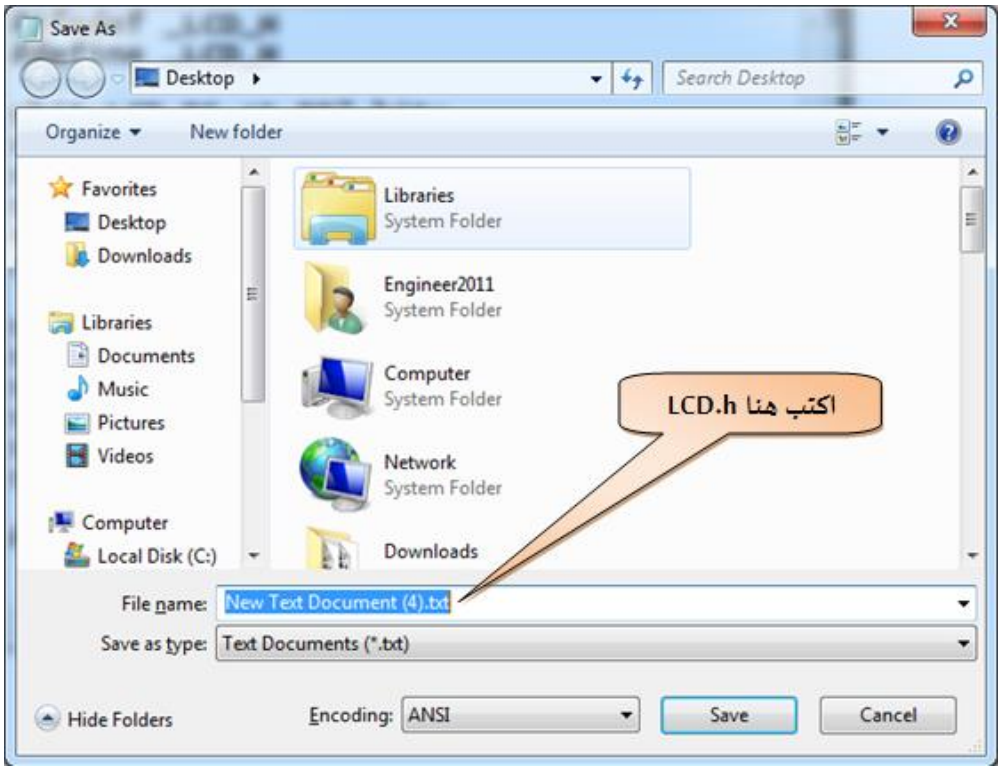
نكتب الأوامر كما بالشكل التالي:

LCD.h - Notepad

File Edit Format View Help

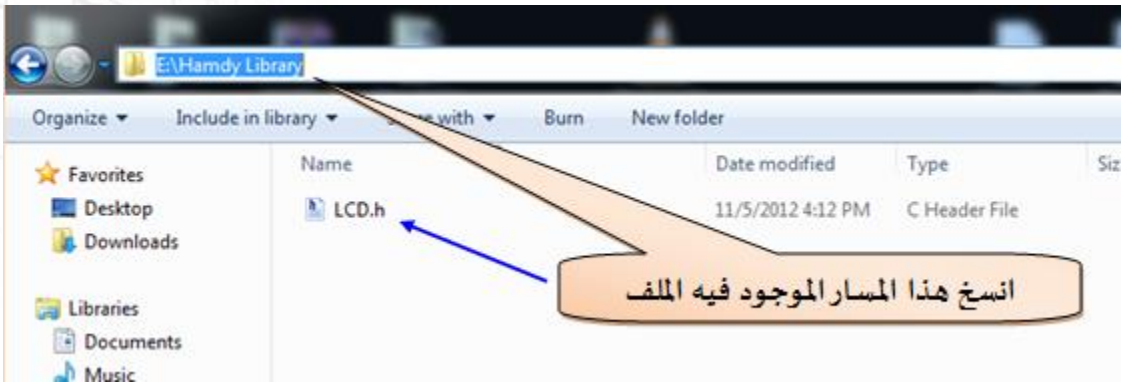
```
#ifndef _LCD_H
#define _LCD_H

sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

#endif
```

ثم من قائمة file اختر Save as فتظهر لك نافذة اكتب فيها اسم الملف كما هو موضح في الصورة:

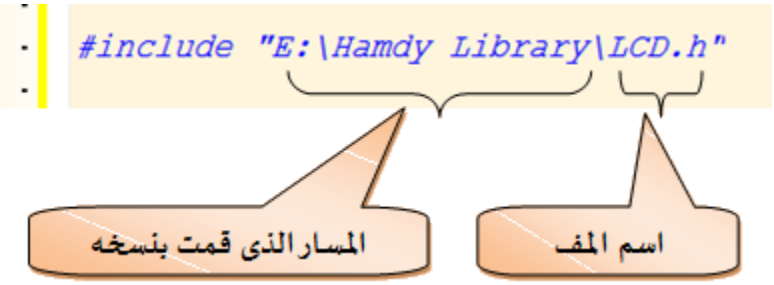


واحفظ هذا الملف في فولدر باسمك داخل أي مكان في الكمبيوتر الخاص بك.

وبهذا انتهينا من كتابة الملف بقي أن نضمنه داخل البرنامج ... دعنا ببساطة نذهب إلى المكان المحفوظ فيه الملف وتنسخ المسار منه كما بالشكل:



ثم نكتب أمر التضمين كالتالي:



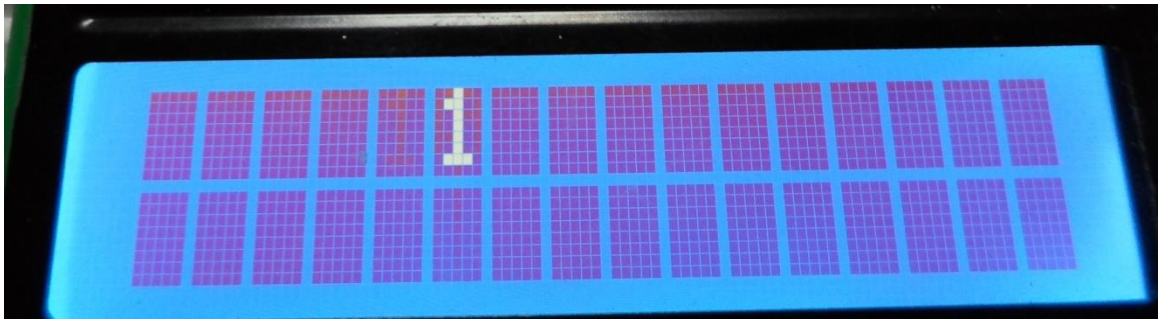
وعندئذ يصبح شكل البرنامج كالتالي بالكامل:

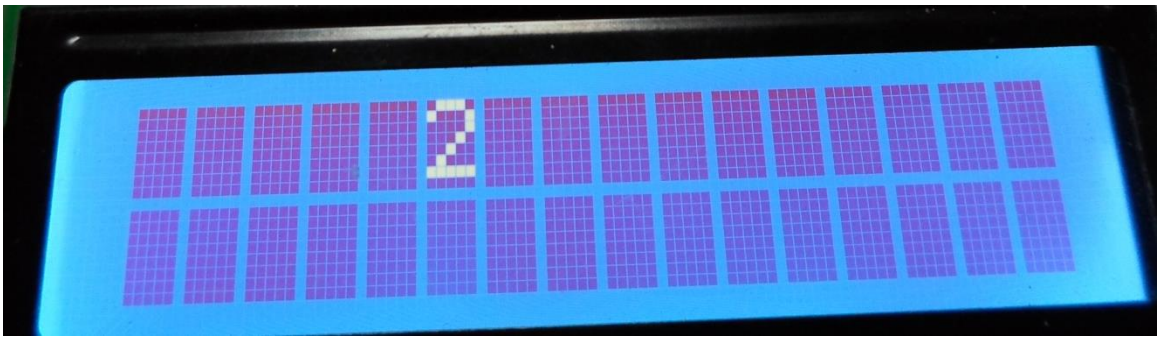
```
#include "E:\Hamdy Library\LCD.h"

int i;
char str[7];
void main()
{
  Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);

    while(1)
     {
       for(i=1;i<=10;i++)
        {
```

```
            Lcd_Cmd(_LCD_CLEAR);
            IntToStr(i,str);
            Lcd_Out(1,1,str);
               delay_ms(250);
         }
      }
}
```

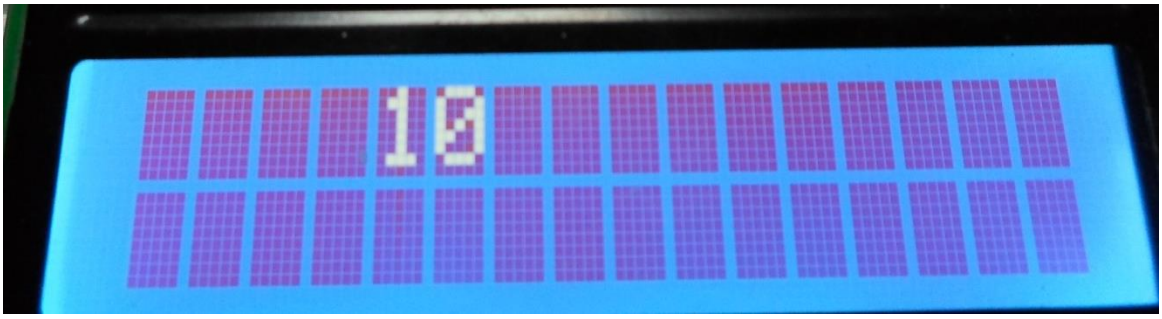
وبالتالي لا نقوم بكتابة ال ١٢ أمر هنا ثانية !!! ..... أمر رااائع ... ولكنى لم أقم بشرحه بالتفصيل ولعلى أفعل في نسخة أخرى من الكتاب إن شاء الله.

الهاردوير





...



What is Fab Lab Egypt ?

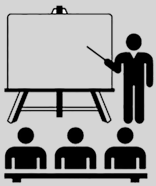
Learn . Make . Share !

Fab Lab Egypt (a member of Massachusetts Institute of Technology Fab Lab global network) is a non-profit, non-governmental, community-run public makerspace and digital fabrication lab.

Co-working Space

Machines & Tools

MIT Curriculum



Workshops & Events

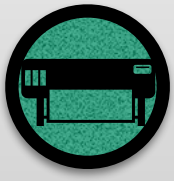
Machines we have

Laser Cutter

Zooba CNC Router

Modela

3D Printer



Vinyl Cutter

Address: 10 Abdulrahman El-Rafei (infront of Shooting club gate #5) St., from Makkah St., Dokki Giza, Egyp

Email: info@fablab-egypt.com
Website: http://www.fablab-egypt.org
Phone no.: +2 0111 160 7406

Fab Lab Egypt

Learn . Make . Share !

الفصل الثامن

التعامل مع لوحة المفاتيح Keypad

لا شك أنه من أهم العمليات التي نستخدمها قبل المعالجة هي عملية إدخال البيانات، وكما في الكمبيوتر يتم استخدام الكيبورد أو الـ Scanner فمع الميكروكنترولر يتم استخدام الكيباد كوسيلة لإدخال البيانات للميكرو

عن لوحة المفاتيح

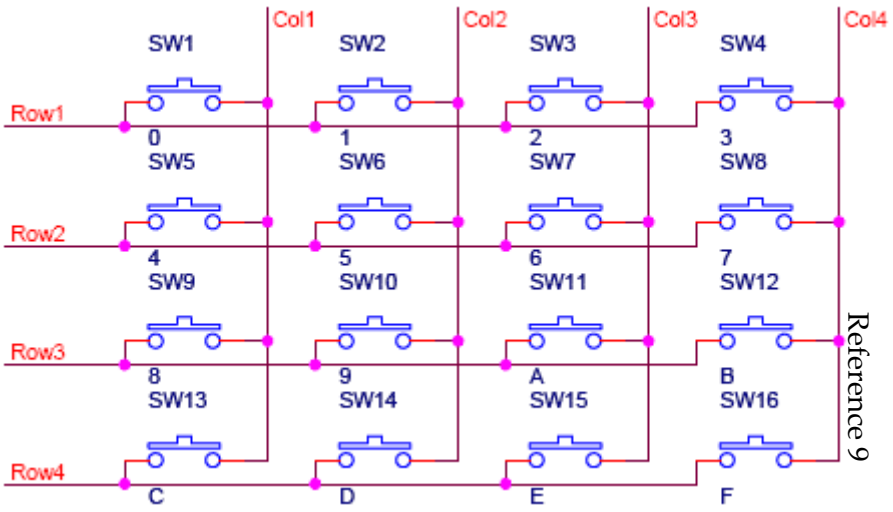
تستخدم لوحة المفاتيح Keypad في إدخال البيانات – أرقام وحروف – إلى المعالج ليقوم بعمل مجموعة من العمليات عليها ...

ومن أمثلة الاستخدامات التي يمكن أن تستخدم فيها لوحة المفاتيح بجانب الميكروكنترولر:

١) في أنظمة الأمان Security System حيث تستخدم لإدخال الرقم السري أو كلمة المرور Password.

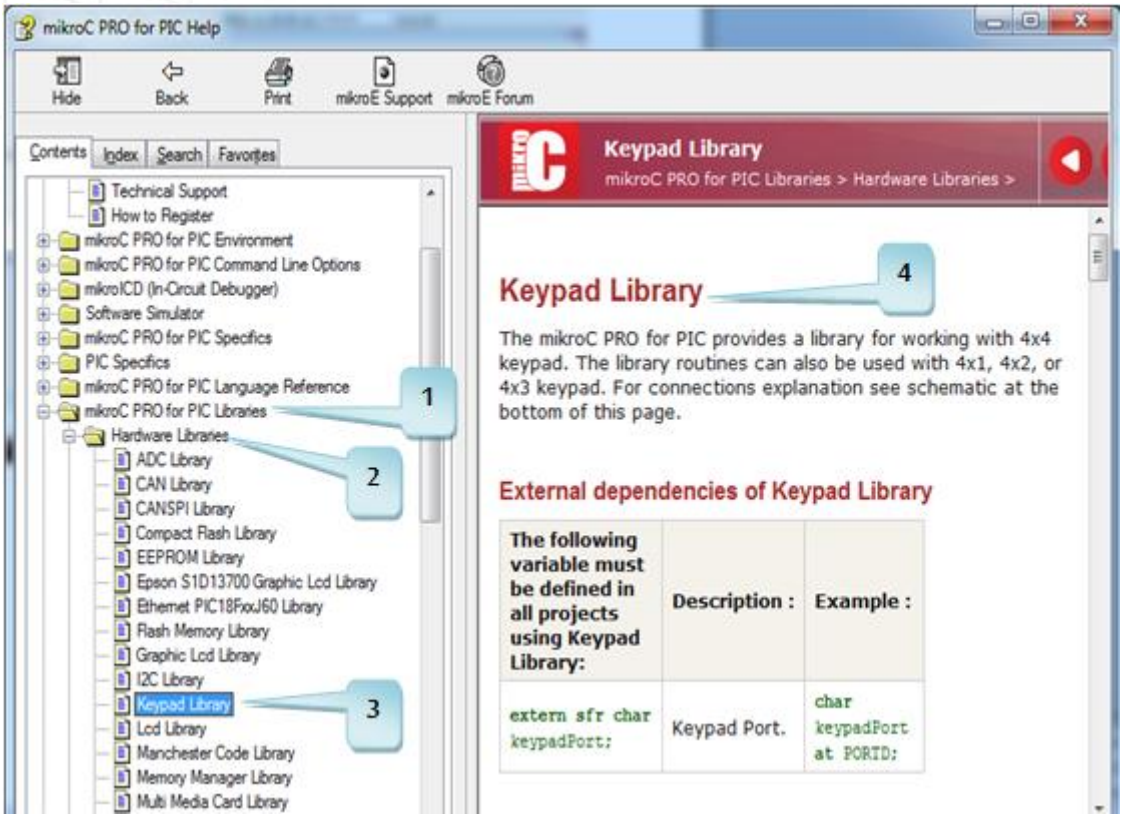
٢) تستخدم مثلا لإدخال درجة الحرارة التي نريد للميكرو أن يحفظ درجة المكان عندها ... وغير ذلك من الكثير من الاستخدامات.

يوجد للكيباد أشكال عديدة إلا أن فكرة عملها واحدة – ولن أتطرق لتفصيلها – فمنها ما يستخدم في الموبايل أو مع الكمبيوتر أو غير ذلك ...



أوامر ودوال الميكرو سي

في الميكرو سي توجد دوال خاصة بالتعامل مع الكيباد، كما هو الحال في الشاشة ويمكنك الحصول عليها جميعا من نافذة المساعدة كما تعلمنا سابقا:



الأمر الأول

يستخدم لأعلام الميكرو بالبورت المتصل عليه الكيباد، كالآتي:

```
char keypadPort at PORTD;
```

تكتب هنا اسم المخرج الذى ستوصل به الكيباد

ففي الأمر السابق اخترنا المخرج D وإذا أردنا توصيل الكيباد على المخرج B فان الأمر يكون كالآتي:

```
char keypadPort at PORTB;
```

الدالة الثانية

تستخدم لتهيئة مخرج الميكرو للاستخدام مع الكيباد، وتكتب داخل الـ main بالشكل التالي:

```
Keypad_Init();
```

الدالة الثالثة

دالة أخرى تخبرنا بالزر الذي تم الضغط عليه، وهي على الشكل التالي:

```
char kp;
...
kp = Keypad_Key_Click();
```

هذا المتغير لنستقبل فيه قيمة الزر ويكتب قبل ال main

هذه هي الدالة حيث لما نضغط على الزر تتخزن قيمته في المتغير KP

توجد أيضا دالة أخرى لقراءة قيمة الزر تكتب كالآتي:

```
char kp;
...
kp = Keypad_Key_Press();
```

هذا المتغير لنستقبل فيه قيمة الزر ويكتب قبل ال main

هذه هي الدالة حيث لما نضغط على الزر تتخزن قيمته في المتغير KP

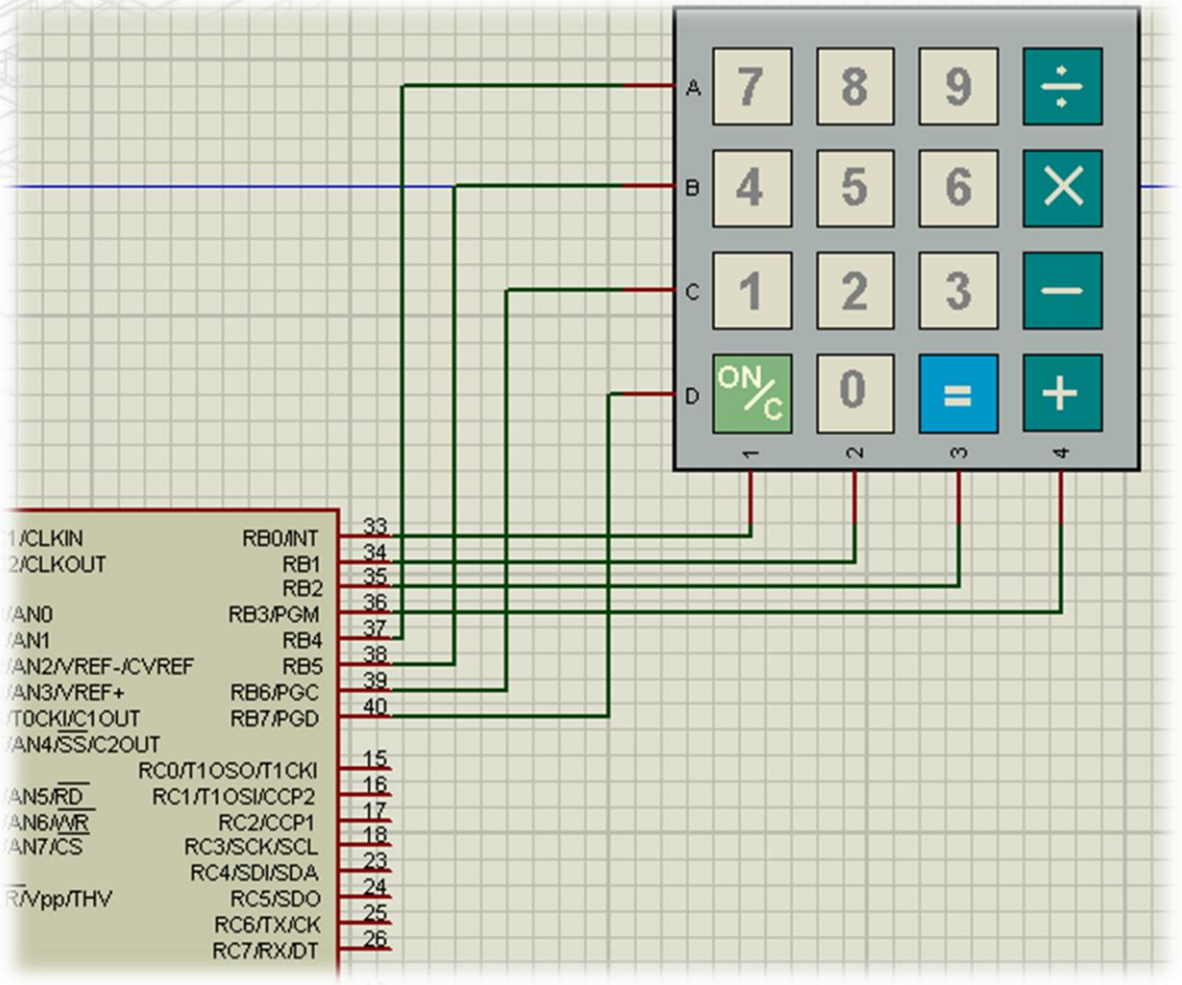
والفرق بين الدالتين هو أن الدالة الثانية لا تنتظر حتى ترفع يدك من على الزر بل بمجرد الضغط عليه ترسل القيمة للبرنامج ويكمل البرنامج تنفيذ أوامره بعد الدالة ... لكن الدالة الأولى لا تأخذ القيمة إلا بعد رفع يدك من على كل الأزرار المضغوط عليها (في حال القيام بالضغط على أكثر من زر في نفس الوقت) ثم بعد ذلك تأخذ فقط أول قيمة قمت بالضغط عليها.

وهنا يتبادر إلى الذهن سؤال هام جدا: ما هي القيم التي نحصل عليها عند الضغط على الأزرار ؟؟؟؟

والإجابة: في حالة الكيباد الـ 4×4 أي الذي يحتوي على أربع صفوف وأربع أعمده كما في الصورة الآتية في هذا الفصل يكون هناك ١٦ قيمة هم القيم من ١ إلى ١٦، وبالتالي عند الضغط على أي زر فان الميكرو سيستقبل قيمة بين ١ إلى ١٦ وإذا لم يتم الضغط على أي زر فان الميكرو يستقبل القيمة صفر.

سؤال آخر: أي الأزرار يعطى القيمة واحد وأيهما يعطى القيمة اثنين وأيها ثلاثة ... وهكذا؟؟؟

ولكن قبل الإجابة على هذا السؤال نتطرق إلى جزئية توصيل الكيباد بالميكرو أولا، وفيما يلي مثال لهذا التوصيل في بروتس، وفيه تتصل الكيباد بالميكرو على PORTB كما بالشكل:

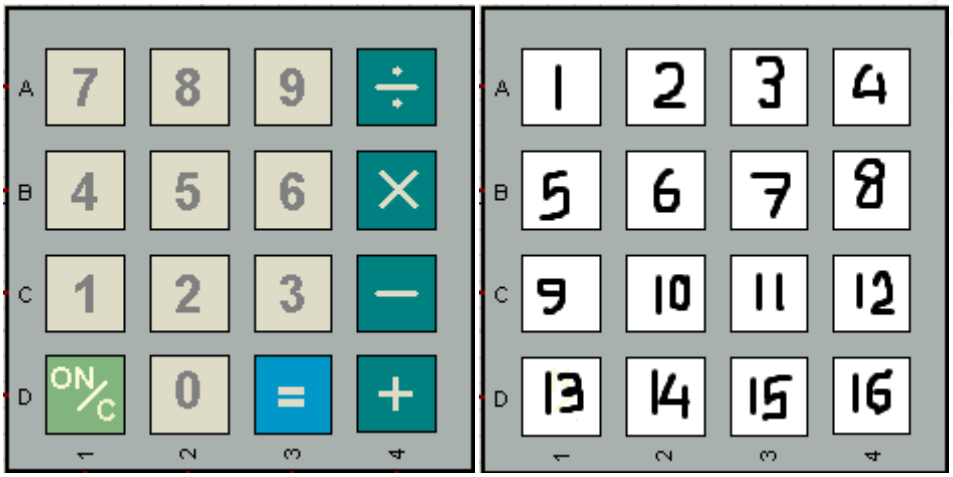


ثم نأتي لإجابة السؤال السابق: إذا قمت بتوصيل الكيباد كما هو موضح بالشكل فان الزر المكتوب عليه ٧ عندما يتم الضغط عليه فإن الميكرو يستقبل من الكيباد القيمة واحد وليس سبعة !!

عند الضغط على الزر المكتوب عليه ٨ فان الميكرو يستقبل من الكيباد القيمة اثنين وليس ٨.
عند الضغط على الزر المكتوب عليه ٩ فان الميكرو يستقبل من الكيباد القيمة ثلاثة وليس ٩.
بالضغط على الزر المكتوب عليه علامة القسمة ÷ فان الميكرو يستقبل من الكيباد القيمة أربعة وليس ÷.
وعند الضغط على الزر المكتوب عليه ٤ فان الميكرو يستقبل من الكيباد القيمة خمسة وليس ٤.
أيضا إذا ضغطت على الزر المكتوب عليه ٥ فان الميكرو يستقبل من الكيباد القيمة ستة وليس ٥.

... وهكذا باقي الأزرار بنفس الطريقة.

ويمكن تلخيص ما سبق في الرسمة الآتية ففيها نجد رسمة الكيباد ويناظرها رسمة أخرى تحتوي على القيم المرسلة من الكيباد إلى الميكرو فعند الضغط على زر من الكيباد يرسل إلى الميكرو القيمة المناظرة له في الشكل المجاور، وبالتالي القيم المرسلة للميكرو ترتيبها بهذا الشكل على اليمين بغض النظر عن مدى الاختلاف الذي قد يطرأ على أسماء وأشكال الأزرار على اليسار ...



ربما يكون هذا حل بسيط يتيح لي كتابة أي شيء على أزرار الهاردوير كرسمة الآلة الحاسبة في الصورة السابقة ويسهل لي التعامل معها في برنامج الميكرو ولكنه أيضا يورث مشكلة متمثلة في التعقيد وعدم فهم البرنامج بسهولة وقد ينتج عنها أخطاء عند البرمجة ... وفعلا هذه مشكلة لكن حلها بسيط واليكم الحل:

مبدئيا: نكتب امر قراءة الأزرار كالتالي:

```
kp = Keypad_Key_Click();
```

وبالتالي عندما نضغط في رسمة الآلة الحاسبة على الزر المكتوب عليه ٧ فان الميكرو يستقبل القيمة واحد طبقا الشكل المجاور، والحل هو كتابة الأمر الاتي بعد امر القراءة:

```
kp = Keypad_Key_Click();
if (kp == 1)    kp = '7';
```

وبالتالي سيقوم أمر الشرط if باختبار القيمة التي استقبلناها فلو كانت ١ هذا يعني أن الزر المضغوط عليه هو الزر المكتوب عليه ٧، وبالتالي يقوم بتعديل قيمة kp لتصبح سبعة لنتعامل معها فيما بعد في البرنامج.

وبالمثل إذا تم الضغط على الرز المكتوب عليه ٨ يستقبل الميكرو القيمة ٢ بدلا من ثمانية فيقوم البرنامج بالتعديل عن طريق شرط مماثل كالآتي:

```
kp = Keypad_Key_Click();
if (kp == 2)    kp = '8';
```

وهكذا بالنسبة لباقي الأزرار، فلو نظرنا للزر المكتوب عليه علامة القسمة من الممكن أن نكتب أمر الشرط له هكذا:

```
kp = Keypad_Key_Click();
if (kp == 4)    kp = '÷';
```

ولعلك إذا بحثت في نافذة المساعدة تجد مثال كامل على ذلك يمكنك الاستفادة منه ...

مشكلة أخرى

عند تنفيذ البرنامج والوصول بالتنفيذ للأمر `Keypad_Key_Click();` فإنه في حالة وجود زر مضغوط عليه ترسل قيمته للميكرو وفي حالة عدم الضغط على أي زر فإن القيمة صفر يتم إرسالها ووضعها في المتغير ثم يستمر البرنامج في التنفيذ من بعده وهو على عكس المطلوب، فالمفترض ألا يتعدى البرنامج هذا الأمر قبل الضغط على أي زر حيث أن سرعة التنفيذ أسرع من سرعتنا في الضغط على الزر ففي الغالب سيستقبل القيمة صفر قبل أن نضغط على أي زر

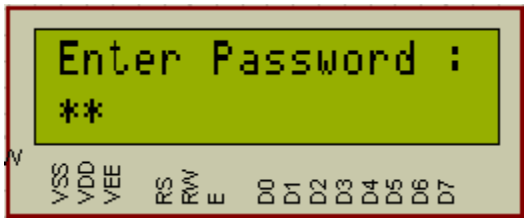
ولحل هذه المشكلة وهذا يتم من خلال وضع هذا الأمر داخل حلقة while بحيث تكون كما بالآتي:

```
while (kp == 0)
{
   kp = Keypad_Key_Click();
}
```

بشرط وضع قيمة المتغير kp في البداية بصفر، وبالتالي يقوم الميكرو باختبار شرط while لأول مرة فيجد قيمة kp تساوى صفر وهذا معناه أن الشرط محقق فيتم تنفيذ ما بداخل while وهو امر القراءة فاذا لم تكن قد ضغطت على أي زر فإن kp تبقي على قيمتها بصفر لمرة ثانية وبالتالي عند اختبار الشرط يكون محقق مرة أخرى فيتم تنفيذ ما بداخل while مرة أخرى أيضا ... وهكذا حتى تضغط على أي زر فعندها يصبح لـ kp قيمة خلاف الصفر من ١ إلى ١٦ وعند اختبار الشرط تكون قيمة kp لا تساوى صفر فلا يتحقق الشرط فلا يتم تنفيذ أوامر while وتخطاها إلى الأمر التالي لها، وهذا هو ما نريده، حيث لن يتم تخطي امر استقبال القيم من الكيباد إلا بعد الضغط على احد الأزرار وهذا مهم جدا.

مشروع تطبيقي

نريد تطبيق نفس مشروع الفلاش المذكور سابقا، لكن لا يتم البدء في الفلاش إلا عند إدخال الباسورد الصحيح وليكن ١٢٣، مطلوب في هذا المشروع أن تظهر جملة Enter Password على الشاشة عند بدء التشغيل ثم عندما يقوم المستخدم بإدخال حرف من حرف الباسورد يظهر بدلا منه علامة * على الشاشة كعادة برامج الباسورد وذلك كما هو بالشكل.



يجب أيضا أن نراعى انه في حالة إدخال باسورد خاطئ فان الميكرو ينبه المستخدم إلى ذلك عن طريق إظهار كلمة Wrong Pass على الشاشة وإعطائه الفرصة لإدخال الباسورد الصحيح مرة أخرى ...

لنبدأ في تنفيذ المشروع ...

أولا: قم بتحديد الموديولات التي ستحتاجها في المشروع (شاشات ـ كيباد ـ ADC ـ إنترنت ـ ...)، في هذا المشروع سنستخدم الشاشة والكيباد، قم أيضا بتحديد أطراف توصيل كل منهم بالميكروكنترولر، وبناء على ما تم في الفصل الماضي فقد قمنا بتوصيل الشاشة على PORTD وقمنا بعمل ملف يحتوي على ١٢ أمر خاصين بالتوصيل وهو الملف LCD.h وبالتالي فسنقوم أيضا بتوصيل الشاشة هنا على PORTD ونقوم بتوصيل الكيباد على PORTB.

ثانيا: قم بتحديد أنواع الدخل والخرج الأخرى ... وبما أنه لا توجد أي سويتشات أو سينسورات فلا توجد دخول أما الخرج فهو ليد وحيد للفلاش و نحدده على الرجل RC0 مثلا.

ثالثا: اجعل دائما أوامر توصيل الشاشة والكيباد في البداية وهذا يتحقق من خلال الأمرين الآتيين:

```
1   #include "E:\Hamdy Library\LCD.h"
·   char  keypadPort at PORTB;
```

ولو لاحظت فإن الأمر الأول مكتوب بجواره الرقم واحد وهو ما يمثل رقم السطر أي تم وضعه في البداية، وإذا أردت مراجعة هذا الأمر الأول فيرجى مراجعة الفصل الماضي الخاص بالشاشات.

رابعا: وهنا نبدأ بتعريف المتغيرات التي سنحتاجها، فما هي ؟؟ أولا الكيباد تحتاج متغير لاستقبال قيم الأزرار ... ما هو نوع هذا المتغير؟ لو رجعت لنافذة المساعدة لوجدته من النوع الحرفي ... ولكن هنا مشكلة بسيطة وهي أن الباسورد المعطى في السؤال مكون من ٣ أرقام وبالتالي لابد أيضا من استقبال

ثلاثة قيم من الكيباد وهذا يستلزم تعريف مصفوفة من ٤ قيم لأن السلسلة الحرفية عند تخزينها تحتاج لحرف زيادة ينهي السلسلة، وأيضا نحتاج إلى متغير صحيح لاستخدامه في الحلقة التكرارية for:

```
int i;
char password[4];
```

بعد ذلك نكتب الدالة الرئيسية ونكتب أول ما نكتب بداخلها الأوامر الخاصة بتحديد اتجاه الداتا (TRIS) وأوامر التهيئة Initialization:

```
void main()
{
    trisc.B0=0;    PORTC.B0 = 0;

    Lcd_Init();    keypad_Init();

    Lcd_Cmd(_LCD_CURSOR_OFF);

    get_password();

    while(1)
     {
       Lcd_Cmd(_LCD_CLEAR);
       Lcd_Out(1, 4, "Flash_Prog");
       PORTC.B0 = ~ PORTC.B0;
       delay_ms(1000);
     }
}
```

لجعل الرجل RC0 تعمل كخرج

أوامر التهيئة للموديولات المستخدمة

أمر الغاء المؤشر في الشاشة

الدالة التي تستخدم لاستقبال الباسورد واختباره ولن يتم الانتقال منها الى الفلاش الا اذا تم ادخال الباسورد الصحيح

هذا يمثل جزء الفلاش والذى من المفترض الا ينفذ الا اذا تم ادخال الباسورد الصحيح

تبقى جزء واحد فقط وهو الدالة التي تقوم باستقبال الباسورد واختباره إذا كان صحيحا أم لا ... هذه الدالة غير موجودة في الميكرو سي ولكن يتوجب عليك كتابتها وكما جرت العادة يمكنك أن تقوم بحفظها في ملف وتناديها في أي مشروع آخر بدلا من كتابتها من جديد، ولكن لابد لك من مراجعة كيفية التعامل مع الدوال الفرعية في الفصل الرابع لان هذا مهم لفهم كيف قمنا بعمل هذه الدالة ... ما الذي نريد لهذه الدالة أن تقوم بتنفيذه ؟؟؟

- **أولا**: تكتب على الشاشة Enter Pass لكي يقوم المستخدم بإدخال الباسورد.
- **ثانيا**: تقوم باستقبال الباسورد من الكيباد.

- **ثالثا:** تقوم بعملية التحويل من القيم التي يستقبلها الميكرو إلى القيم المناظرة لها على الأزرار.
- **رابعا:** تقوم باختبار الباسورد، فإذا كان صحيحا تنهى تنفيذها وترجع للدالة الرئيسية لتنفيذ الفلاش.
- **خامسا:** لو كان الباسورد غير صحيح تطلب من المستخدم إدخاله مرة أخرى:

خطوات التنفيذ:

١- اظهر على الشاشة كلمة Enter Pass.

٢- استقبل قيمة أول زر تم الضغط عليه.

٣- حول القيمة التي تم استقبالها إلى القيمة المكتوبة على الزر نفسه (فلو استقبلت واحد مثلا حوله إلى سبعة وهكذا).

٤- قم بعرض علامة * على الشاشة كإشارة لإتمام عملية إدخال أول رقم.

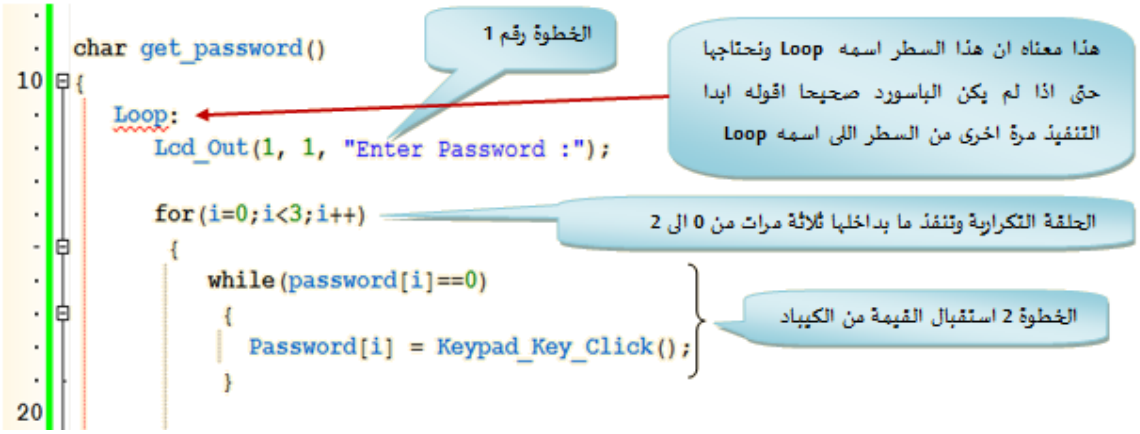
٥- نفذ الخطوات ٢ و٣ و٤ مع القيمة الثانية والثالثة للباسورد ... وهكذا تكون قد انتهيت من استقبال كامل الباسورد.

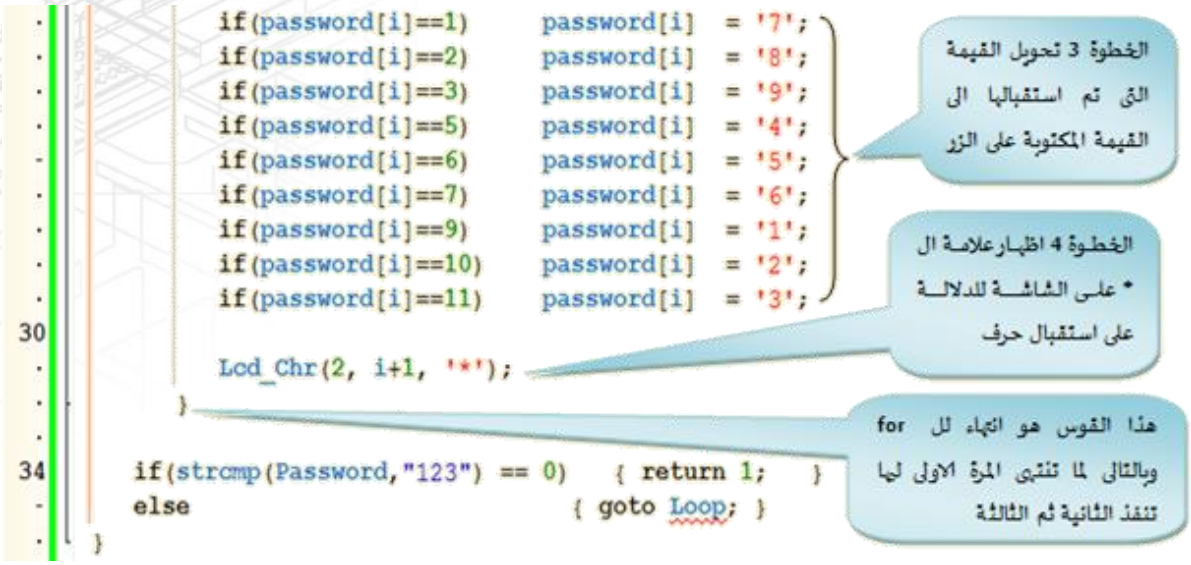
٦- قم باختبار الباسورد باستخدام شرط if.

٧- لو كان الباسورد صحيحا أرجع القيمة واحد إلى الدالة الرئيسة دلالة على مطابقة الباسورد وهنا ينتهي التنفيذ ..

٨- إذا لم يكن الشرط صحيحا نفذ الخطوات من البداية مرة أخرى (أي استقبل الباسورد مرة أخرى).

واليكم الدالة ...





ملحوظة: السطر البرمجي الآتي:

```
if (strcmp(Password, "123") == 0)  {return 1;}
```

يستخدم لاختبار الباسورد هل يساوى الباسورد الصحيح ١٢٣ أم لا، فإن الدالة strcmp هي اختصار للاسم String Compare والتي تستخدم لمقارنة سلسلتين حرفيتين فإذا كانتا متساويتين ترجع القيمة صفر وإلا ترجع قيمة مغايرة للصفر، وبالتالي عندما تكون الباسورد صحيحة أي مساوية للسلسلة الثانية ١٢٣ فإن الدالة سترجع القيمة صفر وبالتالي سيتحقق شرط التساوي وعندها يتم تنفيذ ما بداخل الشرط أي إرجاع القيمة واحد إلى الدالة الرئيسية وإنهاء تنفيذ الدالة get_password ... وبالتالي تقوم الدالة الرئيسية بإكمال التنفيذ أي تقوم بتنفيذ برنامج الفلاش.

ولكن إذا لك تكن الباسورد صحيحة يتم تنفيذ الأمر التالي وهو:

```
else{  goto Loop;  }
```

والذي معناه أكمل التنفيذ بالرجوع مرة أخرى للسطر الذي يحمل العنوان Loop وهذا يمثل الخطوة الاخيرة من خطوات التنفيذ.

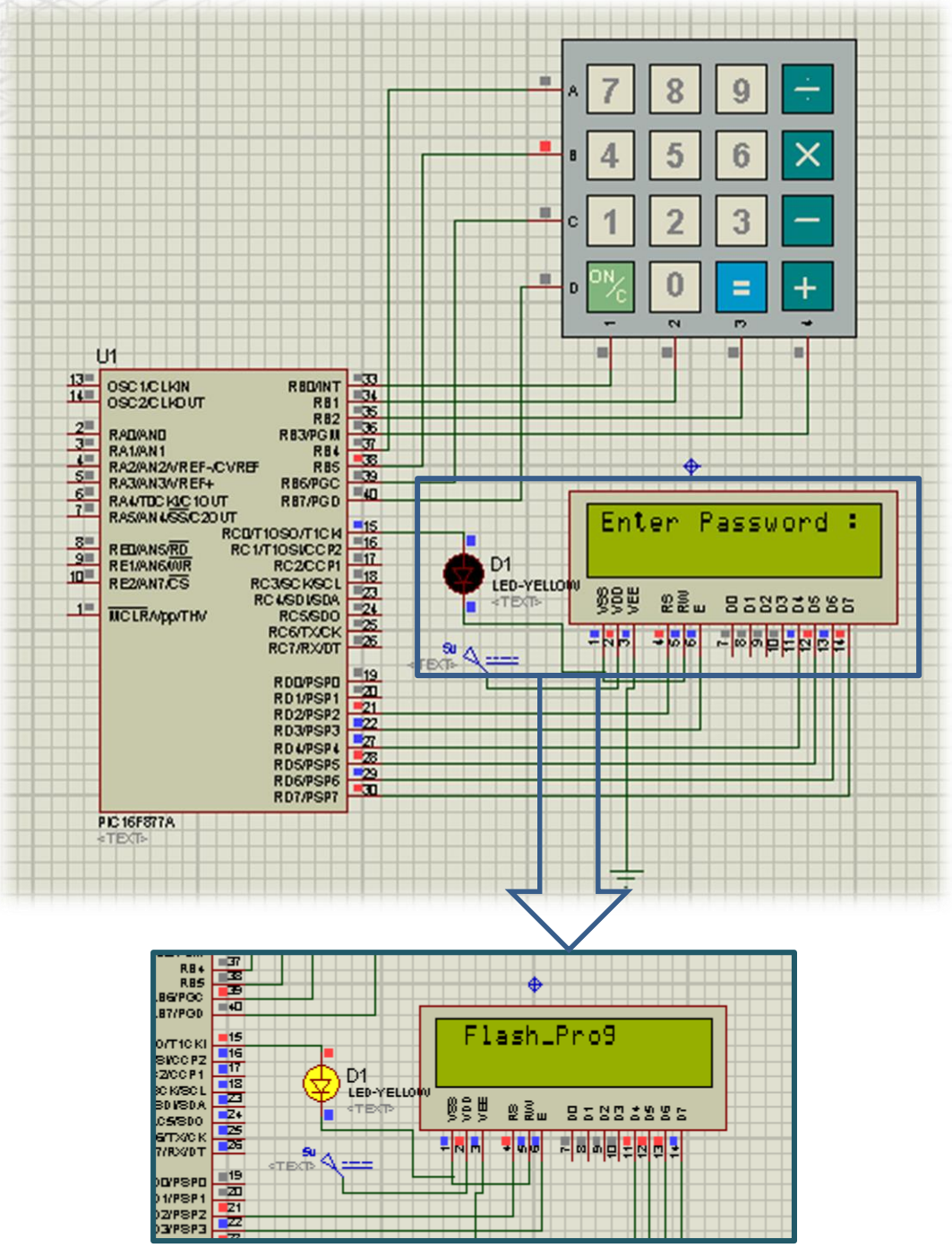
وفيما يلي كامل الكود كامل ولكن بعد حذف جزء لن يهمنا في هذا المشروع وهو الجزء الذي يحتوي على الـ if الخاصة بعملية التحويل، لكن لابد لك أن تكتبهم في باقي المشاريع، وعموما ستجد في الأسطوانات المدرجة مع الكتاب هذه الأكواد ودوائر بروتس الخاصة بها:

```
#include "E:\Hamdy Library\LCD.h"
char  keypadPort at PORTB;
int i;
char password[4];
char get_password()
{  Loop:
    Lcd_Out(1, 1, "Enter Password :");

    for(i=0;i<3;i++)
     {
       while(password[i]==0) { Password[i] = Keypad_Key_Click(); }
       if(password[i]==9)     password[i]  = '1';
       if(password[i]==10)    password[i]  = '2';
       if(password[i]==11)    password[i]  = '3';
       Lcd_Chr(2, i+1, '*');
     }
  if(strcmp(Password,"123") == 0){ return 1; } else  { goto Loop; }
}
void main()
{   trisc.B0=0;   PORTC.B0 = 0;
    Lcd_Init();   keypad_Init();
    Lcd_Cmd(_LCD_CURSOR_OFF);
    get_password();
    while(1)
     {
       Lcd_Cmd(_LCD_CLEAR);
       Lcd_Out(1, 4, "Flash_Prog");
       PORTC.B0 = ~ PORTC.B0;
       delay_ms(1000);
     }
}
```

ثم أخيرا قم بعمل build للبرنامج للحصول على ملف الهكسا في مجلد البرنامج والذي سيستخدم في المحاكاة بعد قليل.

المحاكاة

ننتقل لخطوة المحاكاة على برنامج بروتس، نقوم بعمل مشروع جديد وندرج فيه الكيباد كما تعلمنا سابقا، ونراعي طريقة التوصيل وأطراف الميكرو المذكورة في صورة سابقة في هذا الفصل، أيضا ندرج شاشة ونراعي توصيلها بنفس الطريقة المذكورة في نهاية فصل الشاشات والتي على أساسها كتب الملف LCD.h المتضمن في بداية البرنامج، وإن كنت قد قمت بتعديل أي أمر فيه أثناء تجربتك للعمل على الشاشات فلابد أن يعود لأصله وإلا تقوم بتعديل مخطط الدائرة في بروتس عن هذا المذكور في الصورة التالية:



طريقة مختصرة

يمكن اختصار البرنامج بكتابة الدالة الفرعية في ملف خارجي واستدعائها في البرنامج بأمر واحد بنفس الطريقة المستخدمة في نهاية فصل الشاشات كالآتي:

١) افتح برنامج Notepad بنفس الطريقة المذكورة سابقا واكتب فيه الدالة الفرعية

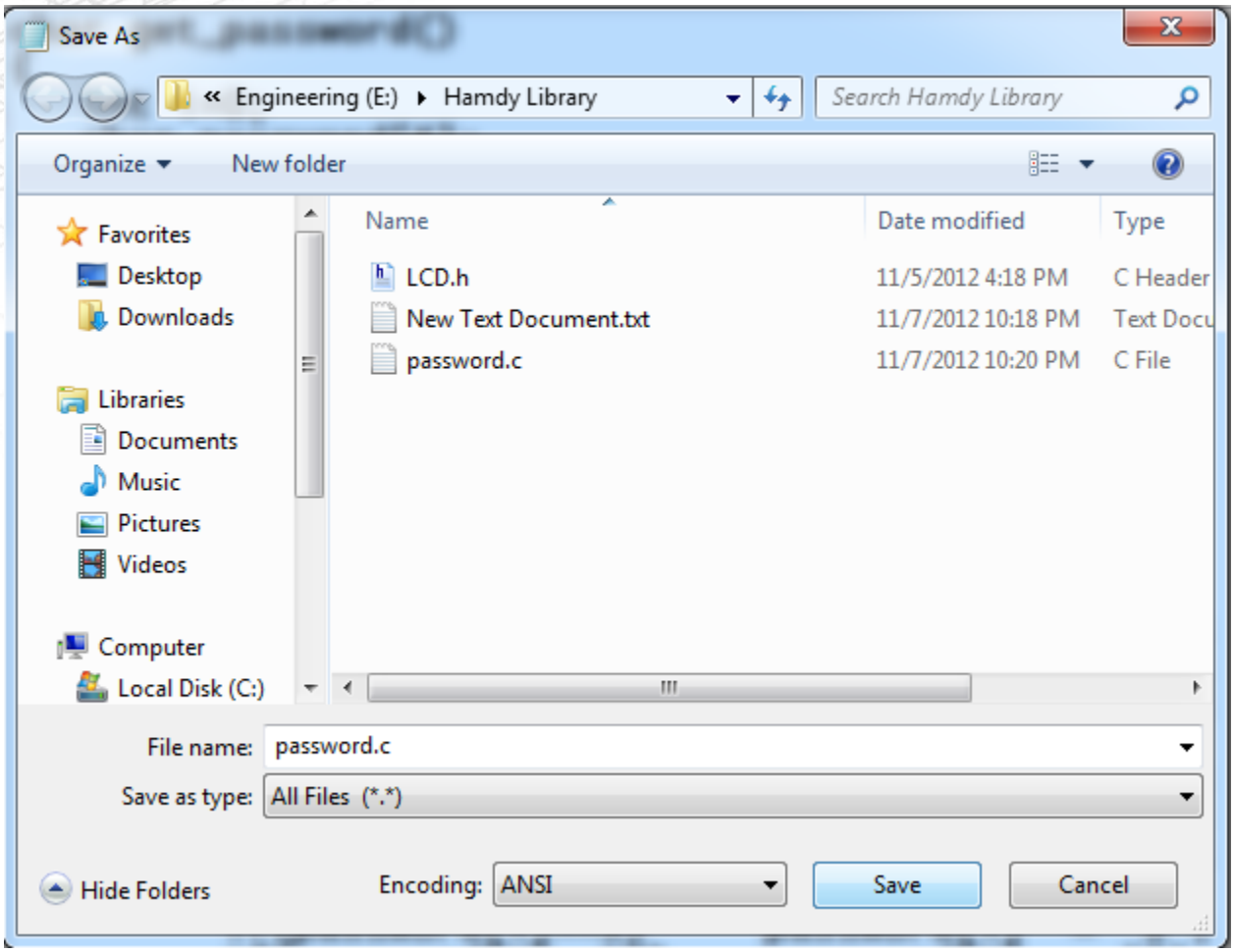
password.c - Notepad

File Edit Format View Help

```
char get_password()
{
   int i=0;
   char password[4];

   Loop1:
   Lcd_Cmd(_LCD_CLEAR);
   Lcd_Out(1, 1, "Enter passWord :");

    for(i=0;i<3;i++)
     {
        while (password[i]==0)
         {
           password[i] = Keypad_Key_Click();
         }

         if(password[i]==1)     password[i]  = '7';
         if(password[i]==2)     password[i]  = '4';
         if(password[i]==3)     password[i]  = '1';

         if(password[i]==5)     password[i]  = '8';
         if(password[i]==6)     password[i]  = '5';
         if(password[i]==7)     password[i]  = '2';

         if(password[i]==9)     password[i]  = '1';
         if(password[i]==10)    password[i]  = '2';
         if(password[i]==11)    password[i]  = '3';

         Lcd_Chr(2, i+1, '*');
      }

    if(strcmp(password,"123")==0)    { return 1;   }
   else { goto Loop1; }
}
```

لاحظ أن المتغير i والمتغير password تم تعريفهم هنا داخل هذه الدالة أي يمكن استخدامهم داخلها ولا يمكن استخدامهم خارجها ولا يمكن أن يتم تعريفهم في الدالة الرئيسية لأنهما سيستخدمان فيها فقط ولن يكونا متاحين داخل الدالة الفرعية.

من قائمة File اختر Save As (أو اضغط اختصارا على Ctrl+S من لوحة المفاتيح) واذهب إلى المكان الذي تريد الحفظ فيه واكتب اسم الملف password.c (لاحظ الامتداد c.) مع اختيار All Files من قائمة Save as type أسفل الاسم، كما بالشكل:



والآن يمكنك عمل مشروع جديد في الميكرو سي أو تعديل المشروع الحالي بحيث بدلا من كتابة الدالة الفرعية يتم حذفها والاستعاضة عنها بذكر امتداد الملف السابق حفظه وذلك بالأمر الآتي:

```
#include "E:\Hamdy Library\LCD.h"
```

وبالطبع يمكنك استدعاء هذه الدالة واستخدامها في أي مشروع آخر عن طريق فقط كتابة الأمر السابق في بداية البرنامج.

وبالتالي يصبح كود المشروع كما بالآتي:

```
#include "E:\Hamdy Library\LCD.h"
#include"E:\Hamdy Library\password.c"
char  keypadPort at PORTB;
```

```
void main()
{
    trisc.b0=0;      portc.b0=0;
    Lcd_Init();      keypad_Init();
    Lcd_Cmd(_LCD_CURSOR_OFF);

    get_password();

   while(1)
    {
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,2 , "Flash_Prog");
        portc.b0 = ~ portc.b0;
         delay_ms(1000);
    }
}
```

والآن قم بعمل build وتشغيل هذا الكود بالدائرة السابقة في بروتس تجد أنها تعمل تماما مثل المشروع السابق ...

وبهذا نكون قد تعلمنا كيفية التعامل مع اكثر من ملف في المشروع الواحد وأيضا استدعاء هذه الملفات في أي مشروع آخر.

الفصل التاسع

التعامل مع الجهود العالية

علمنا أن الميكرو يتعامل مع جهد قيم من صفر فولت إلى خمسة فولت فقط ... وبالتالي لا يمكن توصيل الميكرو – مباشرة – بأحمال تتعامل مع جهد أكبر من خمسة فولت، وإنما يتم ذلك من خلال interface بين الحمل والميكرو وهذا ما سنتعرف عليه في هذا الفصل بإذن الله

من المعلوم أن الميكرو يخرج إما صفر أو ٥ فولت وتيار ٢٥ ميللي أمبير (في حالة البك 16F877A) وبالتالي لا يمكن توصيل ما يلي مباشرة على الميكرو:

- موتور يعمل على خمسة فولت لكنه يحتاج تيار ١٠٠ ميللي أمبير.
- موتور يعمل على جهد أكبر من خمسة فولت.
- الأحمال ذات الجهود العالية مثل ٢٢٠ فولت تيار متردد ... وغيرهم.

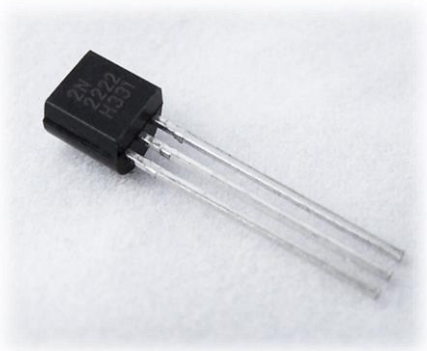
التحكم في الأحمال الثابتة

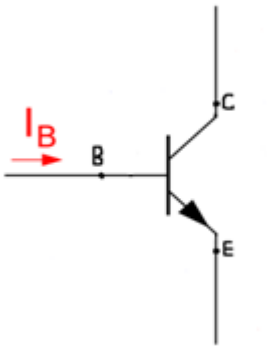
يقصد بالأحمال الثابتة DC Loads الأحمال التي تعمل على التيار أو الجهد المستمر.

نفترض أن لدينا موتور يعمل على ١٢ فولت وتيار ١٠٠ ميللي أمبير ونريد أن نتحكم فيه من خلال الميكروكنترولر، ولكن – كما تعلمنا – لا يمكننا توصيله مباشرة بالميكرو ولذلك لابد من دائرة توضع بين الموتور والميكرو تتعامل مع خرج الميكرو وتزيده ليتعامل مع الموتور.

استخدام الترانزستور

تسمى هذه الدائرة Transistor As a Switch بمعنى استخدام الترانزستور كسويتش، كيف ذلك؟

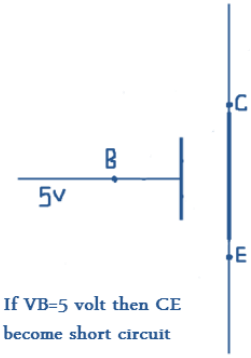




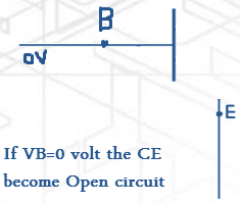
مبدئيا دعنا في هذا الكتاب نستخدم الترانزستور رقم 2N2222 أو BC377 أو غيرهم ... وفي الشكل المجاور رمز الترانزستور في الدائرة وشكله كهاردوير.

فكرة العمل

وفكرة عمل الترانزستور في حالة استخدامه كسويتش هي كالآتي: إذا ادخلنا جهد أكبر من ٠,٧ فولت على النقطة B في صورة الرمز فإن النقطة C والنقطة E يصبح بينهم short circuit أي أنهما يعتبرا متصلين، ولو أدخلنا جهد أقل من ٠,٧ فولت فإن النقطتين C، E يصبح بينهم Open circuit أي غير متصلين، وذلك لأنه في حالة توصيل الجهد الأكبر من ٠,٧ فولت يعمل

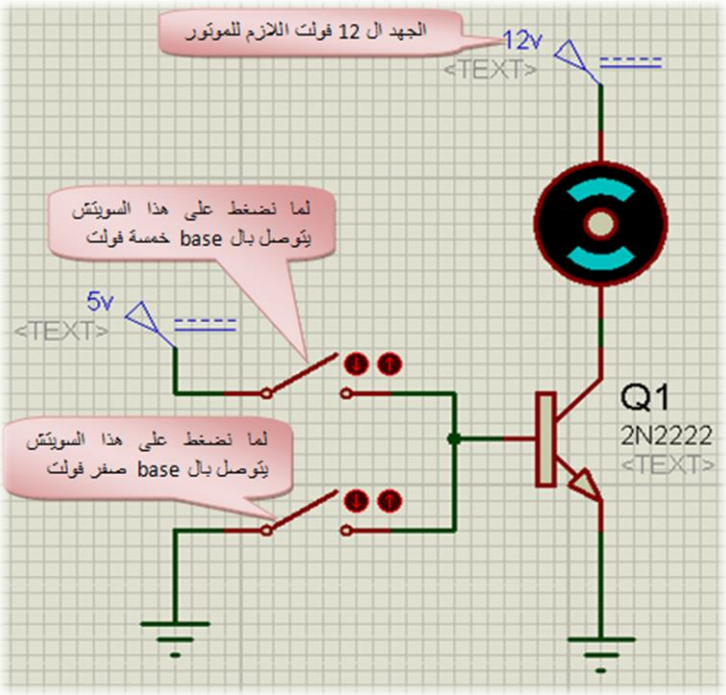


If VB=5 volt then CE become short circuit

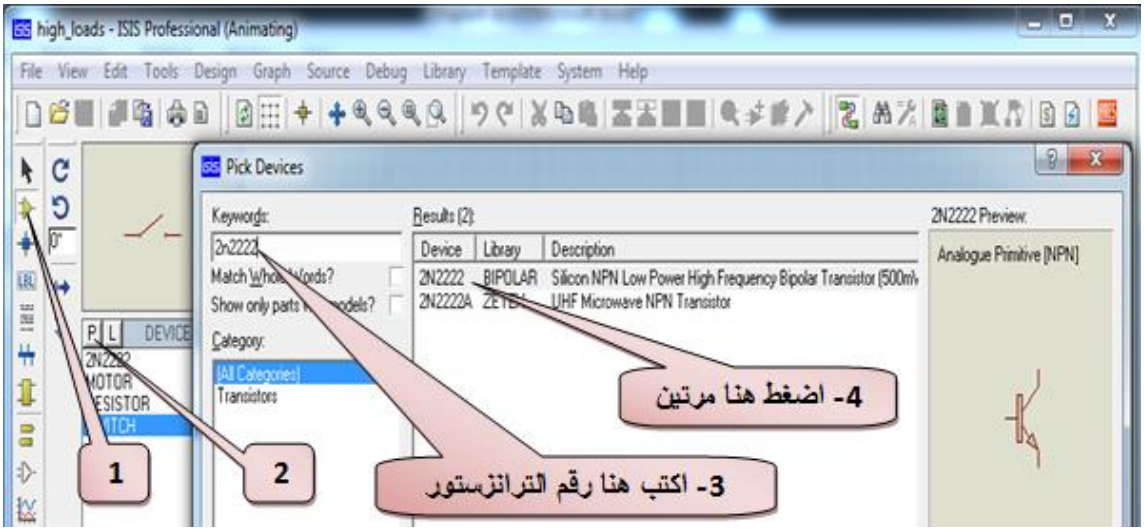


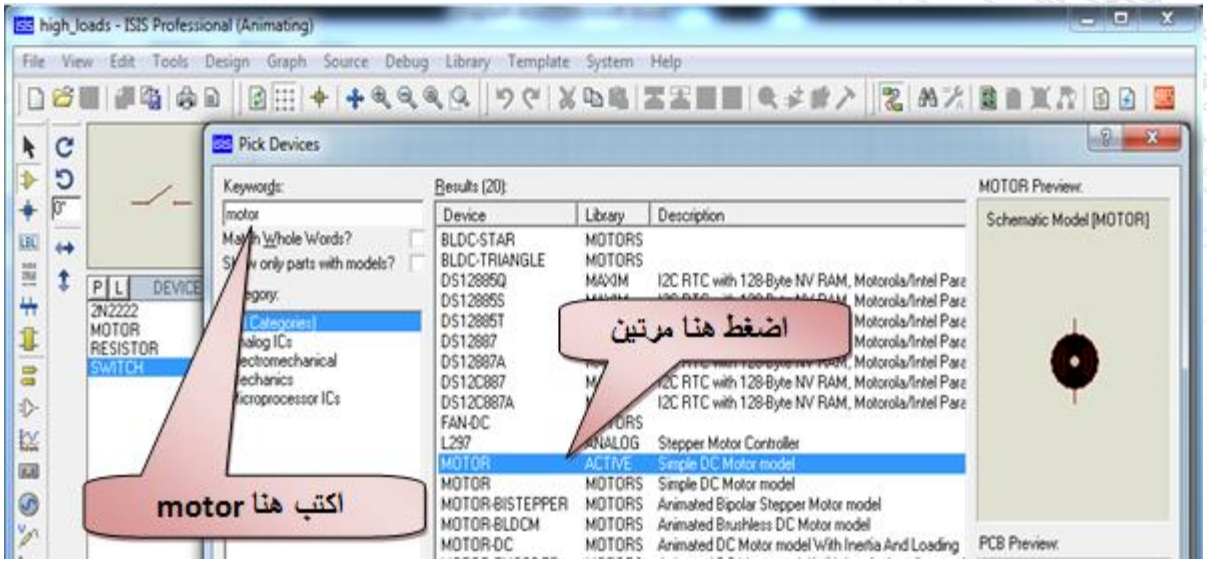
الترانزستور في حالة التشبع والتي يكون فيها الجهد VCE يساوى صفر أي متصلين كما بالشكل المجاور وعندما يكون الجهد أقل من ٠,٧ فولت يعمل الترانزستور في حالة الـ Cut Off والتي يكون فيها VCE بقيمة كبيرة تجعلنا نعتبر وجود Open Circuit بين النقطتين C، E أي غير متصلين كما بالشكل المجاور.

وبالتالي يمكن استخدام هذا الترانزستور كسويتش كما في الدائرة الأتية:

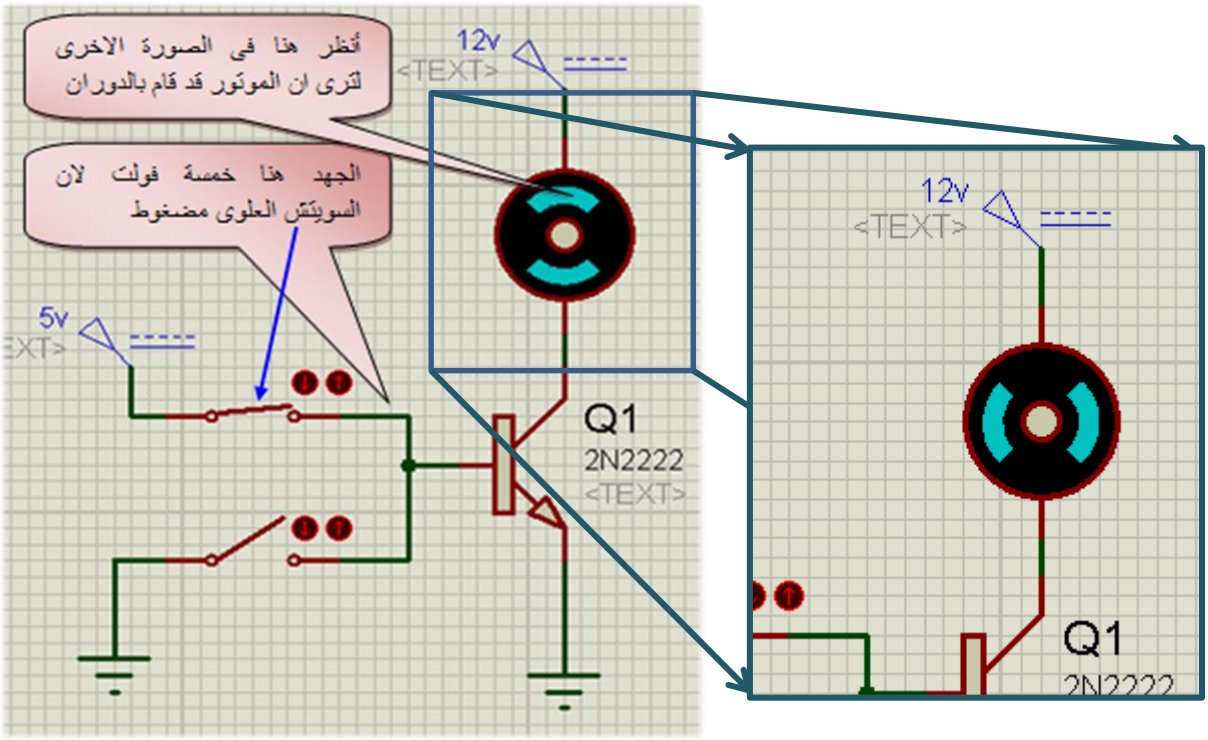


قم بعمل الدائرة السابقة على بروتس وقم بتجربة ما يلي: يمكن إضافة الترانزستور والموتور بالخطوات التالية:

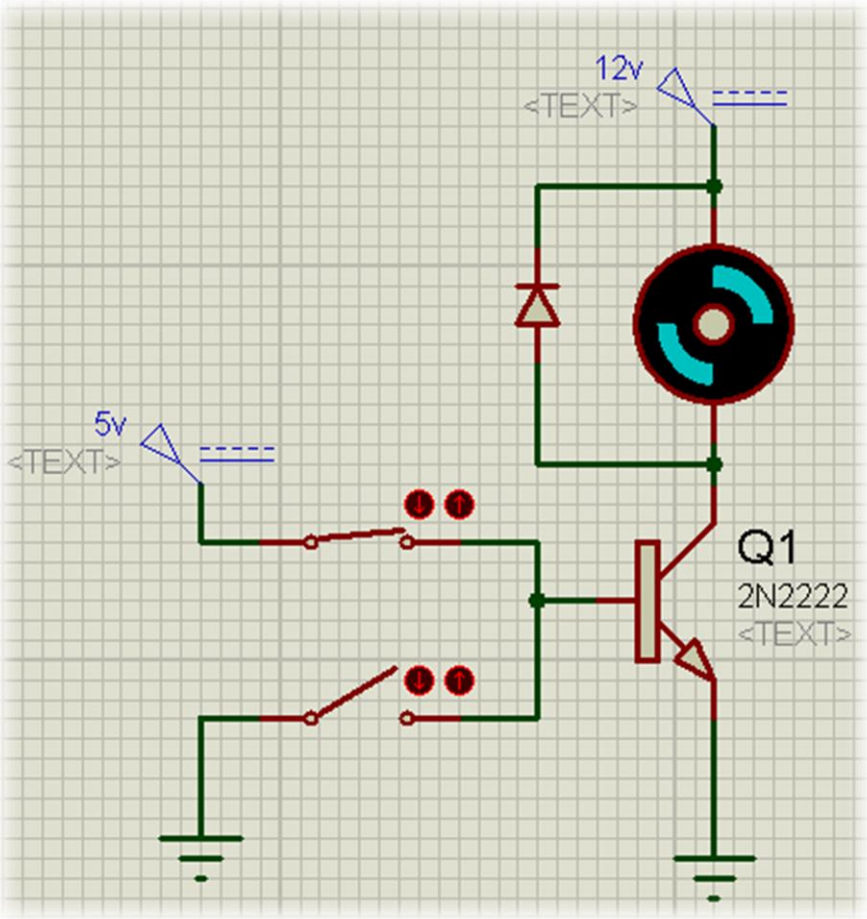




إذا أدخلت صفر فولت أي غلق المفتاح السفلي يصبح الترانزستور open circuit فلا يدور الموتور، وإذا أدخلت خمسة فولت (أكبر من ٠,٧ فولت) على الـbase أي غلق المفتاح العلوي فان الترانزستور يصبح short circuit أي يمكن وضع سلك مكانه، وهذا معناه أن الطرف الأعلى للموتور سيكون متصل بـ ١٢ فولت والطرف السفلى متصل بالأرضي (حيث أن الترانستور أصبح short circuit) فيدور الموتور وهو ما يظهر في دوران الجزء الداخلي له على المحاكاة، وهذا ما تمثله الأشكال الأتية:



ملحوظة هامة: يلزم توصيل ديود على أي Inductive loads وهو أي حمل يحتوي في مكوناته على ملفات مثل الموتور أو الريلاي أو غيرهما، وفائدة الدايود هي حماية الترانزستور والميكرو من الـ reverse current الذي ينتج عن الموتور عند توقفه ثم يسري في الترانزستور بقيمة كبيرة نسبيا فيؤدي إلى تلفه، ويوصل الدايود كما بالشكل الآتي:



ويمكنك تنزيل الدايود في بروتس بكتابة diode في مكان البحث عن المكونات فيظهر لك ثم تضيفه.

وطبقا لخصائص الترانستور وما يتحمله من تيار وجهد يمكن توصيل الأحمال المختلفة بنفس الطريقة السابقة، وهناك ما يسمى power transistor الذي يستخدم في حالة لو كان الموتور أو الحمل يحتاج تيار كبير جدا.

وبالطبع يمكنك معرفة اقصى جهد وتيار يعمل عليه الترانزستور من الداتا شيت، وبالنظر في الداتا شيت الخاصة بالترانزستور 2N2222 أو 2N2222A لوجدنا الآتي:

NPN switching transistors — 2N2222; 2N2222A

أقصى تيار يمر في الجمل

FEATURES

- High current (max. 800 mA)
- Low voltage (max. 40 V).

APPLICATIONS

- Linear amplificat

أقصى قيمة للجهد يتحملها ال VCE وهو يعتبر اقصى جهد يمكن توصيله للحمل

DESCRIPTION

NPN switching tran
PNP complement: 2N2907A.

PINNING

PIN	DESCRIPTION
1	emitter
2	base
3	collector, connected to case

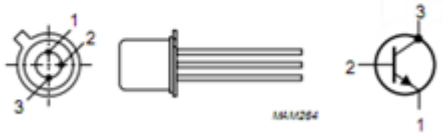


Fig.1 Simplified outline (TO-18) and symbol.

أو من خلال صفحة أخرى:

QUICK REFERENCE DATA

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	2N2222		–	60	V
	2N2222A		–	75	V
V_{CEO}	collector-emitter voltage	open base			
	2N2222		–	30	V
	2N2222A		–	40	V
I_C	collector current (DC)		–	800	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C	–	500	mW
h_{FE}	DC current gain	I_C = 10 mA; V_{CE} = 10 V	75	–	
f_T	transition frequency	I_C = 20 mA; V_{CE} = 20 V; f = 100			
	2N2222		250	–	MHz
	2N2222A		300	–	MHz
t_{off}	turn-off time	I_{Con} = 150 mA; I_{Bon} = 15 mA; I_{Boff} = –15 mA	–	250	ns

اقصى جهد يمكن توصيله للجمل

أقصى تيار يمر في الجمل

وبنفس الطريقة يمكنك الحصول على هذه القيم لأي نوع آخر مثل NC546، BC547:

NPN general pur — BC546; BC547

أقصى قيم للجهد والتيار

FEATURES

- Low current (max. 100 mA)
- Low voltage (max. 65 V).

APPLICATIONS

- General purpose switching and amplification.

DESCRIPTION

NPN transistor in a TO-92; SOT54 plastic pa ge.
PNP complements: BC556 and BC5

يمكنك من هذا الجزء معرفة ال B وال C وال E للترانزستور

PINNING

PIN	DESCRIPTION
1	emitter
2	base
3	collector

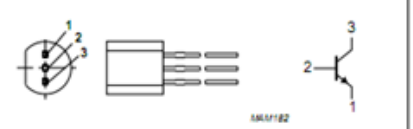
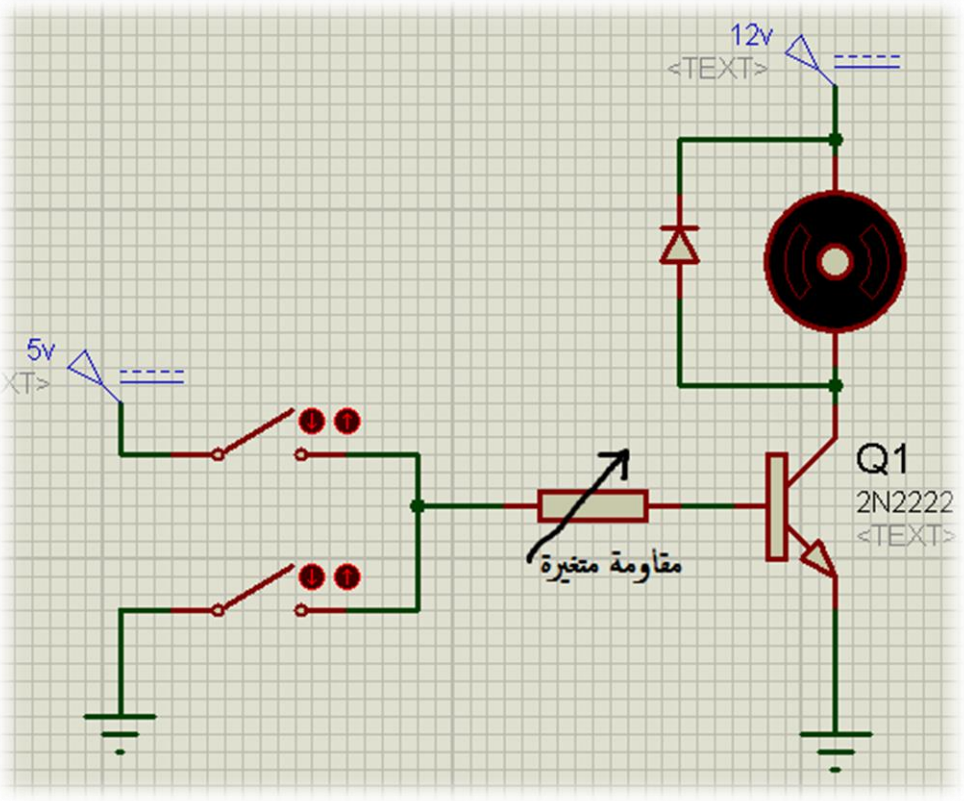


Fig.1 Simplified outline (TO-92; SOT54) and symbol.

ولكن ينبغي عليك ألا تجعل الدائرة تصل إلى هذه القيم العظمى وتصممها بحيث تكون القيم العظمي أقل من القيم العظمي للترانزستور حتى لا يتم حرقه.

ملحوظة: الترانزستورات السابقة من النوع NPN ولكن عند استخدام ترانزستور من النوع الآخر PNP نضطر إلى إدخال صفر فولت على قاعدته لجعل الترانزستور Short Circuit وليس خمسة فولت كما في النوع NPN والعكس بالعكس، أي أن العملية عكسية ...

مهارة: يمكنك توصيل مقاومة متغيرة على قاعدة الترانزستور وذلك للتحكم في تيار القاعدة والذي بدوره يتحكم في التيار بين الطرفين الآخرين أي التيار المار في الحمل، وذلك كما بالشكل التالي:

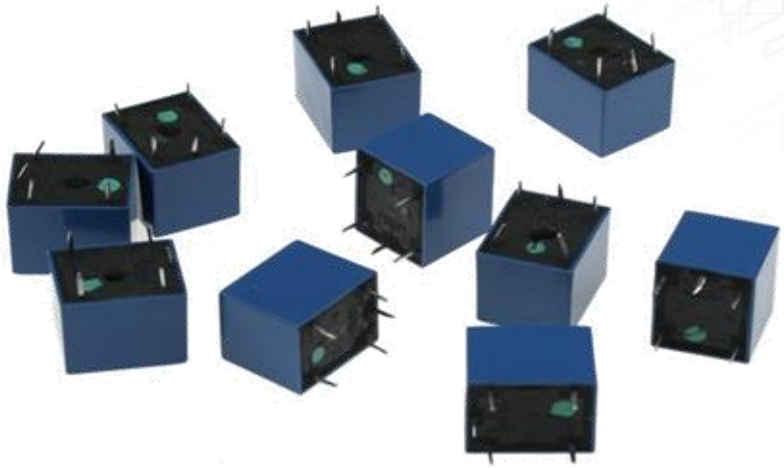


التحكم في الأحمال المترددة

يقصد بالأحمال المترددة أو المتغيرة AC Loads الأحمال التي تعمل على تيار أو جهد متردد كخرج الحائط الذي يحمل قيمة للجهد ٢٢٠ فولت.

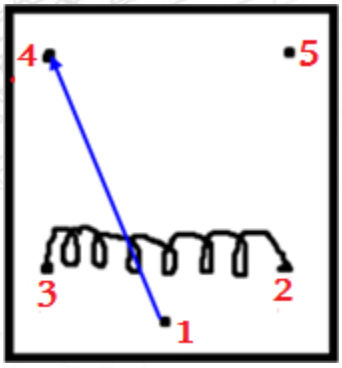
استخدام الريلاي

لإجراء هذا التحكم في هذه الأحمال يلزم بداية معرفة الريلاي وكيف يعمل هذا الريلاي ... فيما يلي شكله كهاردوير:



وبالتدقيق في الصورة نلاحظ أن الريلاي – في الغالب – يحتوي على خمسة رجول مقسمين إلى جزأين:

- الجزء الأول خاص بتوصيل الجهد الـ ٢٢٠ فولت على الحمل وهم الرجول ١ و٤ و٥.
- الجزء الثاني ويتمثل في الطرفين ٢ و٣ ويوصل عليهم الجهد الـ DC، هذا الجهد قد يكون ٥ أو ٦ أو ١٢ أو ٢٤ فولت وذلك على حسب الريلاي ونحن سنركز على الريلاي الخمسة فولت، ومواصفات الريلاي تكون مكتوبة عليه كما في الشكل الآتي:



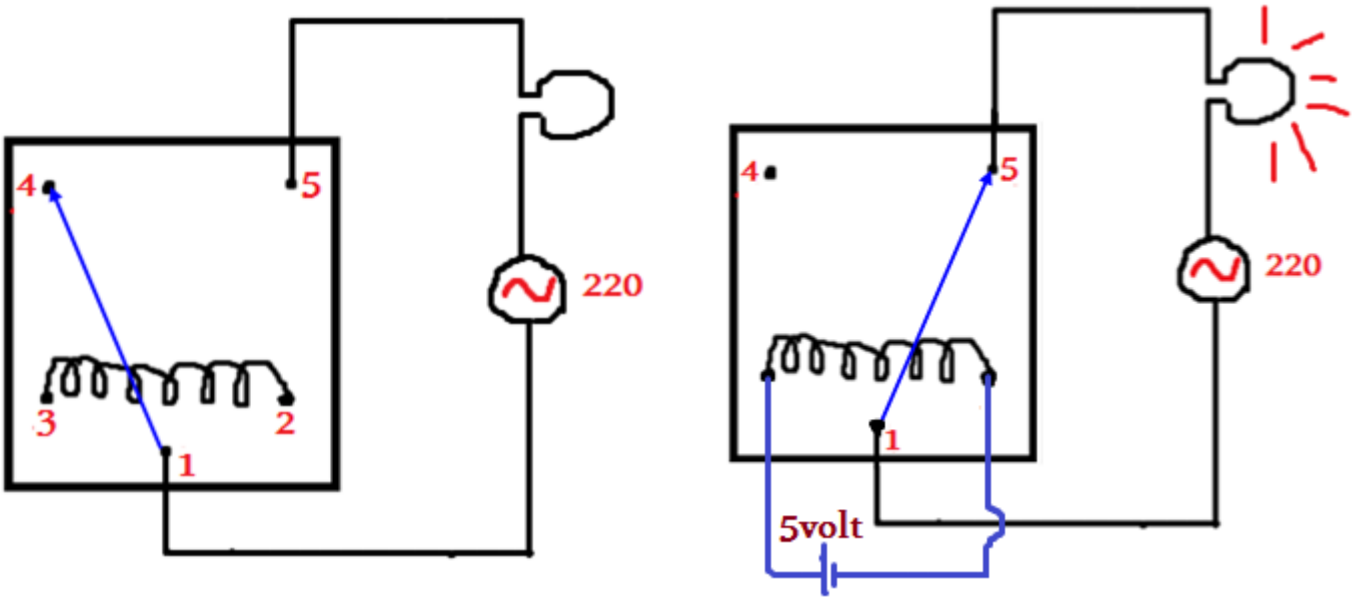
والريلاي من الداخل يظهر كما بالشكل المجاور: حيث يكون بين الطرفين ٢ و٣ ملف، والطرف واحد موصل غالبا بالطرف ٤ بطبيعة الحال (وفي بعض الأحيان بالطرف ٥).

الطرف رقم ٤ الموصل بالطرف ١ في الحالة العادية أي عندما يكون الجهد الخمسة فولت غير مطبق على الملف فهذا الطرف ٤ يسمى Normally Closed، وذلك لان الريلاي في حالته الطبيعية مغلق على هذا الطرف، كما يسمى الطرف ٥ Normally Opened وذلك أيضا لأن طرف هذا الريلاي في حالته الطبيعية يكون غير متصل على الطرف ١ أي أن الطرف ٥ هو open circuit مع الطرف ١ الذي يسمي أيضا common.

فكرة العمل

عندما نطبق الجهد الخمسة فولت على الطرفين ٢ و٣ فان الملف يولد مجال مغناطيسي هذا المجال يؤثر على الـ metal الواصلة بين ١ و٤ بقوة مغناطيسية تجعله يتحرك من النقطة ٤ إلى النقطة خمسة بحيث تكون متصلة بين الطرفين ١ و٥.

وبالتالي عند توصيل الجهد الـ ٢٢٠ ومعه الحمل كما في الشكل الآتي:

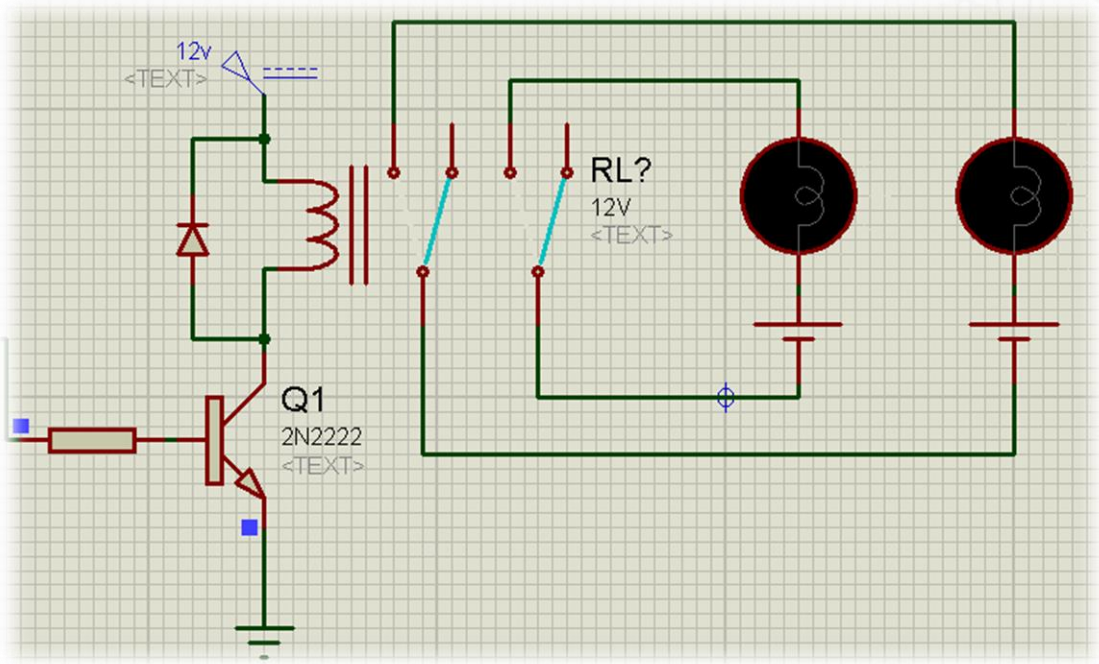


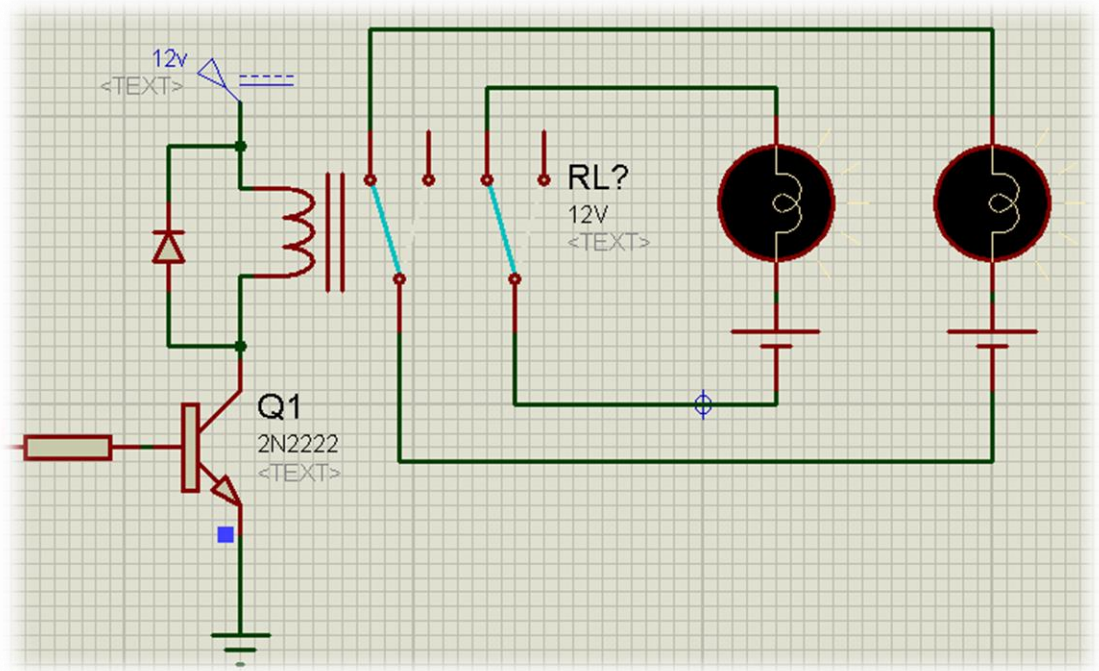
فكما ترى فإنه في الدائرة اليسرى التي لم يكن فيها جهد خمسة فولت موصول على طرفي الملف فان دائرة المصباح تكون open circuit وبالتالي لن يمر تيار وبالتالي لن يضئ المصباح، أم في الدائرة

اليمنى التي تم وضع الجهد الخمسة فولت على طرفي الملف فإن الحديدة انتقلت لتوصل بين النقطتين ١ و ٥ وبالتالي أصبحت دائرة المصباح مكتملة، وبالتالي سيمر التيار ويضئ المصباح ...

معلومة إثرائية:

الريلاي قد يحتوى على أكثر من خمس رجول وهذا النوع يستخدم غالبا للتحكم في أكثر من جهاز في نفس الوقت، وشكل الريلاي من الداخل في هذه الحالة كالآتي:

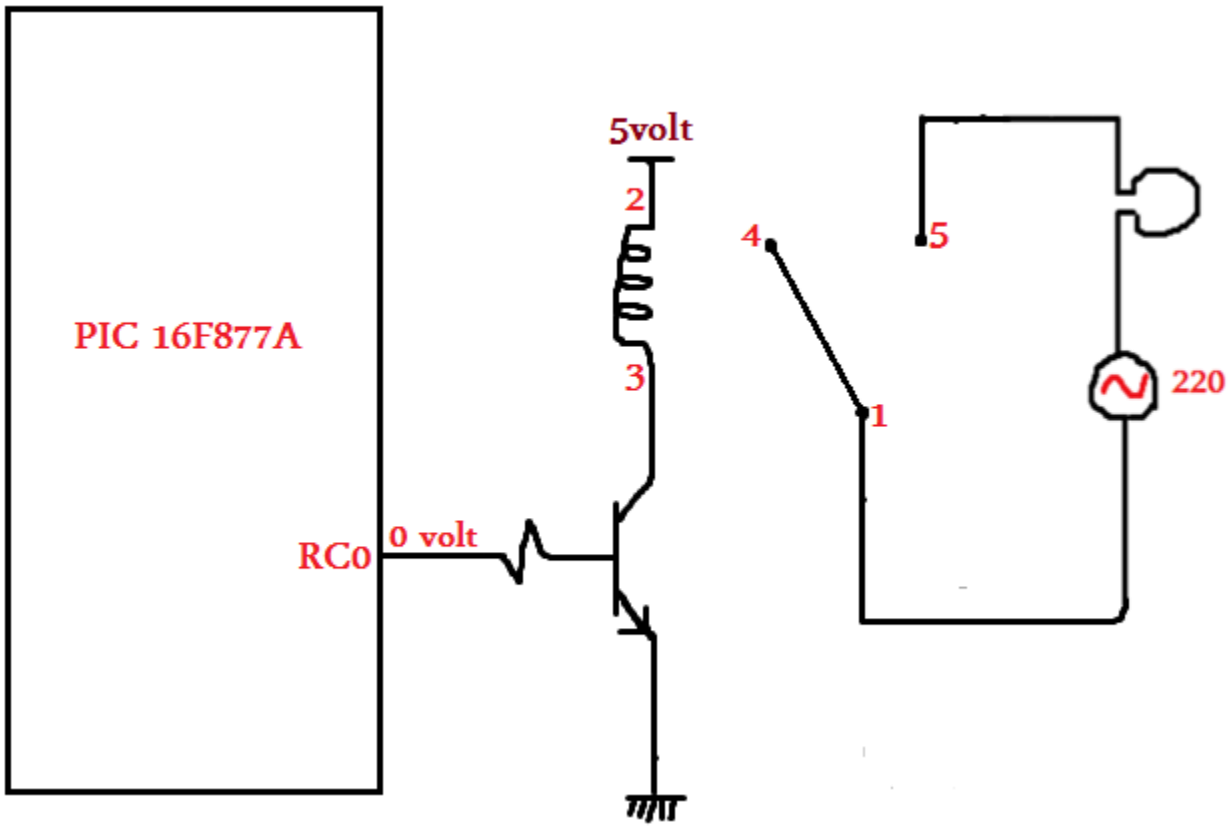




حتى هنا نكون قد فهمنا طريقة عمل الريلاي ... وبالتالي يمكن عن طريق بطارية خمسة فولت أن نتحكم في الجهد الـ ٢٢٠ فولت ... لكن في كتابنا هذا ليس هذا هو المطلوب فالمطلوب هو التحكم في الـ ٢٢٠ فولت من خلال الميكروكنترولر ... وبالتالي يكون الحل البديهي هو الحصول على الخمسة فولت المطلوبين لملف الريلاي من الميكرو حيث أن الميكرو يخرج قيم خمسة فولت أو صفر فولت، وفي هذا الحل خطأ غير ملحوظ يؤدي إلى عدم عمل الريلاي ...

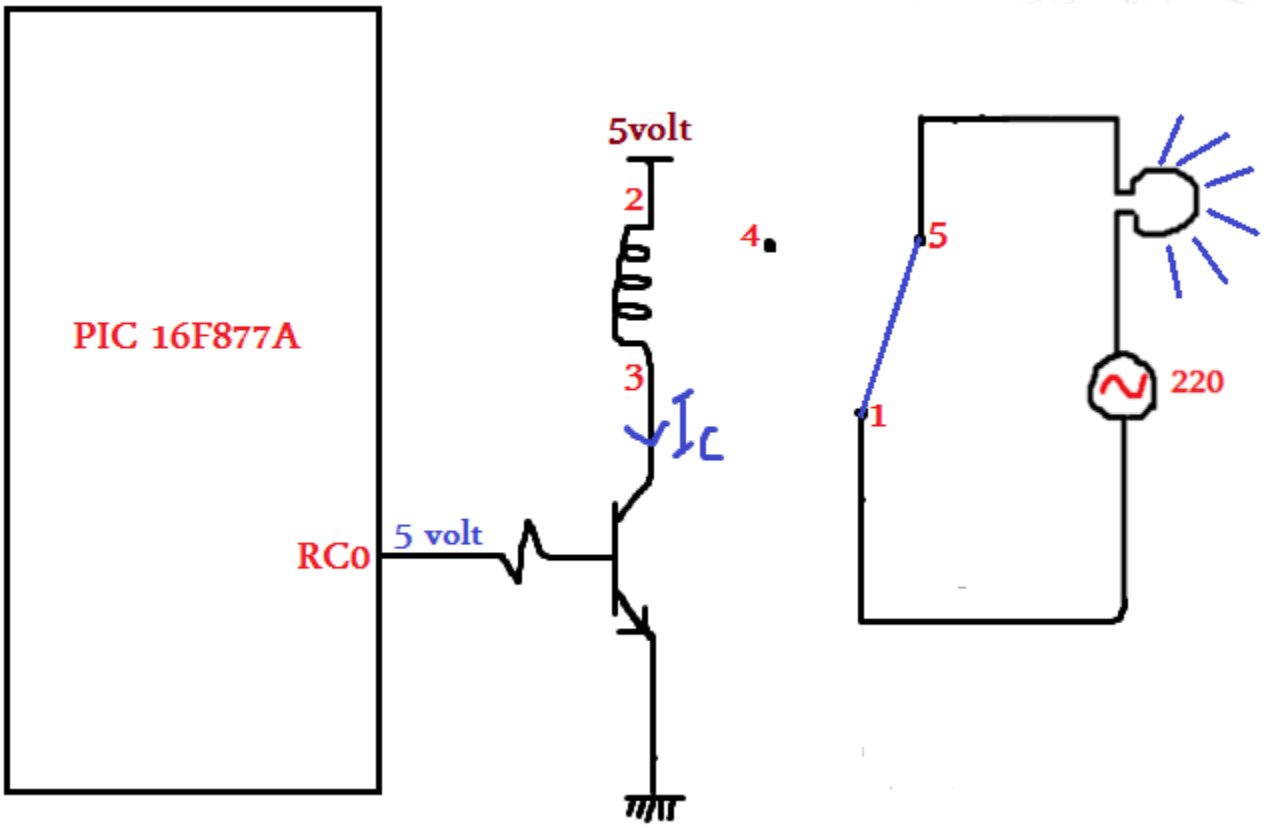
هذا الخطأ يتمثل في قيمة التيار، وذلك لأنه بالرغم من أن الملف يحتاج جهد خمسة فولت فإنه يحتاج أيضا تيار كبير نسبيا – على الأقل أكبر من ٢٥ ميللي أمبير التي تخرج من الميكرو – لكي يولد القوة اللازمة لنقل الـ metal وبالتالي لا يمكن توصيل طرفي الملف مباشرة بالميكرو ... ما الحل إذن ؟؟

يتمثل الحل في استخدام نفس الدائرة السابقة Transistor as a switch ولكن نستبدل الموتور فيها بملف الريلاي أي بالنقطتين ٢ و٣ لتصبح الرسمة الكلية كما هو في بالشكل:



وبالتالي لو أخرج الميكرو صفر فولت سيكون السويتش مفتوح (open circuit) وبالتالي لن يمر تيار في ملف الريلاي وبالتالي لن تتحرك الـ metal من مكانها ولن يصل الجهد على المصباح مما يجعله غير مضيء.

أما إذا أخرجنا خمسة فولت من الميكرو فسيصبح الترانزستور short circuit بالتالي تصبح النقطة ٣ وكأنها متصلة بالخمسة فولت مباشرة مما يجعل التيار يمر في الملف وبالتالي يعمل الريلاي وتتحرك الـ metal من النقطة ٤ إلى النقطة ٥ فتكتمل دائرة المصباح مما يجعله يضئ كما بالشكل الآتي:



مشروع تطبيقي

وبكدة نكون قد تعلمنا دائرة التحكم في الجهود العالية عن طريق الميكروكنترولر، بقي الآن فقط أن نقوم بعمل برنامج بسيط يضئ مصباح ويقوم بإطفائه ...

بالنظر في هذه الفكرة البسيطة نجدها نفس فكرة أول مشروع تم تنفيذه وهو مشروع الفلاش، وذلك لأن المطلوب من الميكرو فقط هو أن يقوم بإخراج خمسة فولت وإخراج صفر فولت ثم تتولي دوائر الهاردوير السابقة تعديل هذا الجهد ليتعامل مع المصباح ...

وفيما يلي تذكير ببرنامج الفلاش على الميكرو سي:

```
void main()
{

   TRISC.B0 = 0;

while(1)
   {
   PORTC.B0 = 1;
    delay_ms(1000);

   PORTC.B0 = 0;
    delay_ms(1000);
   }
}
```

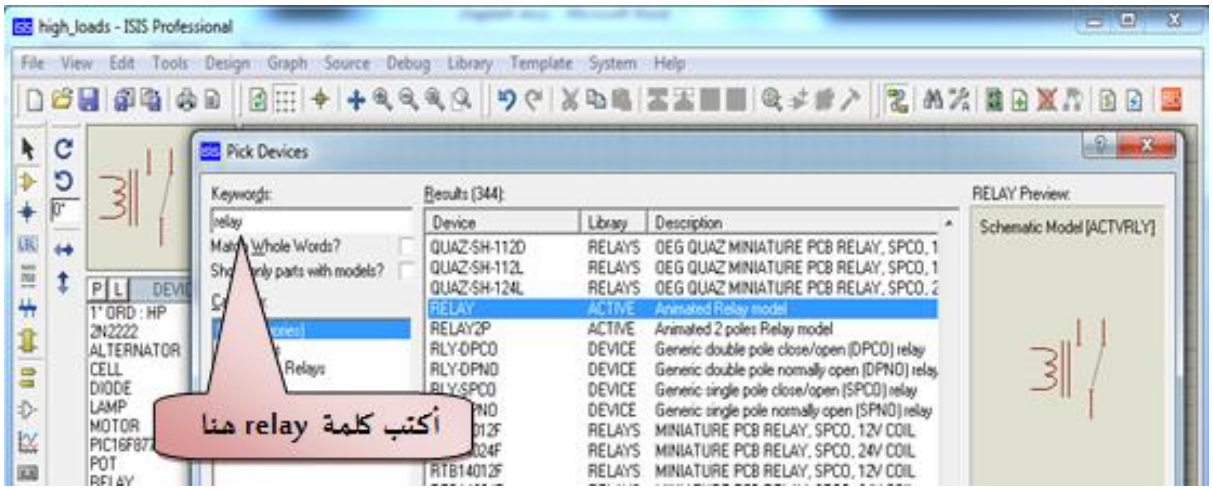
يتم تكرار الاوامر التى بداخلها عدد لا نهائى من المرات

اخراج خمسة فولت على RC0 لمدة ثانية

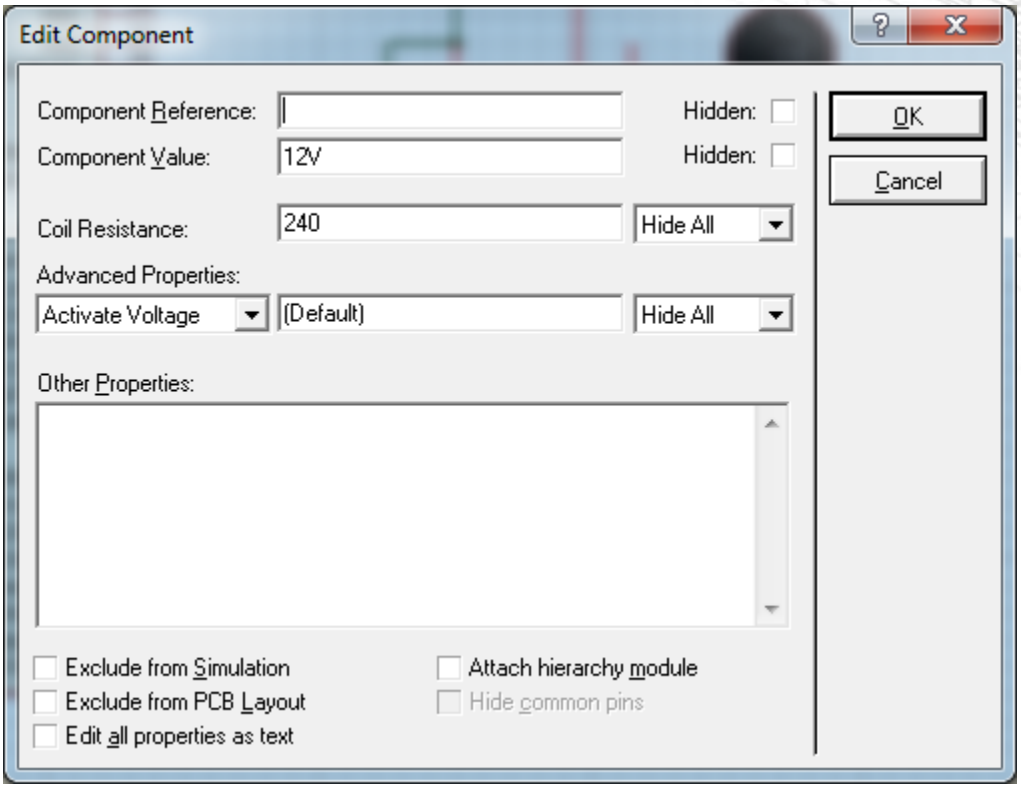
اخراج صفر فولت على RC0 لمدة ثانية

المحاكاة

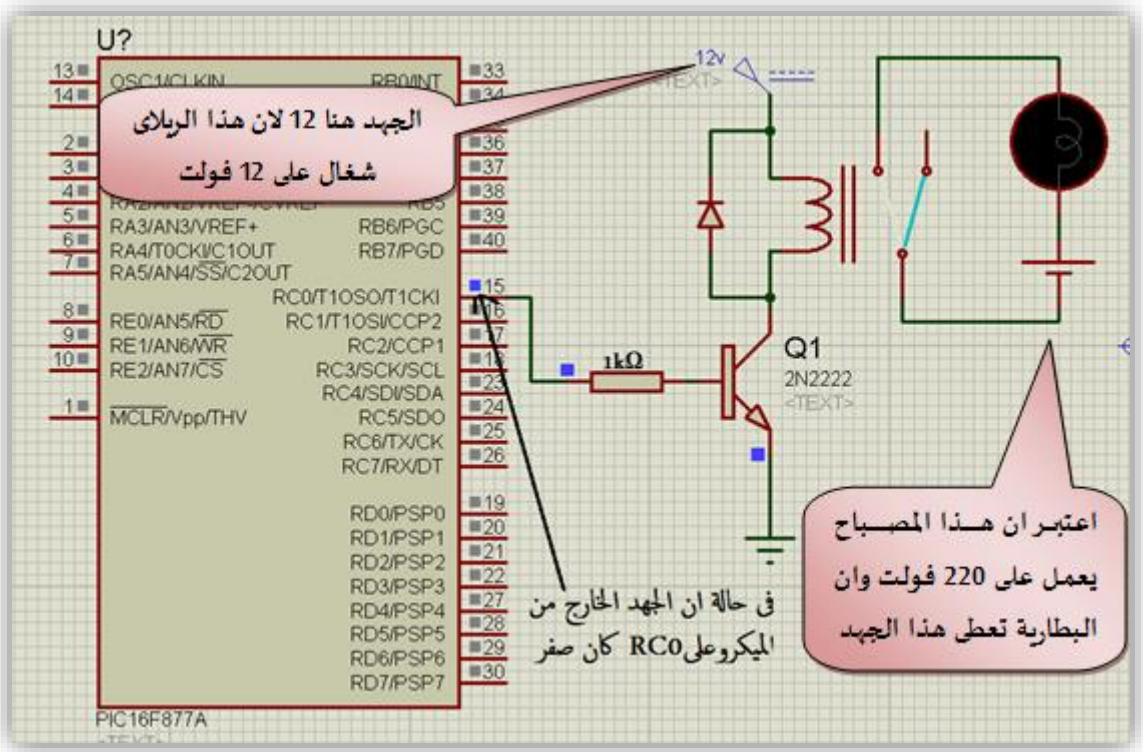
إليكم أولا كيفية الحصول على الريلاي والبطارية في بروتس:



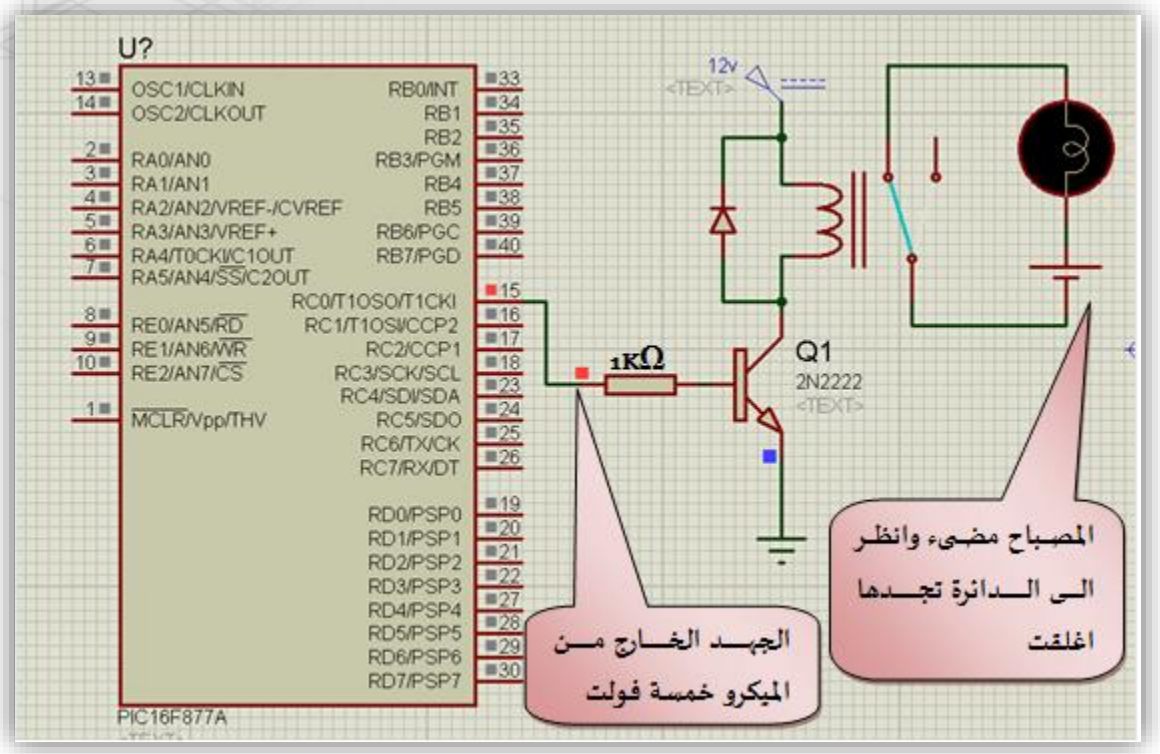
ويمكن الاستعاضة عن الجهد المتردد مجازا بالبطارية وللحصول عليها في بروتس أكتب كلمة battery في خانة البحث وبعد تنزيلها ووضعها في منطقة تصميم الدائرة اضغط عليها مرتين لتعديل قيمة الجهد ليصبح حسب ما تريده أنت، أيضا إذا أردت أن تعلم الجهد الذي يعمل عنده الريلاي أو تعديله في بروتس اضغط عليه مرتين، في الحالتين السابقتين ستظهر لك الشاشة كما في الصورة الأتية والتي من خلالها يمكن تغيير الجهد إلى ٥ أو ١٢ أو ٢٤ أو غير ذلك:

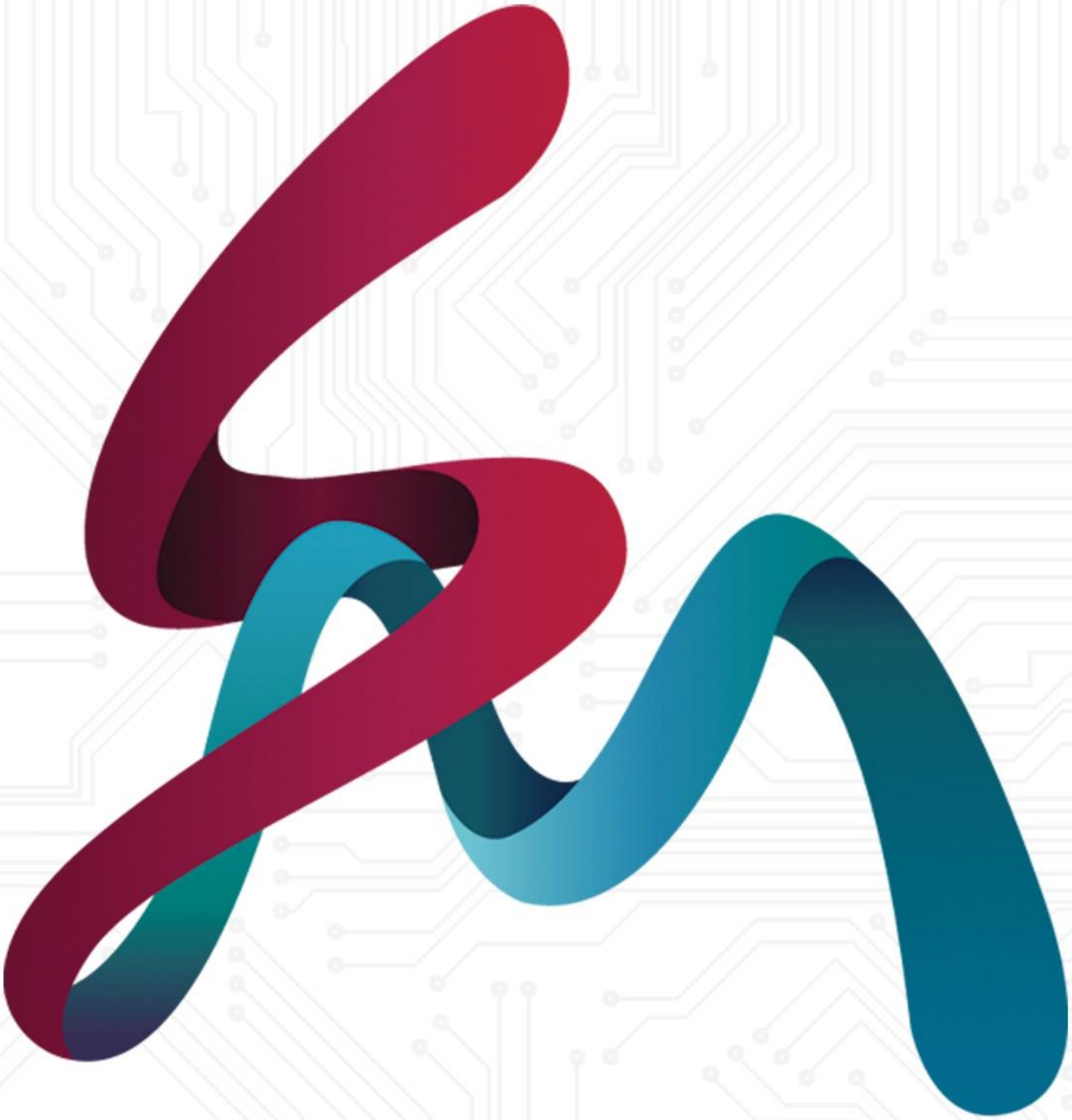


وبالتالي يكون الشكل التالي للدائرة كاملة:



في الصورة السابقة كان خرج الميكرو صفر فولت أما عند إخراج خمسة فولت فسيضيئ المصباح كما بالشكل التالي:





SmartMethods

الأساليب الذكية

كل مايحتاجه المبتكر من أنظمة

إلكترونية وميكانيكية

www. s - m .com.sa

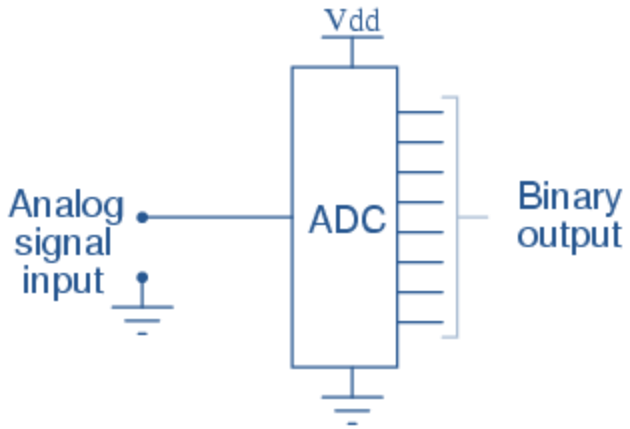
الفصل العاشر

التعامل مع الاشارات التناظرية

تعلمنا في الفصول الماضية كيفية استخدام الميكروكنترولر مع الإشارات الد يجيتال سواء دخل أو خرج والان جاء الدور لكي نتعرف على كيفية التعامل مع الإشارات التماثلية بالميكروكنترولر

ADC Interface

سبق وأشرنا أن المعالجات بطبيعتها تتعامل فقط مع الإشارات الد يجيتال ولكي نستطيع التعامل مع الإشارات التماثلية لابد من تحويلها إلى د يجيتال أولا لكي يستطيع المعالج فهمها، وما يقوم بهذا هو الـ ADC Interface حيث يكون له دخل واحد لاستقبال الإشارة ويكون له عدد رجول في الخرج قد تكون ٨ أو ١٠ أو ١٢ أو ١٦ أو أكثر على حسب الإنترفيس يظهر عليها القيمة الد يجيتال وقد تكون متصلة بالمعالج أو بالذاكرة.



يحتوي الميكرو 16F877A ثمانية رجول لقراءة الإشارات التماثلية Analogue Signals أي أننا يمكنا قراءة ثمانية إشارات مختلفة (من ثمانية سيسنورات مختلفة)، ولكن هناك أنواع من الميكرو قد لا تجد فيها ADC Interface من الأساس، لذلك لابد من قراءة الداتا شيت جيدا لمعرفة ما يحتويه الميكرو من Interfaces.

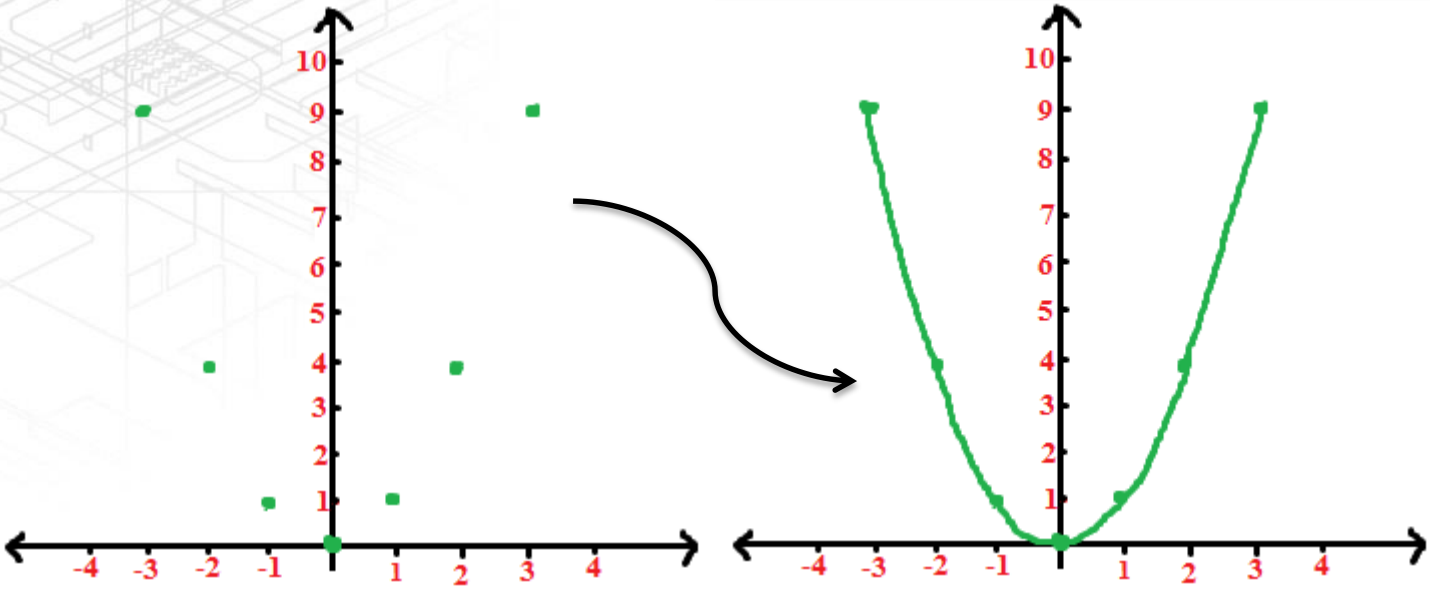
عملية التحويل

تتم عمليات تحويل الإشارات إلى د يجيتال عن طريقة ثلاثة خطوات أساسية هي Sampling ثم Quantization ثم Coding، و نحن لسنا بصدد شرحهم في هذا الكتاب بالتفصيل ولكن بعض المعلومات البسيطة لا تضر ...

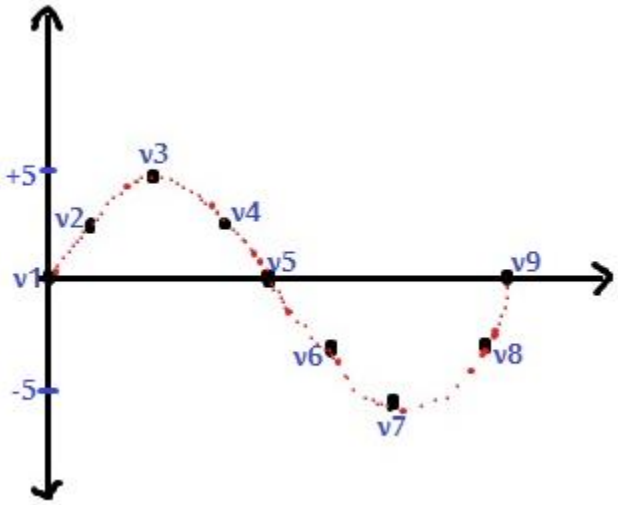
إذا أعطيتك معادلة تربيعية ولتكن مثلا $Y = X^2$ وطلبت منك رسمها فماذا ستفعل ؟؟؟ كما تعلم أغلبنا في الثانوية العامة وما بعدها أو ما قبلها سنقوم بإعطاء قيم معلومة للمتغير X ثم نعوض في المعادلة لتحصل على القيم المقابلة للمتغير Y كما في الجدول الآتي:

X	-3	-2	-1	0	1	2	3
Y	9	4	1	0	1	4	9

ثم نقوم بتوقيع هذه النقط على المحاور ثم نوصل هذه النقط بمنحنى تربيعي لنحصل على المنحنى كما بالشكل:



والشاهد من هذا الجزء هو أنه بالرغم من أن المنحنى يحتوي على ألاف النقاط إلا أنا مجموعة النقط التي قمنا بحساب قيمتها تمثل المنحنى المطلوب وبتوصيلها نحصل على المنحنى المطلوب، وهذا بالضبط هو ما يقوم بفعله الـ ADC Interface حيث يتم وضع إشارة على دخله تمثل المنحنى حيث أن لها قيم مختلفة عند كل لحظة زمنية، ولتكن مثلا إشارة الـ sin فيقوم بأخذ مجموعة من القيم تمثل هذه الإشارة ويتعامل معها، وذلك كما الشكل المجاور.



والسؤال هنا: هل القيم التي يقرأها الميكرو تمثل قيم الجهد الفعلية الموجودة على رجله ؟؟ بمعنى آخر: في الشكل المجاور هل ستكون:

- قيمة V1 تساوى 0 ؟
- قيمة V2 تساوى 2.5 ؟
- قيمة V3 تساوى 5 ؟
- قيمة V4 تساوى 2.5 ؟
- قيمة V5 تساوى 0 ؟
- قيمة V6 تساوى 2.5- ؟
- قيمة V7 تساوى 5- ؟
- قيمة V8 تساوى 2.5- ؟
- قيمة V9 تساوى 0 ؟

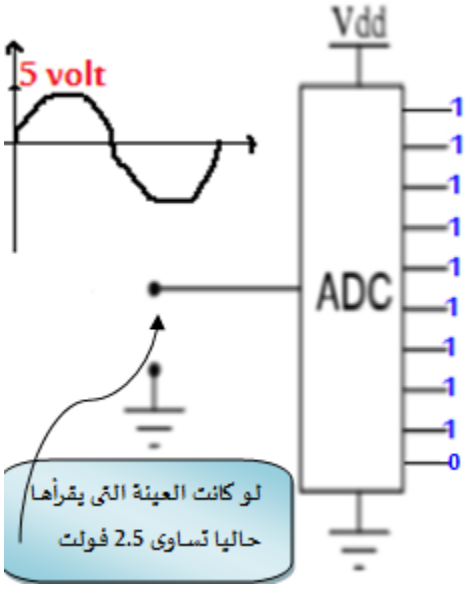
وهكذا إلى آخر باقي القيم على المنحنى، هل إجابة الأسئلة السابقة بنعم أم بلا؟؟ هل هذه هي القيم التي نحصل عليها من الـ ADC Interface ؟؟؟ الإجابة لا، فالقيم التي يخرجها الـ ADC Interface تتوقف على عدد الرجول الموجودة في خرجه، وفى حالة الميكرو 16F877A فان الـ ADC Interface يحتوي على ١٠ رجول في الخرج، وبالتالي فإن أقصى قيمة متاحة لهذا العدد بالنظام الثنائي هي ١٠٢٣ وبالتالي فإن مجال القيم المتاحة عليه من صفر إلى ١٠٢٣، هل هذا يعني أن الميكرو يتعامل مع ١٠٢٣ فولت !!! بالطبع لا فهي قيمة عالية جدا جدا، إذن فعندما يخرج هذه القيمة فأي قيمة يقصدها بها ؟؟

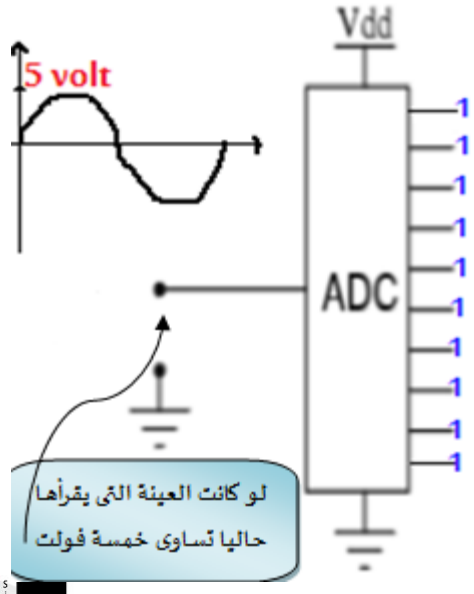
كل ADC Interface له قيمة تسمى الـ reference أو القيمة المرجعية، وهذه القيمة تمثل أقصى قيمة يمكنه قراءتها، وهي هنا في الميكرو تساوي خمسة فولت، لكن يمكنك تغييرها من خلال الأوامر.

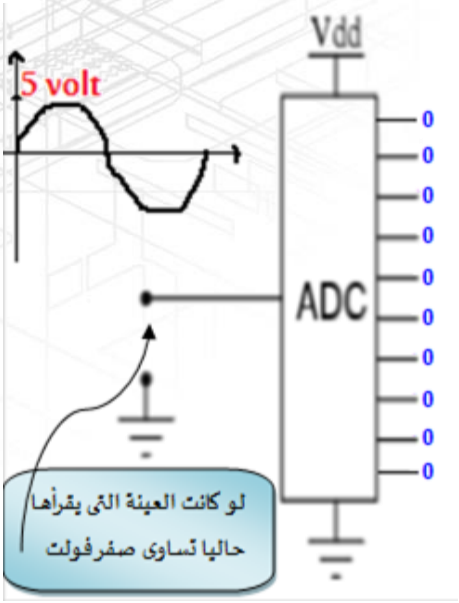
وبالتالي فإن الـ ADC Interface يوضع على دخله قيم *أنالوج* من صفر إلى ٥ فولت (وليس إما صفر وإما خمسة كما تعودنا بل متاح القيم البينية)، و يخرج قيم *د يجيتال* من صفر إلى ١٠٢٣ على خرجه، وبالتالي نستطيع استنتاج أنه يحول من range إلى range آخر بمعنى أن:

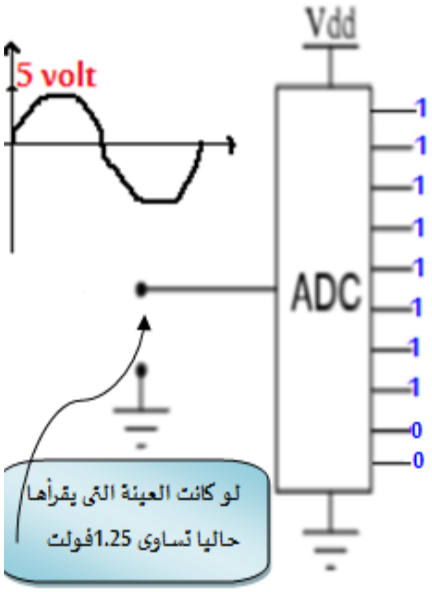
- القيمة ٥ فولت على الدخل يخرجها ١٠٢٣ وهي بالثنائي بالباينارى ١١١١١١١١١١
- القيمة ٢,٥ (نصف القيمة العظمى للدخل) يخرجها ٥١١ (نصف القيمة العظمى للخرج) وهي بالثنائي ٠١١١١١١١١١
- القيمة ١,٢٥ (ربع القيمة العظمى للدخل) يخرجها ٢٥٥ (ربع القيمة العظمى للخرج) وهي بالثنائي ٠٠١١١١١١١١
- القيمة صفر على الدخل يخرجها صفر على الخرج ... وهكذا

وهذا الرسومات توضح ما سبق:









وبالتالي لو قمنا بتخزين القيمة المقروءة والتي تتراوح بين ٠ و ١٠٢٣ في متغير X ونريد أن نحصل على القيمة الفعلية التي تتراوح بين ٠ و ٥ فولت لنتعامل معها، يمكننا التعامل مع معادلة بسيطة للنسبة والتناسب كما يلي:

```
Y = X * 5 /1023;
```

حيث Y هي القيمة الفعلية للجهد.

عمليا تكون عدد العينات أو النقاط التي يأخذها الـ ADC interface كبير نسبيا وليس قليلا كما بالأشكال السابقة فهو لمجرد الشرح فقط ...

حتى هنا نكون قد انتهينا من فهم الطريقة التي يتعامل بها الـ ADC Interface.

دوال الميكروسي

والآن حان الوقت لنتعرف على دوال الميكروسي المستخدمة في التعامل مع الـ ADC Interface، والتي يمكن الحصول عليها كما أشرنا في الفصول السابقة من نافذة المساعدة بالضغط على زر F1 من لوحة المفاتيح أو من قائمة Help ثم Help فتظهر نافذة المساعدة كما في الصورة التالية، ومن الجانب الأيسر نختار مكتبات البرنامج ثم نختار مكتبات الهاردوير Hardware Libraries ثم نختار مكتبة الـ ADC Library، فتظهر لنا الدوال المستخدمة مع الـ ADC interface كما يلي:

الدالة الأولى

```
ADC_Init();
```

تستخدم لتهيئة الـ ADC Interface، ومن أمثلة هذه التهيئة هنا مثلا تحديد الزمن بين كل قيمة يتم قراءتها وأخرى، وتظهر في جدول المثال السابق في الفرق بين قيمتين متتاليتين للمتغير X، بالإضافة لإجراءات أخرى يتم ضبطها، وهذه الدالة يتم كتابتها داخل الدالة الرئيسية.

الدالة الثانية

```
ADC_Read(2);
```

وهى التي تستخدم لقراءة القيم من رجول الدخل، وحيث أنه يوجد ٨ دخول في البيك 16F877A يمكن استخدامهم في قراءة الإشارات الأنالوج فلابد أن نحدد للدالة أي هذه الرجول ستقرأ منها وهو ما يتمثل في

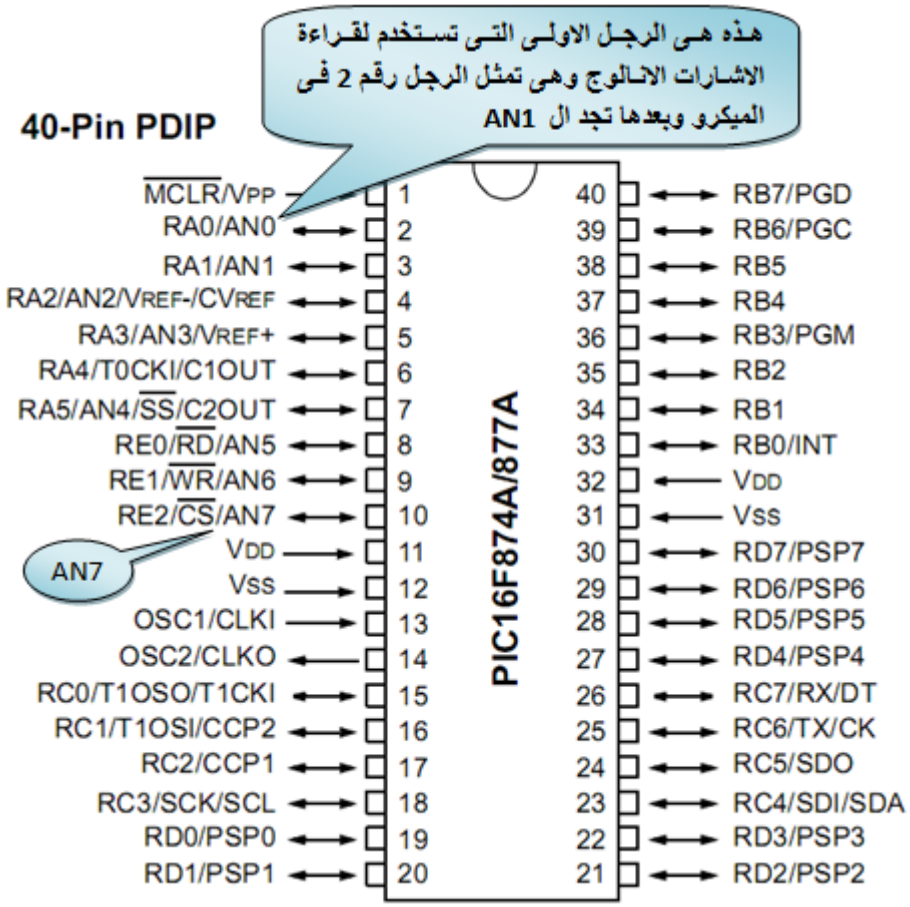
الرقم بين الأقواس، فمثلا بفرض توصيل السينسور المراد قراءة إشارته الأنالوج على الرجل AN0 نكتب الدالة كالآتي:

```
ADC_Read(0);
```

وإذا كان متصل على الرجل AN5 مثلا فتصبح الدالة هكذا:

```
ADC_Read(5);
```

وهكذا ... وفي هذه الصورة تظهر أماكن الرجول المستخدمة في قراءة الإشارات الأنالوج في الميكرو المذكور:



المشروع التطبيقي الأول

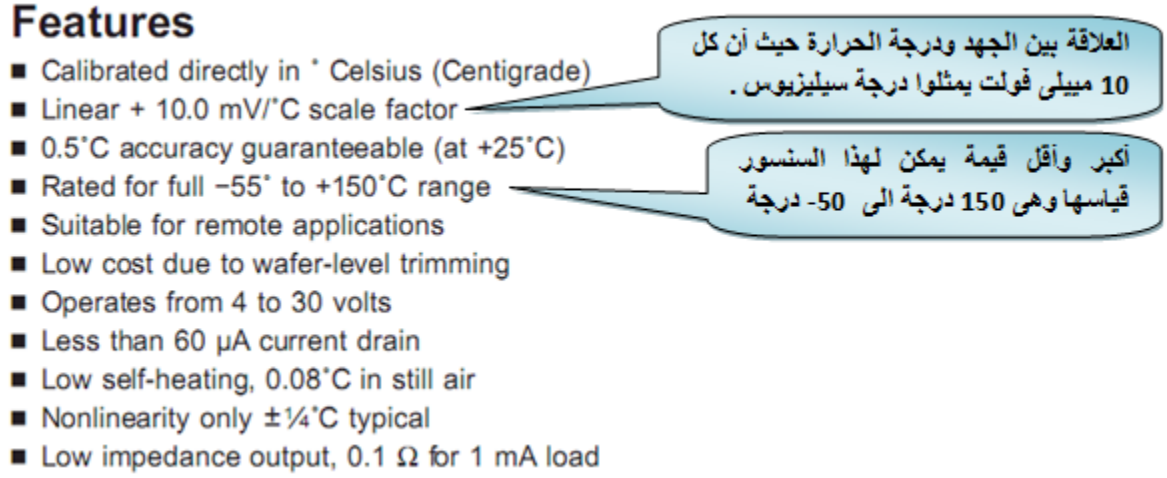
فكرة المشروع بسيطة جدا وهي التحكم في درجة حرارة مكان ما ... فإذا زادت درجة حرارة المكان عن ٣٠ درجة يتم تشغيل التكييف وإذا قلت عن ذلك يتم فصله، على أن يتم عرض قيمة درجة الحرارة الحالية على LCD.

ولكن قبل الخوض في هذا المشروع يلزم في البداية التعرض للسنسور المستخدم في قياس درجة الحرارة ببعض التوضيح:

التعرف على سينسور درجة الحرارة

سينسور درجة الحرارة يقوم بتحويل درجة الحرارة إلى جهد قيمته يتناسب مع درجة الحرارة التي يقيسها، ثم نقوم نحن بقراءة قيمة هذا الجهد بالميكروكنترولر، وعن طريق معرفة العلاقة بين درجة الحرارة والجهد والتي نحصل عليها من الداتا شيت الخاصة بالسينسور فانه يمكنا تحويل قيمة الجهد التي قرأها الميكرو إلى درجة حرارة الحالية.

هناك موديلات مختلفة من السينسورات، والسينسور الذي سنستخدمه في هذا الكتاب يحمل رقم الموديل LM35 ويمكنك تنزيل الداتا شيت الخاصة به لمعرفة خصائصه، ولكن ما يهمنا من هذه الخصائص خاصيتان: الجهد المطلوب له لكي يعمل، والعلاقة بين الجهد الذي يخرج منه وقيمة درجة الحرارة، وعندما نقوم بفتح الداتا شيت ستجد أن الخواص التي نريدها موجودة في أول صفحة كما هو موضح بالشكل الآتي:



ومن هذه المعلومات يمكن استنتاج علاقة نسبة وتناسب أخرى تحول الجهد الخارج من السينسور إلى درجة حرارة كما يلي:

$$10\ mVolt \quad \rightarrow \quad 1^{\circ}C$$

$$\therefore 10 * 10^{-3} volt = 10^{-2} Volt \quad \rightarrow \quad 1^{\circ}C$$

$$Y \quad \rightarrow \quad Z$$

$$\therefore \boxed{Z = Y \times 100}$$

حيث Z تمثل درجة الحرارة و Y تمثل الجهد الخارج من السينسور، مع مراعاة العلاقة السابق استنتاجها:

$$Y = X \times \frac{5}{1023}$$

حيث Y تمثل قيمة الجهد.

برنامج الميكروسي

أول خطوة تتمثل في تحديد الموديولات التي سنحتاجها في المشروع وذلك لكتابة دوال الـ initialization الخاصة بهم، وفى هذا المشروع نحتاج موديول الـADC وموديول الشاشة، وبالتالي تكتب الدالتين في بداية الدالة الرئيسية:

```
ADC_Init();
Lcd_Init();
```

وأيضا نقوم بتحديد المخرج الذي سيتم توصيل الشاشة عليه وليكن PORTD كما هو الحال في الفصول الماضية، ثم نقوم بكتابة أوامر التوصيل الخاصة بالشاشة في بداية البرنامج والتي تعلمنا كتابتها سابقا بأكثر من طريقة:

```
// LCD module connections
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// End LCD module connections
```

ثم نقوم بالتبعية بتحديد باقي أنواع الدخل والخرج:

- نحتاج خرج لتوصيل دائرة التكييف وليكن RC0.
- نحتاج دخل لقراءة إشارة الأنالوج وليكن AN0.

ثم تكون الدالة الأساسية كما يلي:

```
int temp;  int i ;  char txt[7];

void main()
{
  trisc.B0=0;        portc.B0=0;
  ADC_Init();        Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);
  Lcd_Out(1,1,"Temp=");

      while(1)
      {
                temp = ADC_Read(0);
                temp=temp*500.0/1023.0;
                IntToStr(temp, txt);
                Lcd_Out(1,8,txt);

                if(temp>30)
                {
                   portc.B0=1;
                   Lcd_out (2,1,"Overheat");
                }
                else
                {
                   PORTC.B0=0;
                   Lcd_out (2,1,"Normal    " );
                }
      }
}
```

ولكي نفهم البرنامج يمكن تقسيمه إلى أجزاء كالآتي:

السطر الأول يتمثل في بعض المتغيرات التي سنستخدمها أثناء البرنامج، أما الجزء التالي:

```
void main()
{
  trisc.B0=0;        portc.B0=0;
  ADC_Init();        Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);
  Lcd_Out(1,1,"Temp=");
```

لكى نجعل RC0 تعمل كخرج حيث انها موصل عليها دائرة التكييف

لتهيئة الموديولات

لعرض كلمة Temp = على الشاشة وكتبتها قبل ال while لاننا نريد كتابتها مرة واحدة وتظل موجودة

```
while(1)
{
    temp = ADC_Read(0);
    temp=temp*500.0/1023.0;
    IntToStr(temp, txt);
    Lcd_Out(1,8,txt);

    if(temp>30)
    {
      portc.B0=1;
      Lcd_out (2,1,"Overheat");
    }
    else
    {
      PORTC.B0=0;
      Lcd_out (2,1,"Normal    " );
    }
}
```

هذه الدالة تقرأ الاشارة من ال AN0 وتخزنها فى المتغير temp

للحصول على قيمة الجهد الفعلية التى تم قرائتها

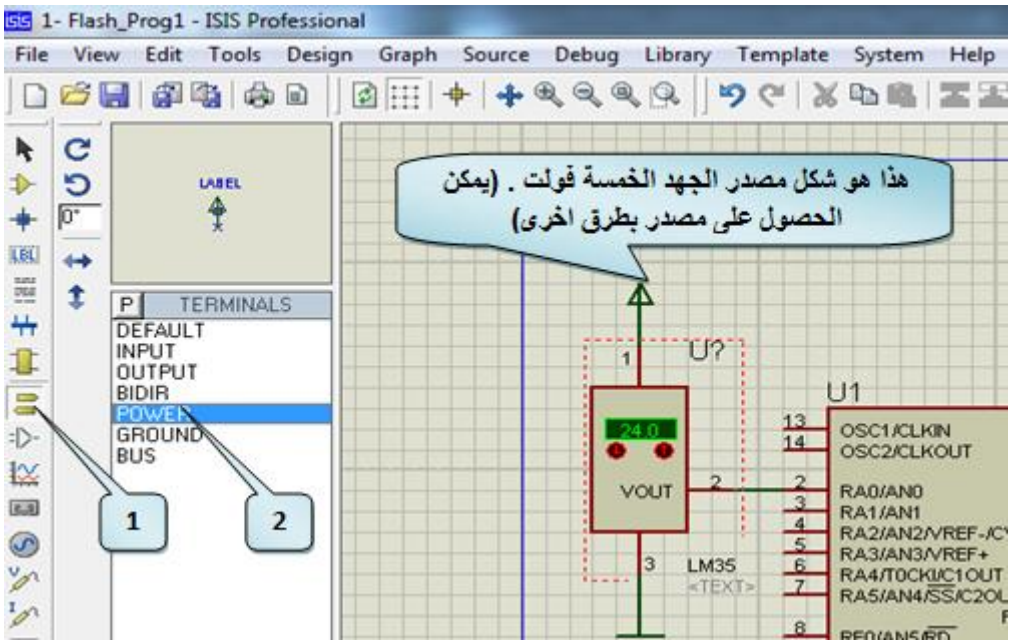
لعرض قيمة درجة الحرارة على الشاشة وتلاحظ اننا حددنا العمود رقم 8 وذلك لان كلمة temp مكتوبة فى اول

لو كانت درجة الحرارة اكبر من 30 يخرج 5 فولت على RC0 ليشغل التكييف

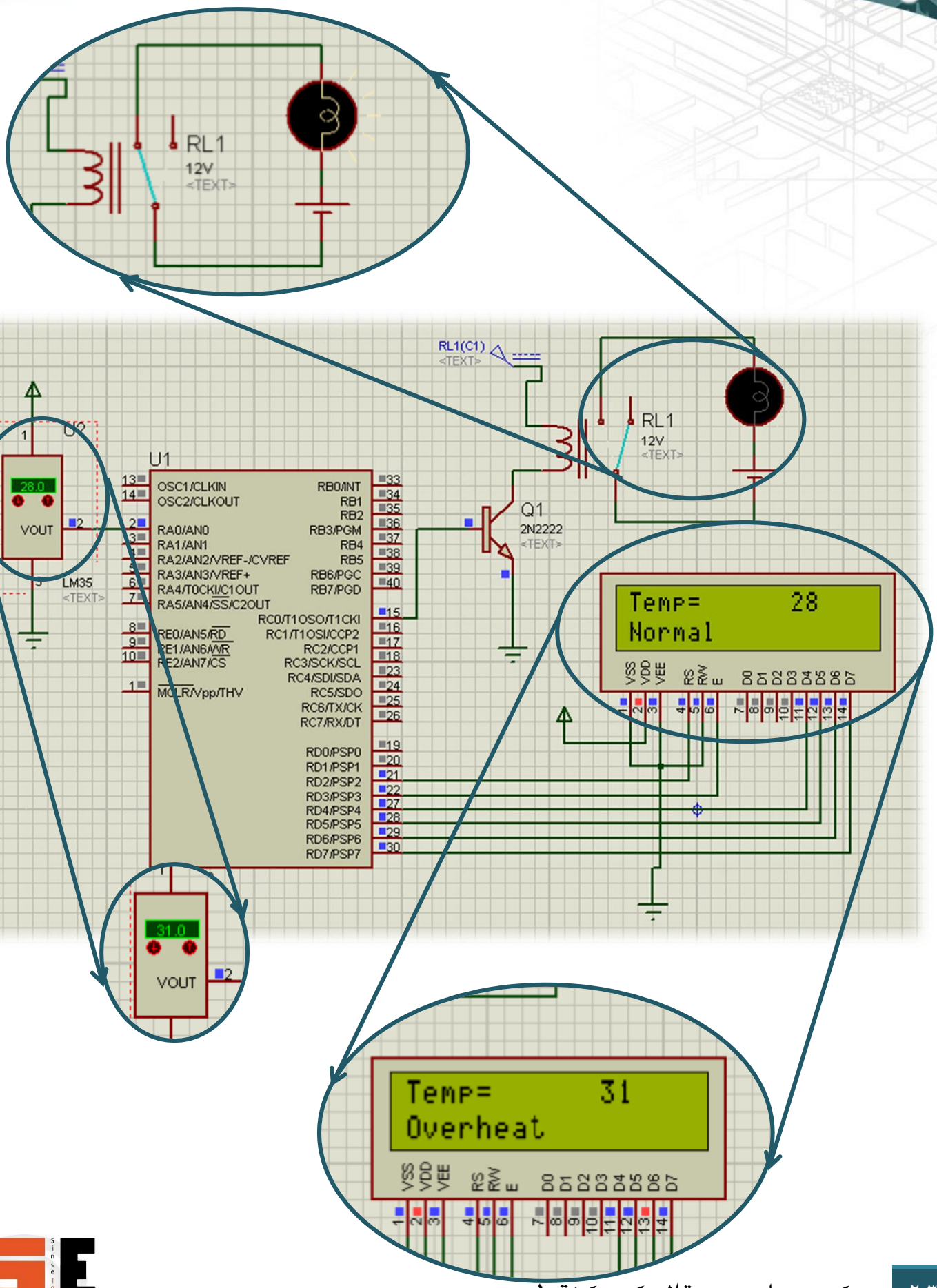
اذا لم تكن درجة الحرارة اكبر من 30 يخرج0 فولت على RC0 ليطفىء التكييف

المحاكاة

للحصول على السينسور في بروتس أكتب LM35 في خانة البحث. أيضا دعنا نتعرف على طريقة أخرى للحصول على مصدر جهد خمسة فولت بطريقة أخرى:

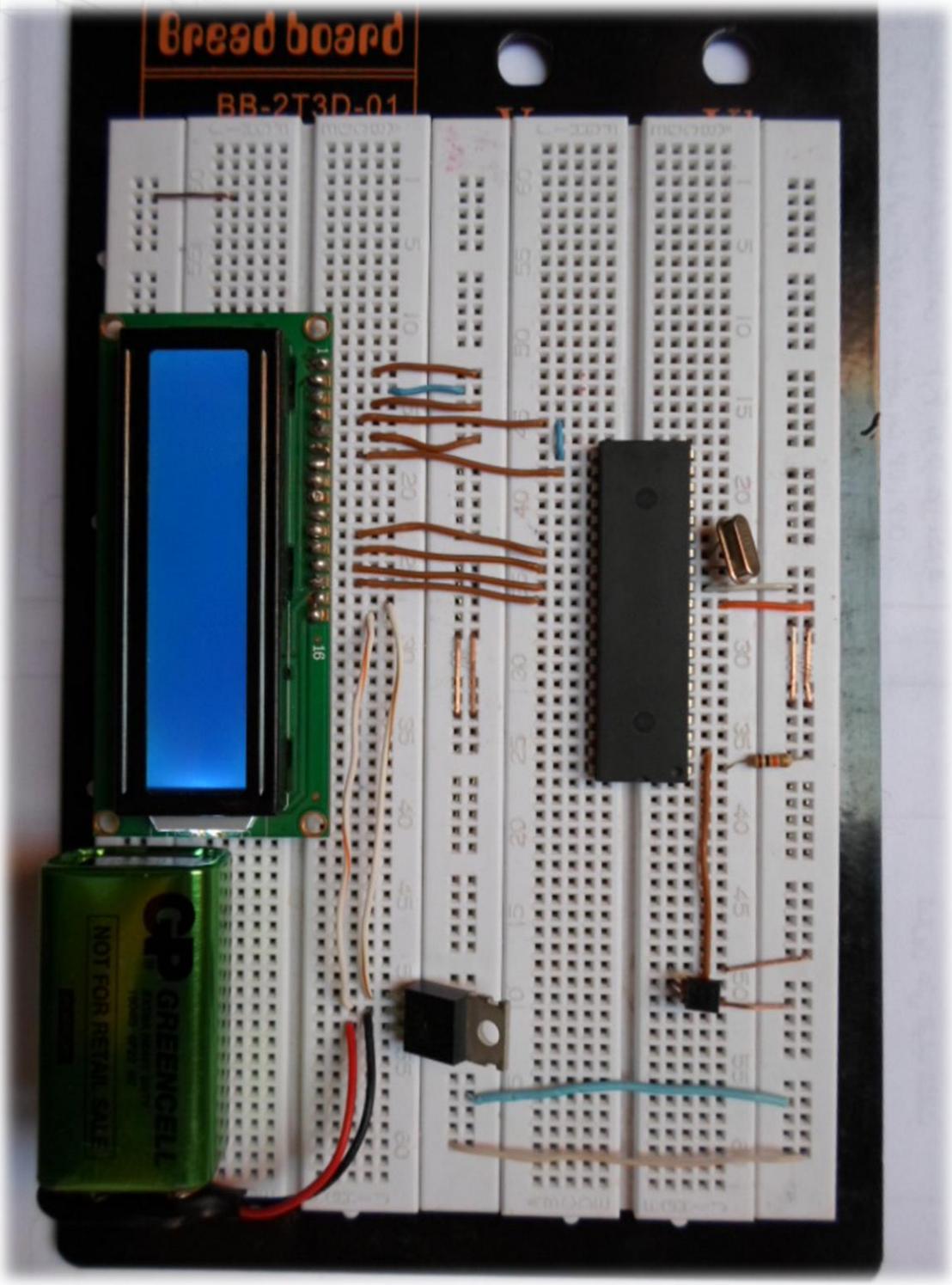


أيضا في بروتس لا يوجد تكييف وبالتالي يمكن التعبير عنه بأي مؤشر للتشغيل وليكن اللمبة الموجودة في الدائرة:

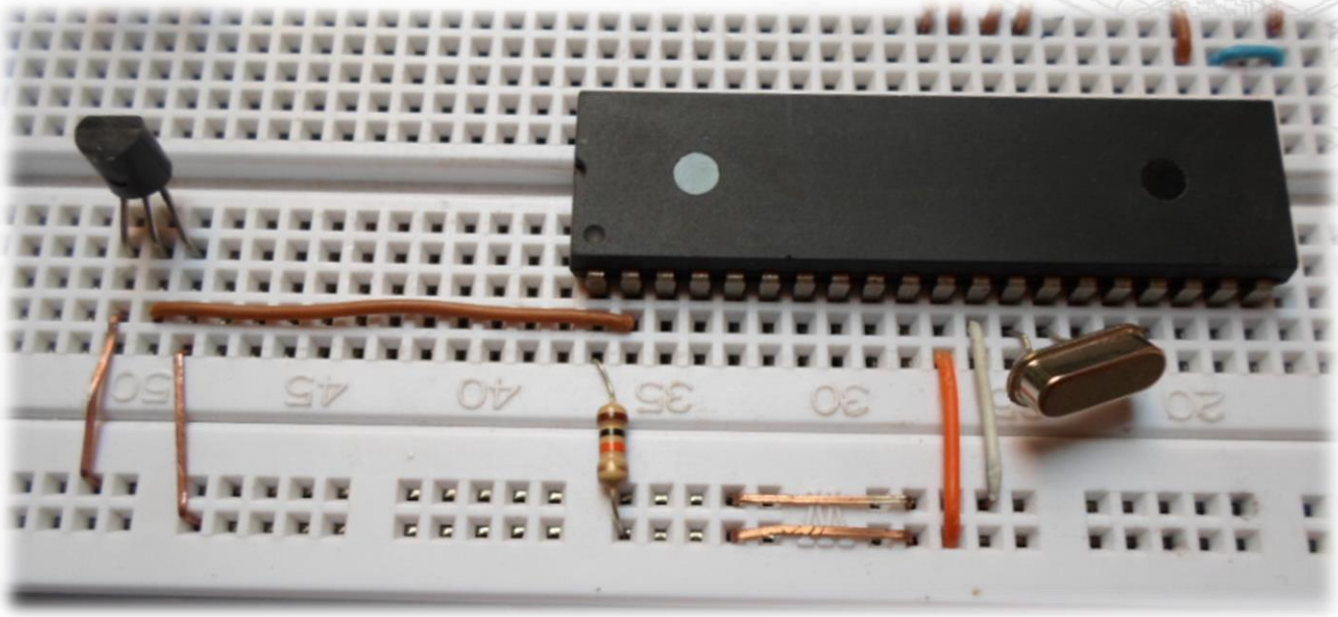


الهاردوير

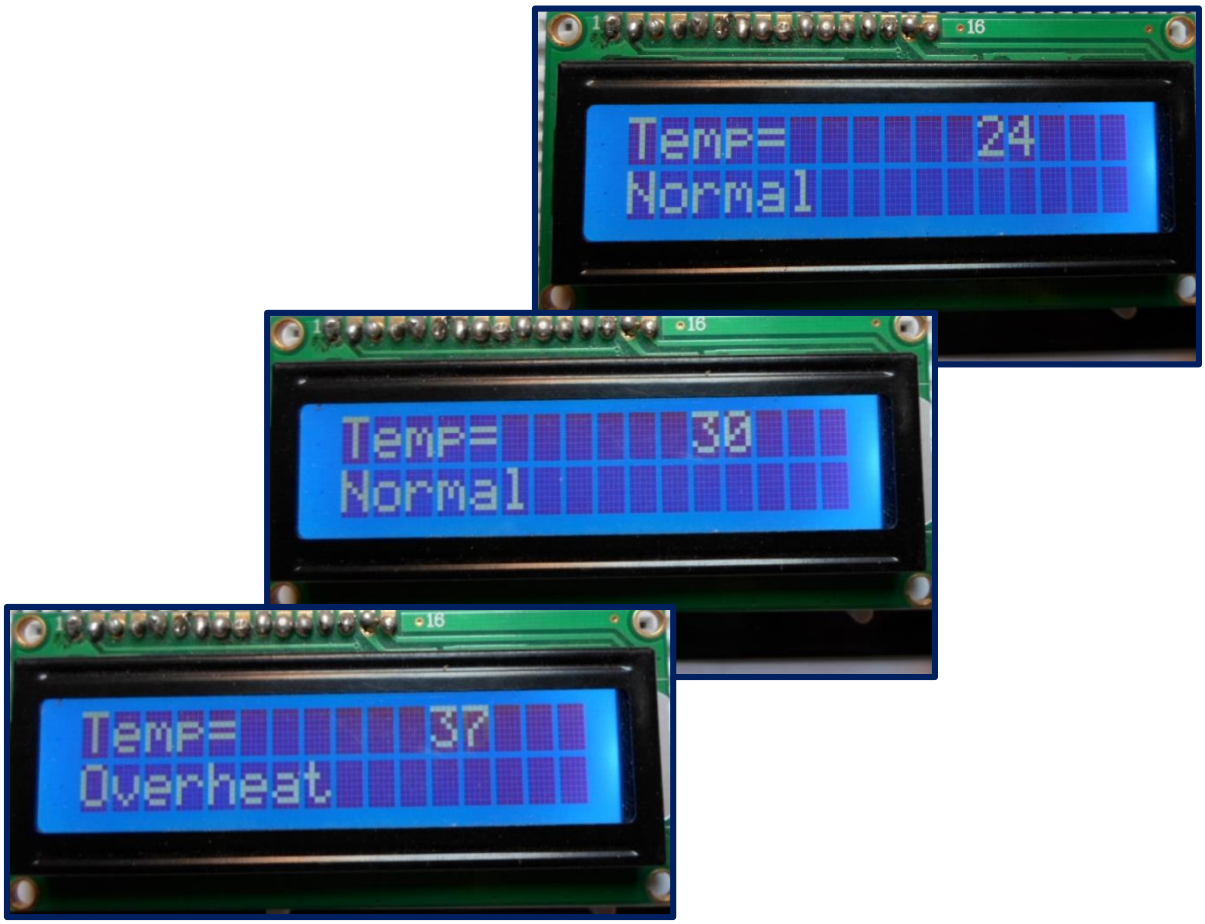
في صور الهاردوير التالية لم أقم بتوصيل دائرة الـ ٢٢٠ فولت ولكن يمكنك أنت توصيلها كما تعلمتها من قبل:



وفيما يلي صورة عن قرب لطريقة توصيل السينسور على الميكرو:



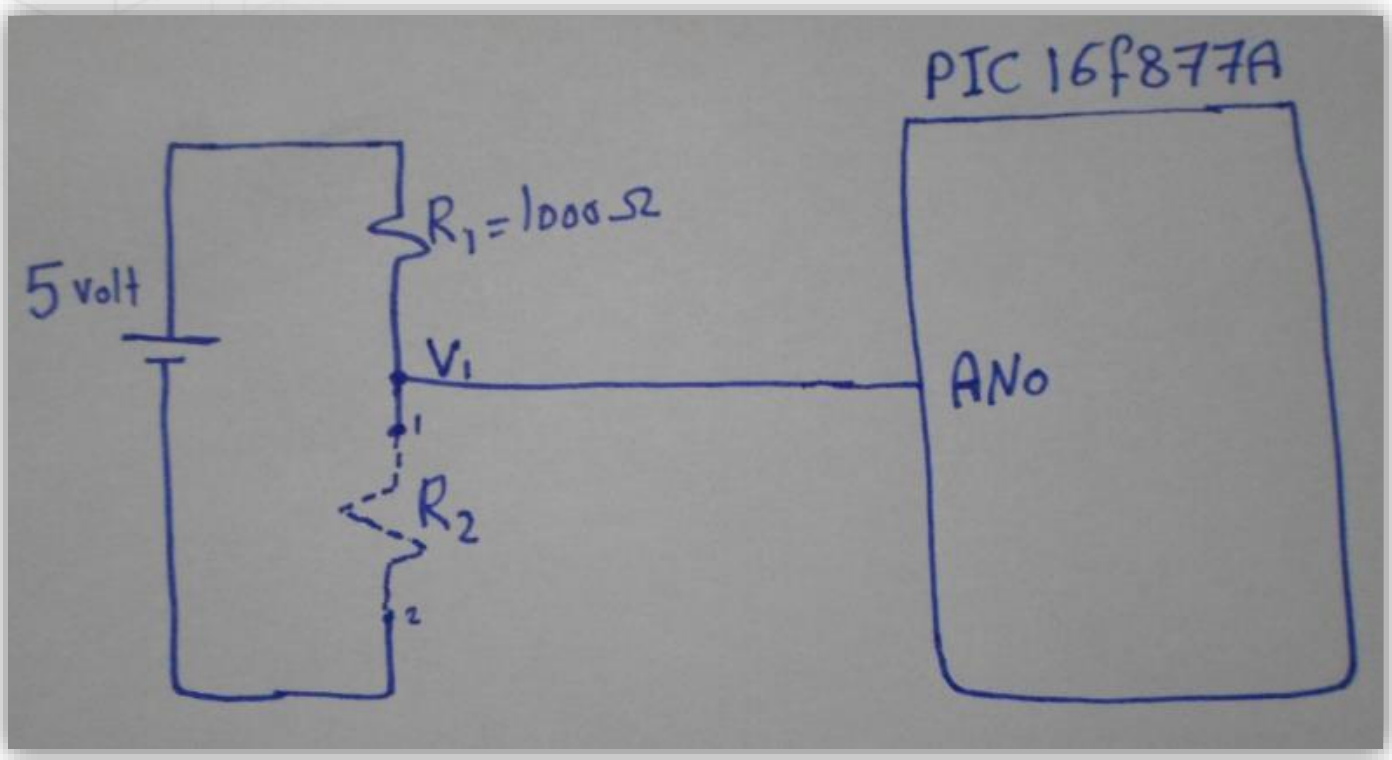
ولاحظ أيضا في الصور التالية قيمة درجة الحرارة ووصفها على الشاشة:



المشروع التطبيقي الثاني

مطلوب عمل جهاز يقيس قيمة مقاومة غير معلومة وهو ما يسمى Ohmmeter.

تعتمد فكرة عمله على دائرة الـ Voltage Divider والممثلة في هذا الشكل:



حيث أن R2 هي المقاومة الغير معلومة القيمة، وعلاقة الجهد بالمقاومة في هذه الدائرة شهيرة وهي كما يلي:

$$V_1 = 5 \times \frac{R_2}{1000 + R_2}$$

حيث أن الجهد V1 هو القيمة التي يقرأها الميكروكنترولر على دخل من دخول الأنالوج وبالتالي فهي قيمة معلومة داخل البرنامج، وبوضع المجهول فقط في المعادلة السابقة وهو المقاومة في طرف واحد تكون العلاقة كالتالي:

$$\therefore 1000V_1 + R_2 \times V_1 = 5R_2$$

$$\therefore R_2 \times V_1 - 5R_2 = -1000V_1$$

$$\therefore \boxed{R_2 = \frac{1000 \times V_1}{5 - V_1} \Omega}$$

البرنامج

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

float V1; unsigned int RES2;  int i ;  char txt[12];
void main()
{
  ADC_Init();       Lcd_Init();
  Lcd_Cmd(_LCD_CURSOR_OFF);
  Lcd_Out(1,1,"Resistance value");
      while(1)
      {
          V1 = ADC_Read(0);
          V1 = V1*5.0/1023.0;

          RES2 = (1000*V1)/(5-V1);
          RES2 = RES2+1;

          IntToStr(RES2, txt);
          Lcd_Out(2,1,txt);
      }
}
```

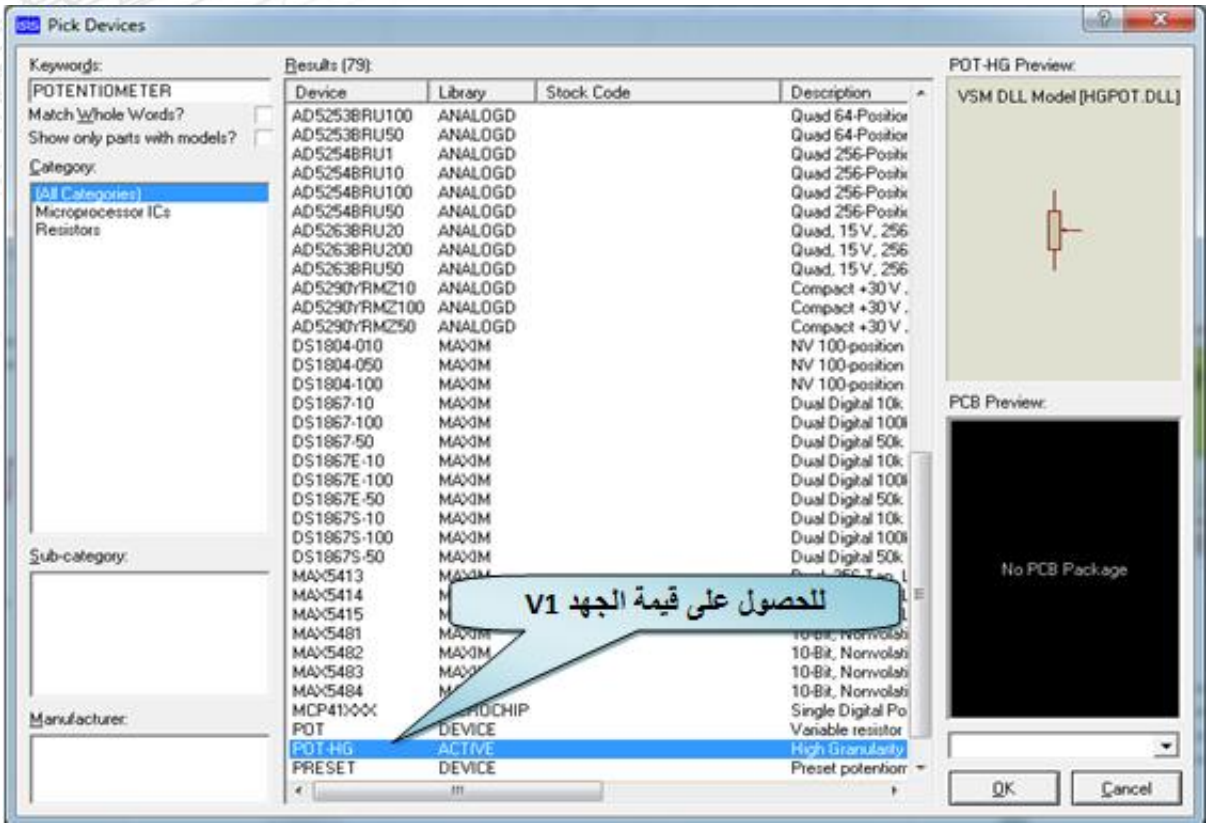
للحصول على قيمة الجهد V1

للحصول على قيمة المقاومة من الجهد في الامر الثانى أضفت واحد على القيمة لتعديل الدقة ويمكنك عدم كتابته

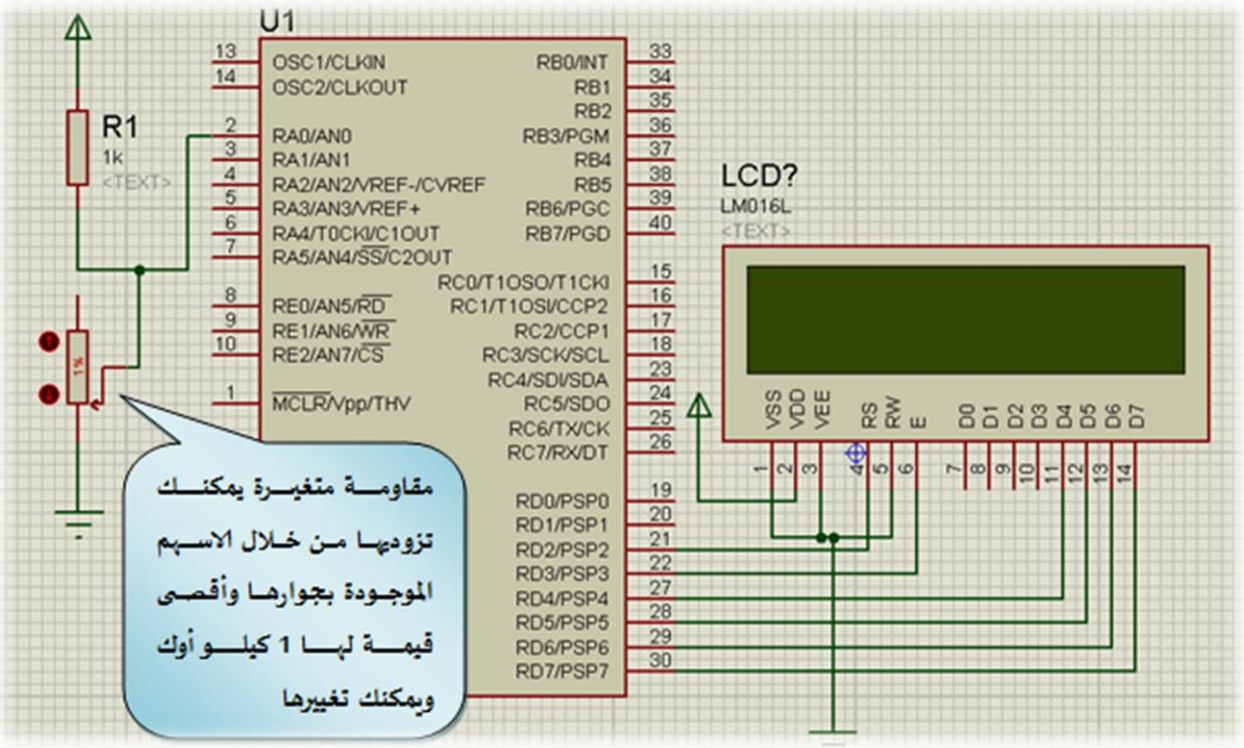
لعرض قيمة المقاومة على السطر الثانى فى الشاشة

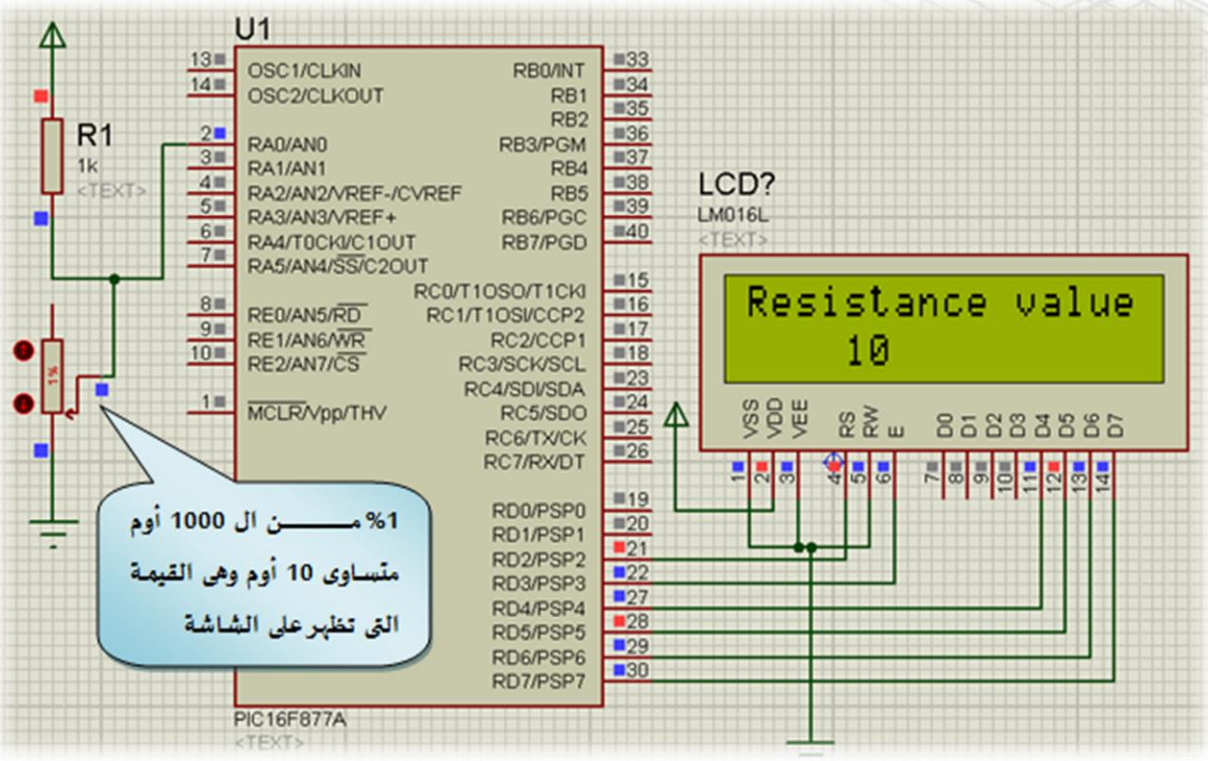
الدائرة

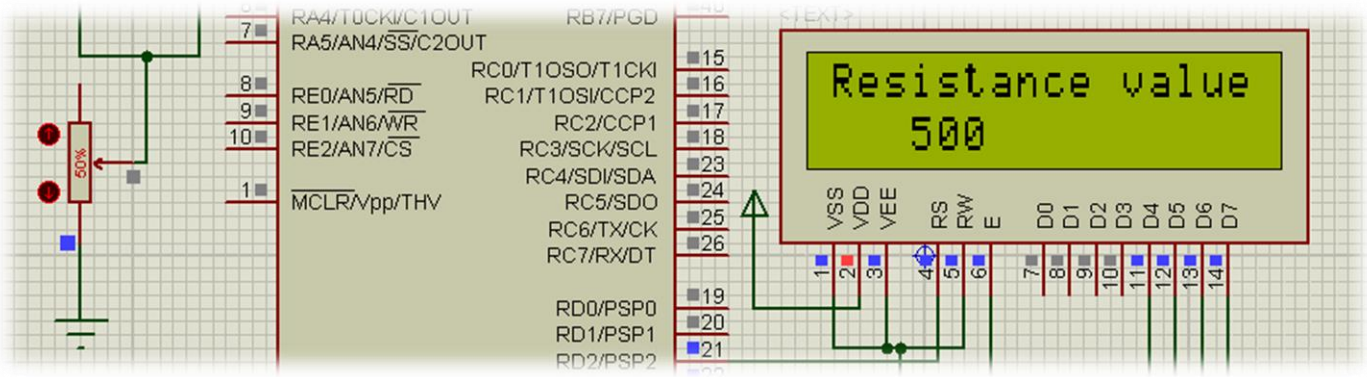
يمكن وضع مقاومة متغيرة كمقاومة مطلوب قياسها وذلك لتغييرها أثناء المحاكاة وملاحظة هذا التغيير على القراءة المعروضة على الشاشة، وللحصول على مقاومة متغيرة في بروتس نكتب كلمة potentiometer في خانة البحث كما بالشكل:

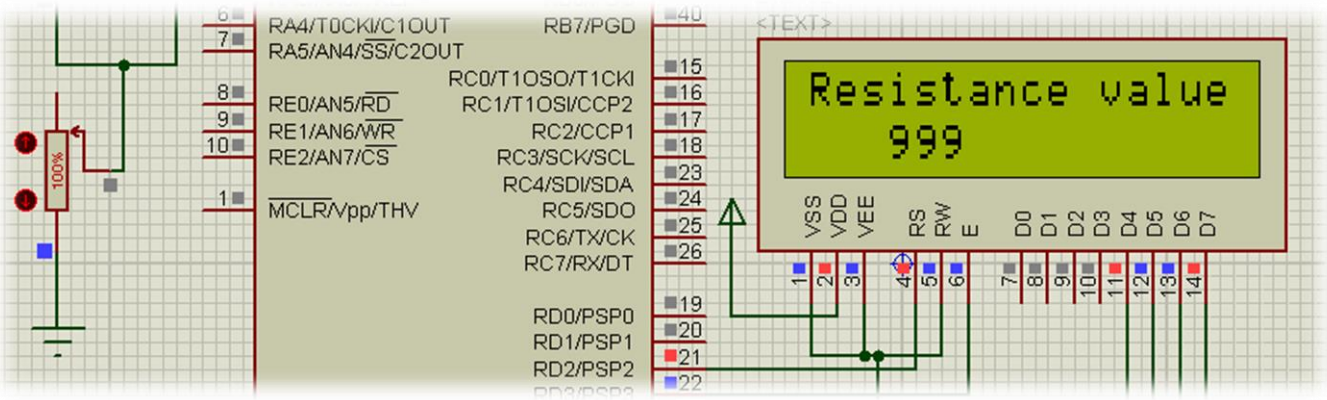


ثم يكون شكل الدائرة كالتالي:

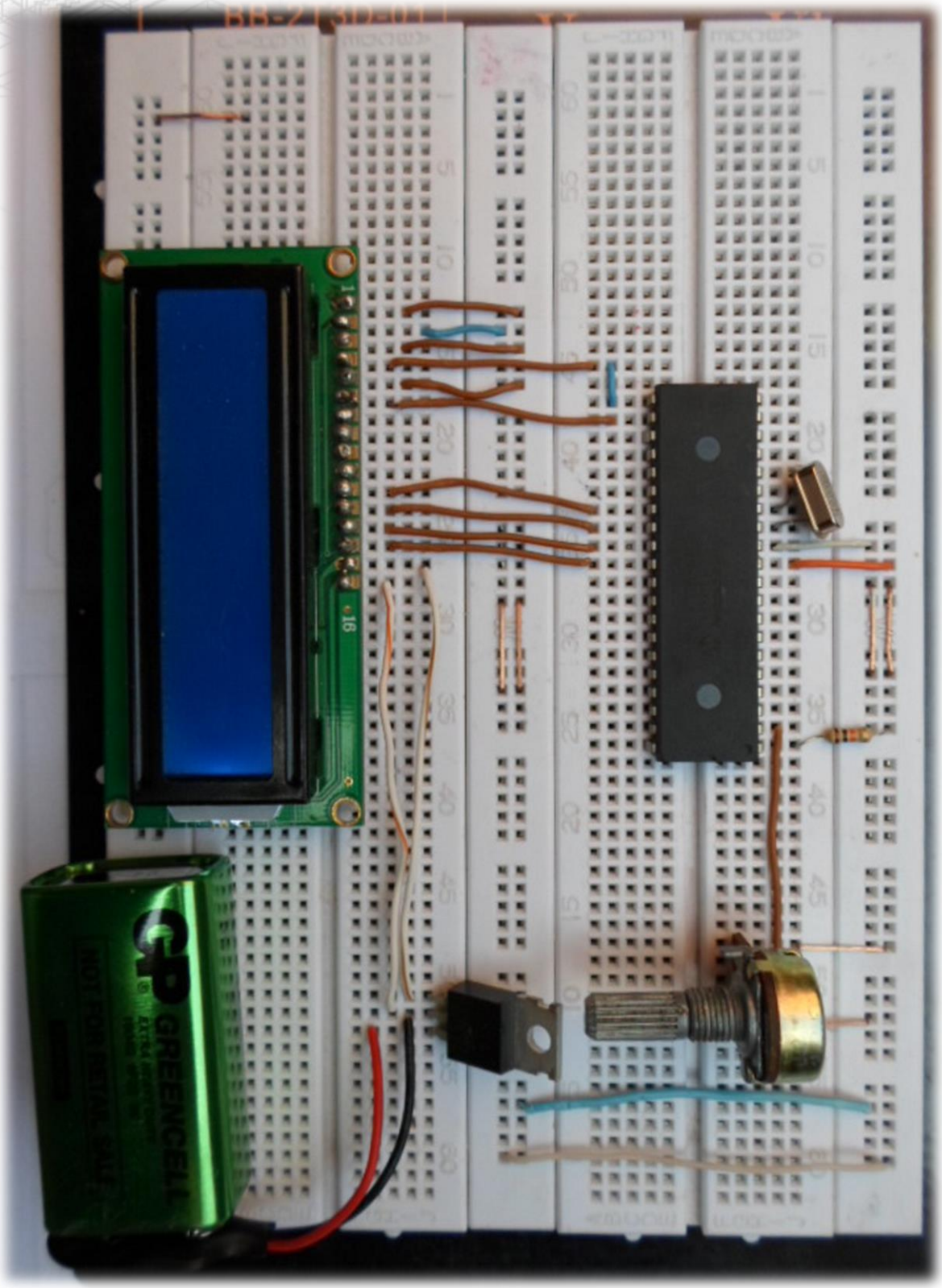




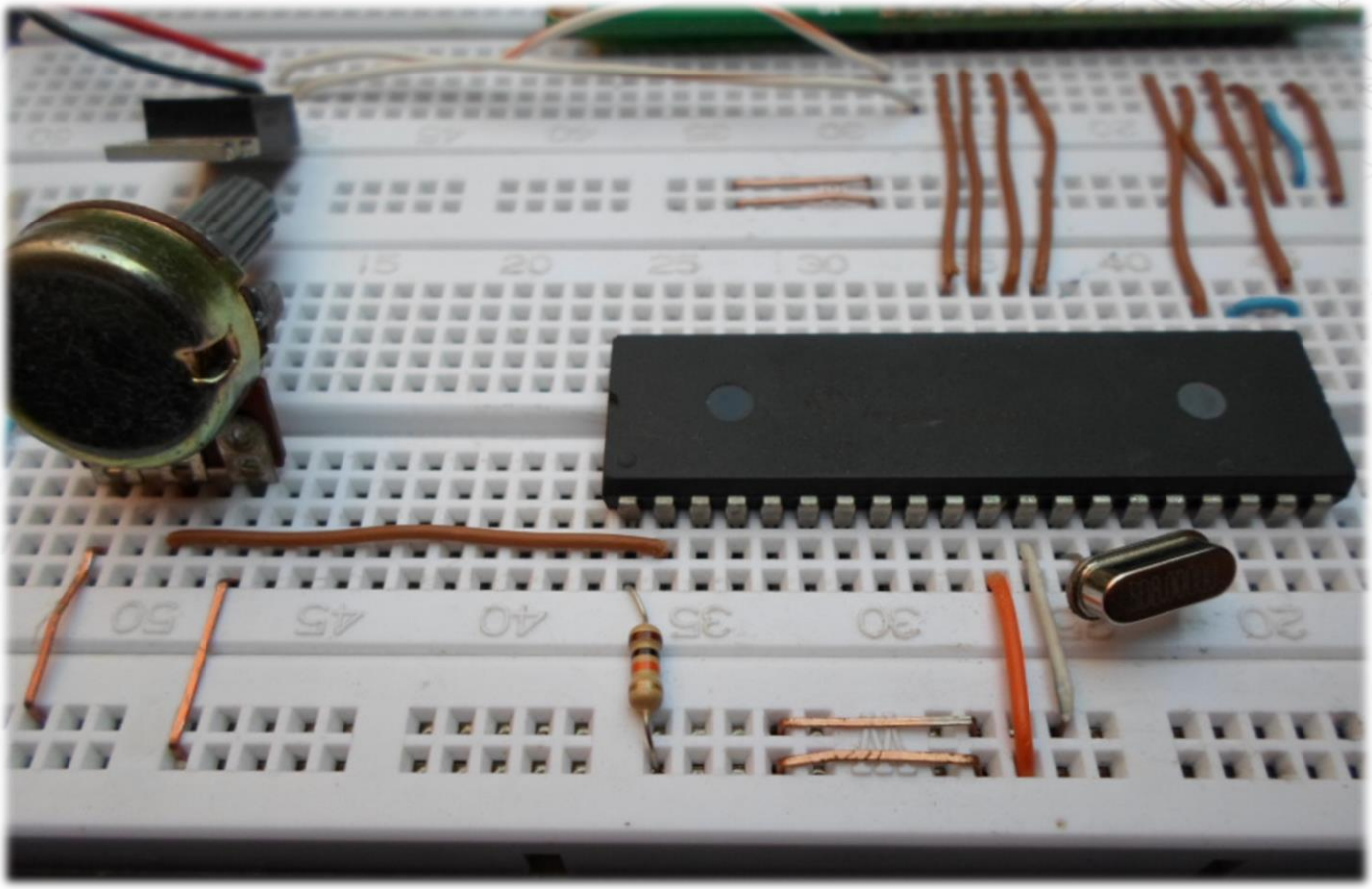




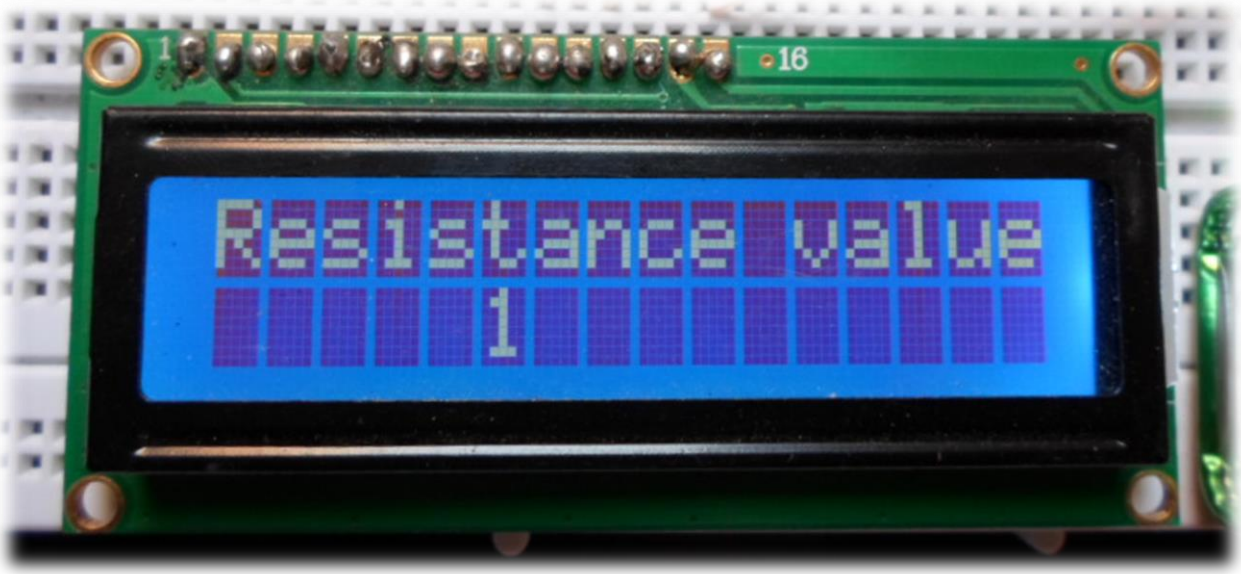
الهاردوير



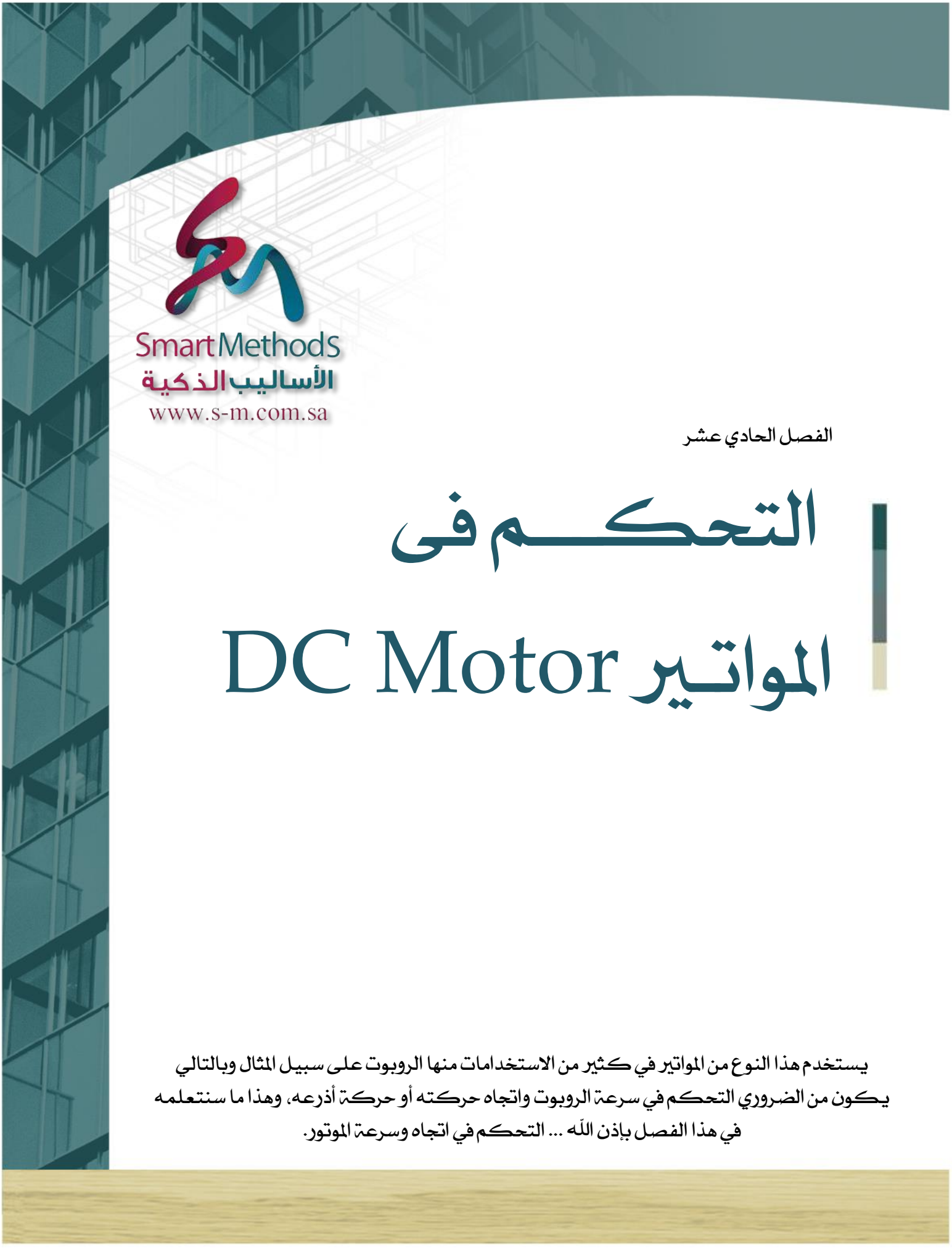
وهنا في الهاردوير قمت بتوصيل مقاومة واحدة وهي المقاومة المتغيرة، وفيما يلي نظرة عن قرب لطريقة التوصيل:



لقطات من القراءات



ملحوظة أخيرة: علمنا أن الميكرو 16F877A يحتوي على ثمانية رجول لقراءة الإشارات الأنالوج فهل هذا معناه أنه يحتوي داخله على ثمانية من الـ ADC Interface؟؟؟ بالطبع لا وهذا يمكن استنتاجه من طريقة كتابة الدالة، فدالة القراءة يتم تحديد لها أي الأرجل سيتم القراءة منها وبالتالي فهو ADC Interface واحد يبدل عملية القراءة بين الأرجل.



الفصل الحادي عشر

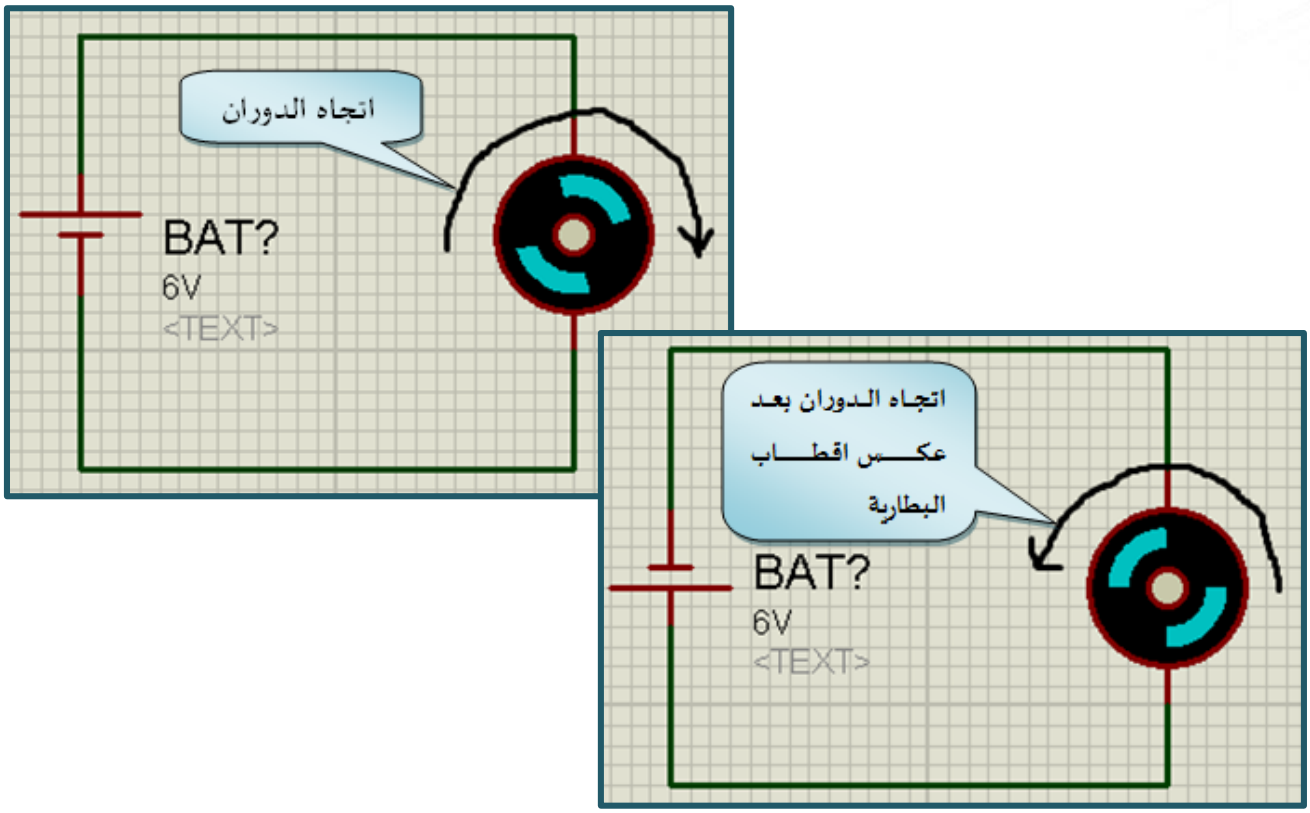
التحكم فى المواتير DC Motor

يستخدم هذا النوع من المواتير في كثير من الاستخدامات منها الروبوت على سبيل المثال وبالتالي يكون من الضروري التحكم في سرعة الروبوت واتجاه حركته أو حركة أذرعه، وهذا ما سنتعلمه في هذا الفصل بإذن اللّه ... التحكم في اتجاه وسرعة الموتور.

ملحوظة: يلزم عند التعامل مع موتور معين أن نكون على علم بما يحتاجه من جهد وتيار ليعمل وذلك لاختيار الـ power supply المناسب لتشغيله وحتى لا يحدث أي تلف في الدائرة.

التحكم في اتجاه الدوران

يمكنك تغيير اتجاه دوران الموتور إذا عكست اتجاه التيار الذي يمر من خلاله وذلك بعكس أقطاب البطارية الموصلة عليها وذلك كما بالشكلين الآتيين:

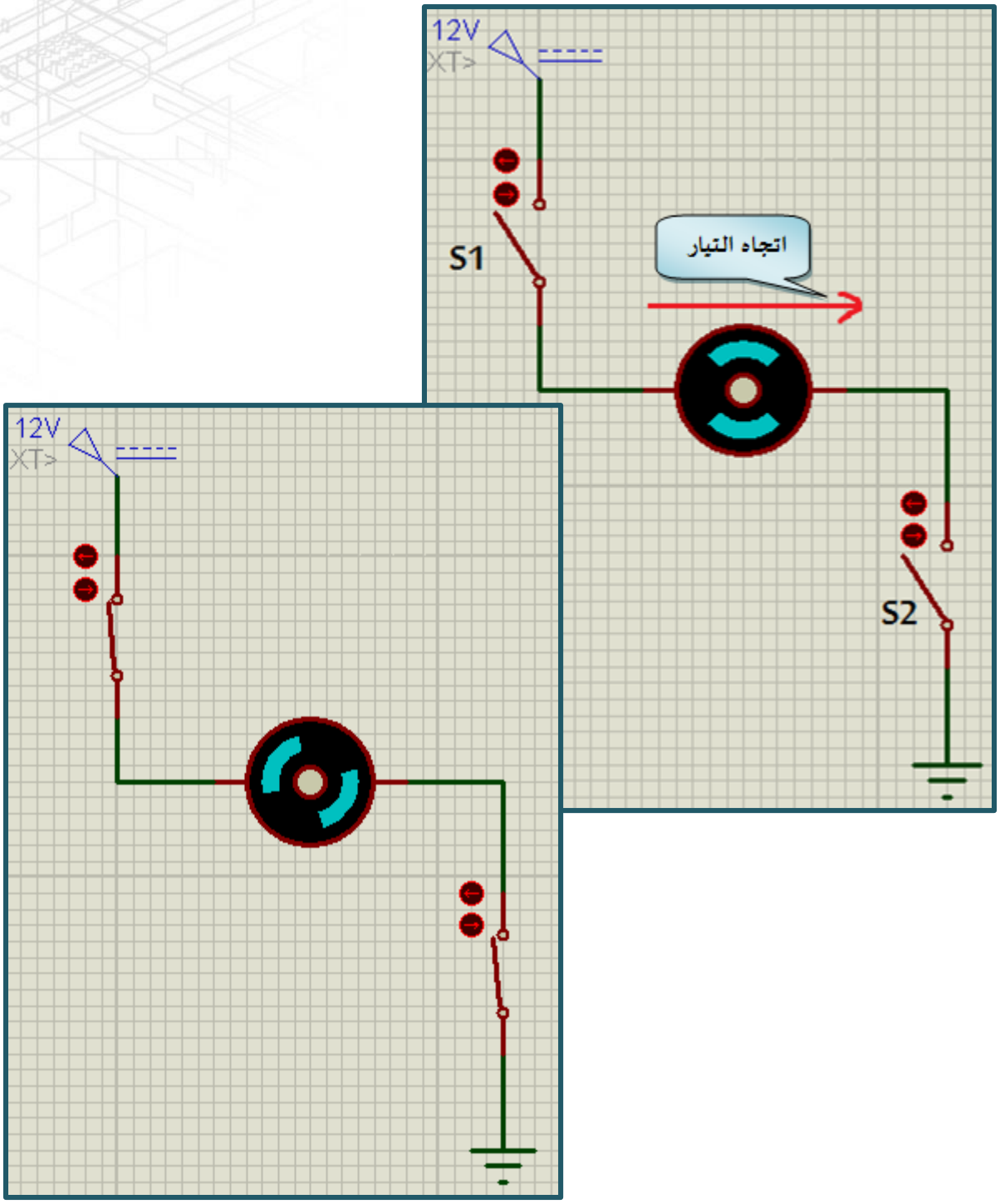


ولو أردنا أن نفعل هذا بالطريقة التقليدية اليدوية فستقابلنا عيوب كثيرة منها مثلا أننا ببساطة سنضطر إلى إيقاف الدائرة حتى نتمكن من عكس أقطاب البطارية، وهذا بالطبع غير مرغوب فيه ... اذا ما الحل ؟؟؟

H-Bridge

يكمن الحل في دائرة شهيرة تسمي H-bridge والتي سنتعلمها تدريجيا ...

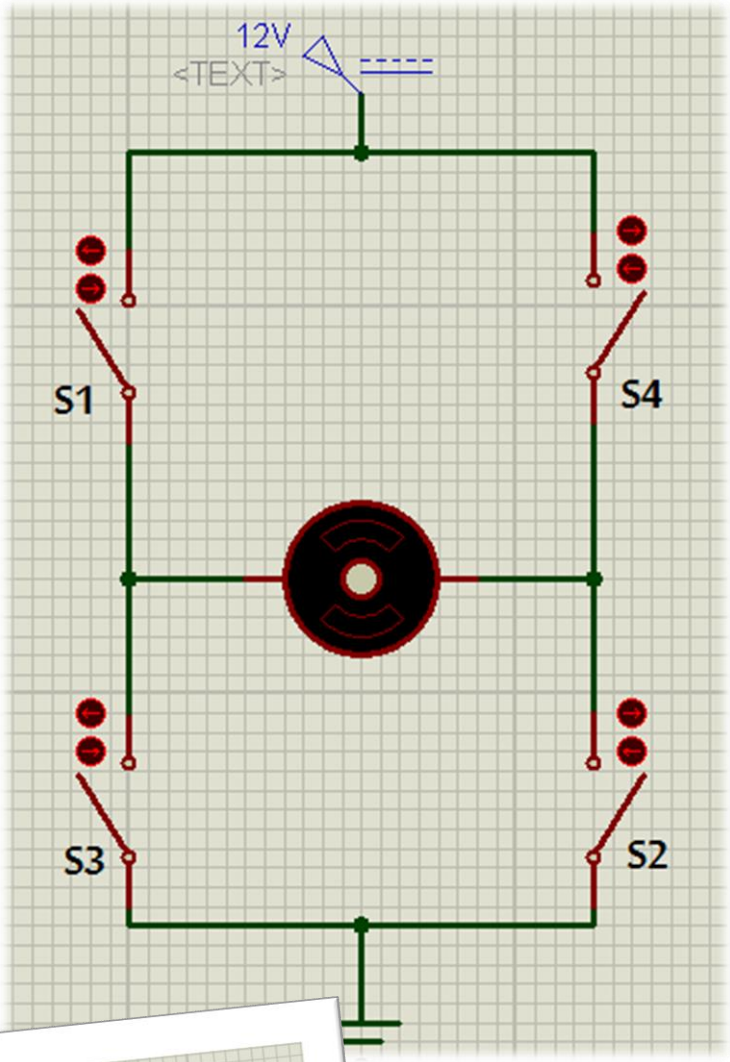
لنفرض أننا سنوصل الدائرة بالشكل الآتي:



سنجد في الشكل السفلي أنه لكي يدور الموتور لابد أن نغلق المفتاحين S1، S2 معا وإلا فلن يدور الموتور، مع الأخذ في الاعتبار اتجاه التيار لأنه هو المتحكم في اتجاه دوران الموتور.

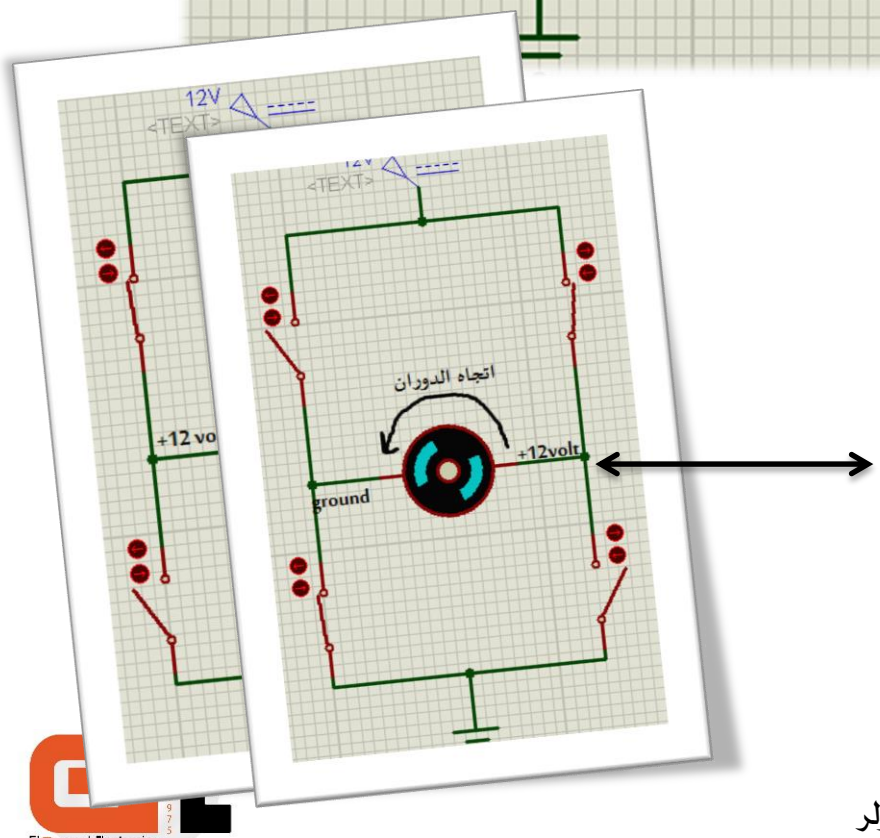
ولكننا حتى الآن لم نستطع أن نجعل الموتور يدور في اتجاهين مختلفين، ولكن إذا أضفنا على الدائرة السابقة جزء آخر مماثل لتصبح كما في الشكل التالي:

بالنظر إلى هذه الدائرة سنجد ما يلي:

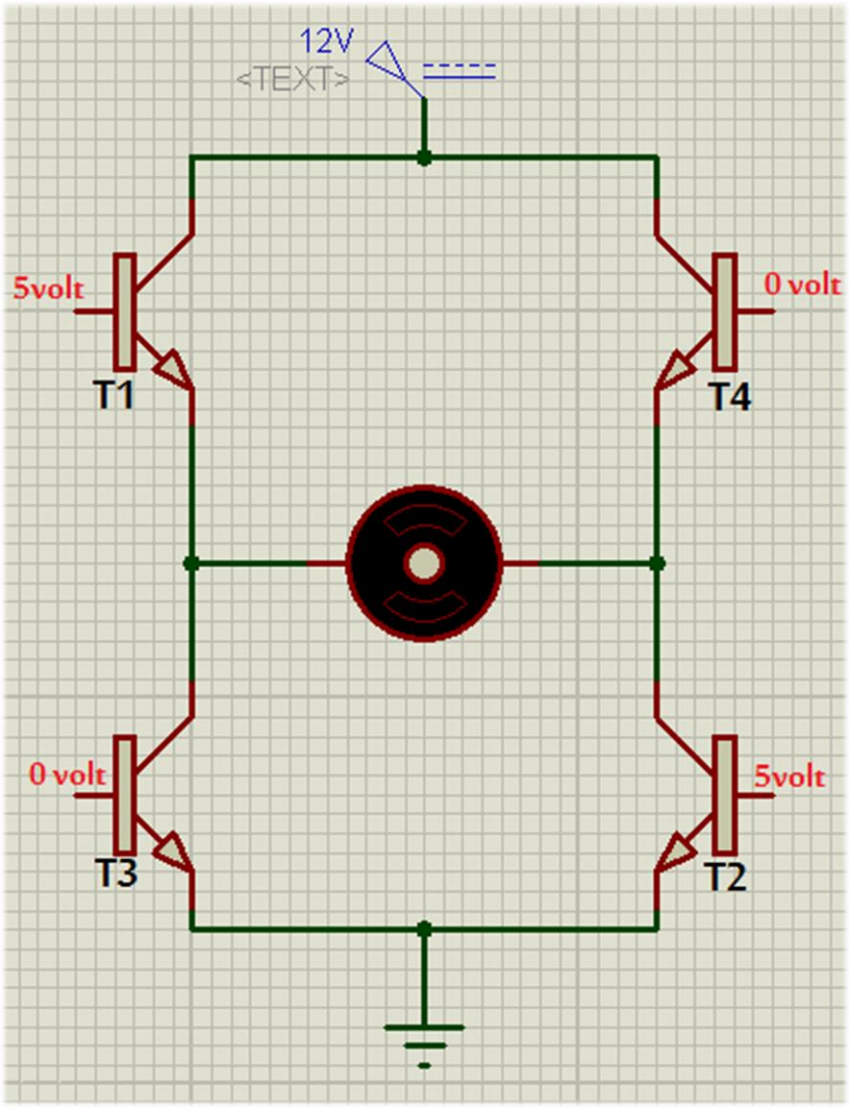


اذا تم إغلاق السويتشين S1، S2 والإبقاء على السويتشين S3، S4 مفتوحين سيتم توصيل الطرف اليسار للموتور بالطرف الموجب للجهد ١٢ فولت وسيصبح الطرف اليمين للموتور متصل بالأرضي، وهما ما يترتب عليهما دوران الموتور في اتجاه معين.

أما في الحالة العكسية أي عند إغلاق السويتشين S3، S4 والإبقاء على السويتشين S1، S2 مفتوحين سيكون الطرف اليسار للموتور متصل بالأرضي والطرف اليمين متصل بموجب الجهد ١٢ فولت، أي أنه تم عكس الأقطاب وبالتالي سيدور الموتور في الاتجاه المعاكس.



وبهذا نكون قد فهمنا الدائرة السابقة والتي من خلالها نستطيع التحكم في اتجاه الموتور وباستخدام بطارية واحدة فقط، إلا أنه بقي بها مشكلة بسيطة وهى أنها تستلزم أن نقوم نحن بالضغط على السويتشات يدويا، مما يجعل عملية التحكم بطيئة وتستلزم تدخل المستخدم وهذا في حد ذاته مشكلة فالمطلوب هو التحكم إلكترونيا عن طريق الميكروكنترولر ... والحل لهذه المشكلة البسيطة يمكن استنباطه من أسلوب تم شرحه سابقا، فحيث أن الميكروكنترولر يخرج جهد إما خمسة وإما صفر فولت فبالتالي يمكن استبدال السويتشات بدائرة أخرى تؤدي نفس وظيفته أي عند دخول الخمسة فولت من الميكرو تقوم بإغلاق السويتش وتوصل الطرفين وعند وضع صفر فولت تكون Open Circuit ... وهو ما ينطبق على دائرة الـ transistor as a switch كما سبق شرحه في الفصول الماضية، وبناء عليه نستبدل الأربعة سويتشات بأربعة ترانزستورات لتصبح الدائرة كما بالشكل:

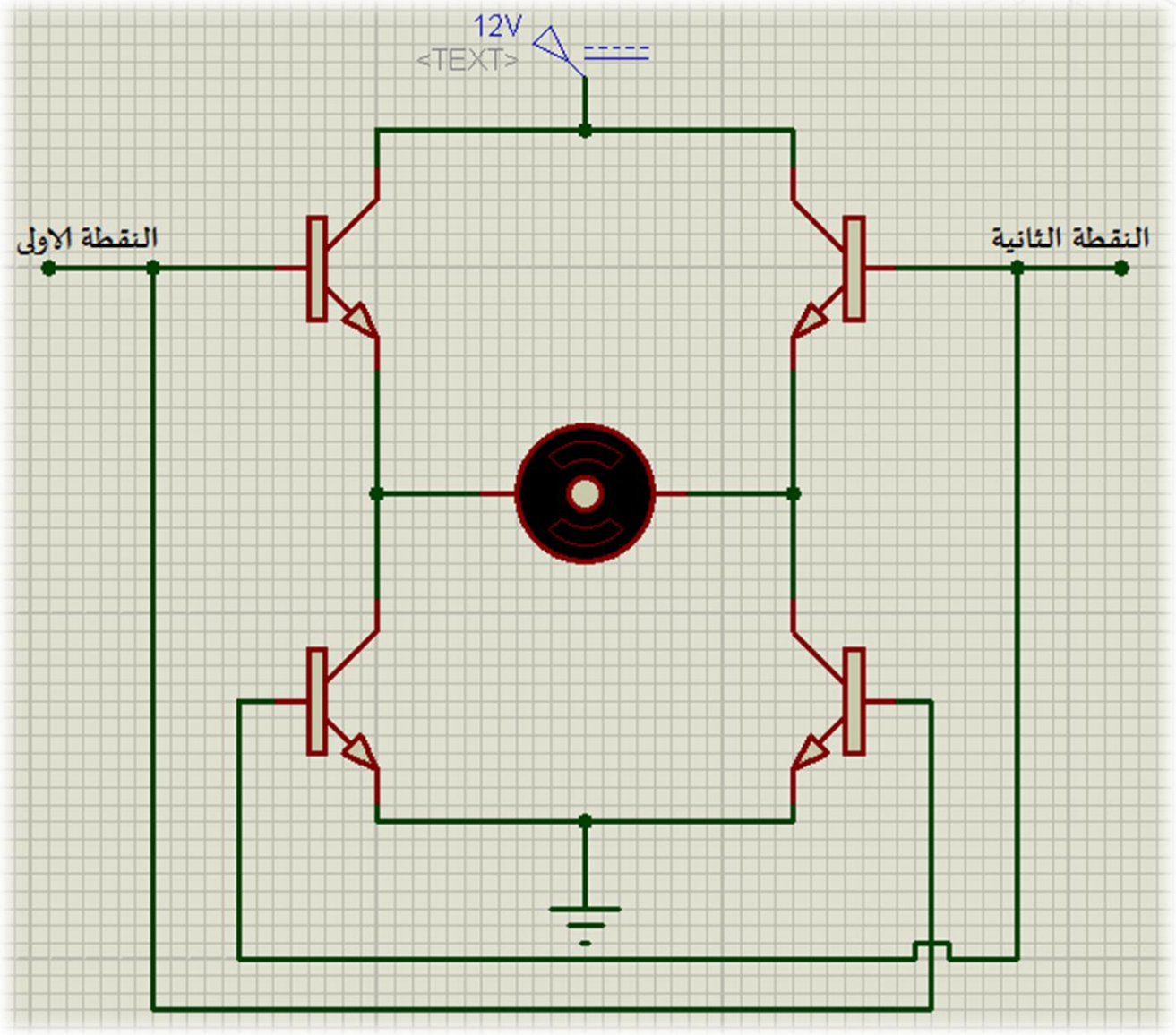


فاذا أدخلنا خمسة فولت على T1، T2 وصفر فولت على T3، T4 فإن الموتور سيدور في اتجاه معين.

والعكس بالعكس فاذا أدخلنا صفر فولت على T1، T2 وخمسة فولت على T3، T4 فإن الموتور سيدور في الاتجاه المعاكس.

مثال: هل فهمت الآن لماذا سميت هذه الدائرة H-bridge ؟؟؟ انظر إلى الشكل المجاور إذن وتجا هل الجهد والأرضي وسيمكنك استنباط ذلك بكل سهولة من رسم حرف H فيها ☺.

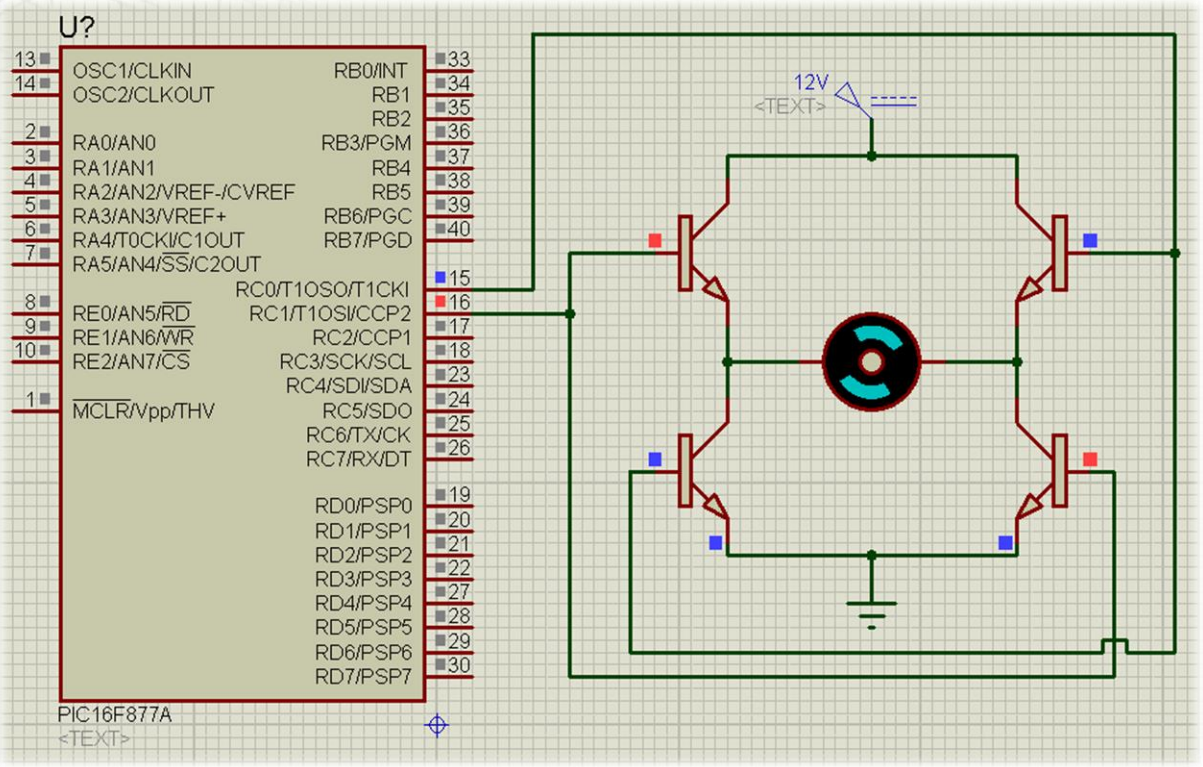
وربما لاحظت من الشرح السابق أن T1، T2 يتصل عليهم دائما نفس القيمة، فلو وصلت خمسة فولت على T1 لابد أن يكون T2 متصل أيضا بخمسة فولت، وكذلك الحال عن توصيل صفر فولت، والمثل بالمثل بالنسبة لـ T3، T4، وبالتالي من المنطقي اختصار كل طرفين متشابهين في طرف واحد لتقليل أطراف الدائرة وذلك كما في الشكل الآتي:



وبالتالي لكى يدور الموتور في الاتجاه الأول نضع خمسة فولت على النقطة الأولى وصفر فولت على الثانية والعكس عندما نريده أن يدور في الاتجاه الأخر، وبالطبع لا داعي للتنويه أن هذه الخمسة فولت يمكنا الحصول عليها من خرج الميكروكنترولر، وبالتالي استطعنا التحكم في اتجاه الموتور من خلال الميكرو.

مثال تطبيقي

بافتراض توصيل النقطة الأولى على RC0 والنقطة الثانية على RC1، فستكون شكل الدائرة على بروتس كما يلي:



وبافتراض أيضا أننا نريد مثلا أن يدور الموتور في الاتجاه الأول لمدة ٣ ثواني ثم يدور في الاتجاه المعاكس لثلاث ثواني أخرى، وبالتالي سيكون البرنامج كما هو موضح:

```
void main()
{
    TRISC.B0 = 0;
    TRISC.B1 = 0;

    while(1)
    {
      PORTC.B0 = 1;
      PORTC.B1 = 0;
       delay_ms(3000);
```

لضبط RC0, RC1 على ان يعملوا كخرج

الامرين الموضحين يتم تنفيذهم بسرعة كبيرة جدا فيبدوا وكأنهم ينفذان فى نفس اللحظة وهذا ما نريده وهما يجعلوا الموتور يدور فى الاتجاه الاول

```
        PORTC.B0 = 0;
        PORTC.B1 = 1;
          delay_ms(3000);
    }
}
```

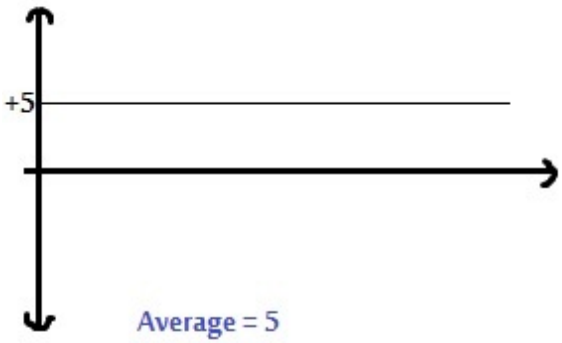
لجعل الموتور يدور فى الاتجاه الاخر

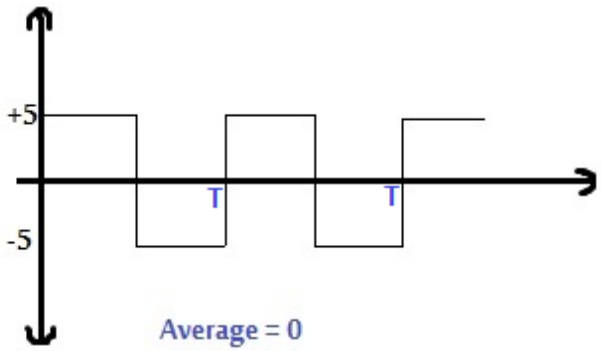
وبالطبع يمكن تعديل عمل البرنامج الماضي بإضافة سويتشين كدخل للميكرو بحيث يدور الموتور في اتجاه معين عند الضغط على السويتش الأول ويدور في الاتجاه المعاكس عن الضغط على السويتش الآخر.

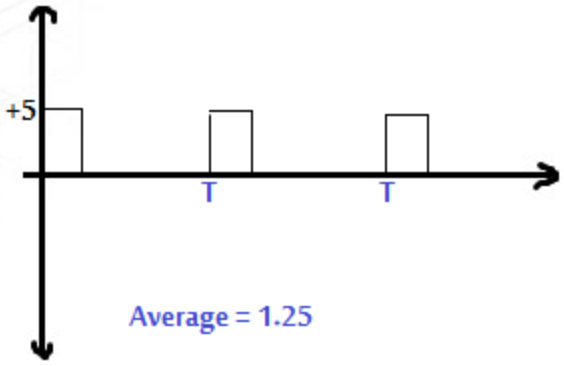
التحكم في سرعة الدوران

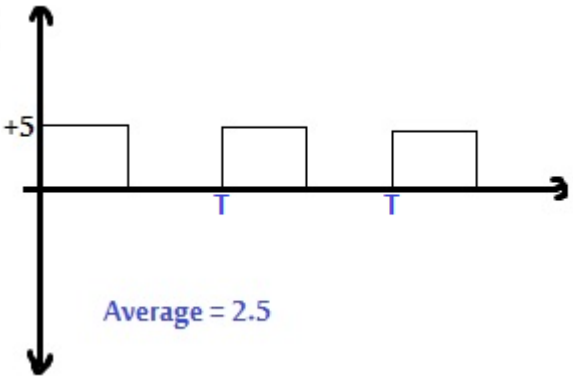
أبسط طريقة للتحكم في سرعة الموتور تكون بتغيير قيمة الجهد المطبق عليه، فلو فرضنا مثلا موتور يعمل على جهد ١٢ فولت، فعند توصيله ببطارية ١٢ فولت سيدور بالسرعة المحددة الطبيعية له، ولكن عند تقليل الجهد إلى ٦ فولت ستقل سرعته للنصف مثلا، ولو قللنا الجهد إلى ٣ فولت ستقل سرعته ... وهكذا، لكن كيف يمكن لنا أن نغير قيمة الجهد التي يتم تطبيقها على الموتور؟؟ الإجابة البديهية هي شراء عدد كبير من البطاريات المختلفة الجهد وربما نحتاج لتوصيل بطاريتين أو ثلاثة لتكوين قيمة معينة وأيضا لن نحصل على كل القيم بل سيكون الفرق بين كل قيمة والمجاورة لها واحد فولت أو على الأقل نصف الفولت ؟؟ فهل هناك ما يفيد في تغيير الفولت باستخدام بطارية واحدة فقط؟؟ ... لنتعرف كيف ذلك ولكن بطريقة تدريجية ...

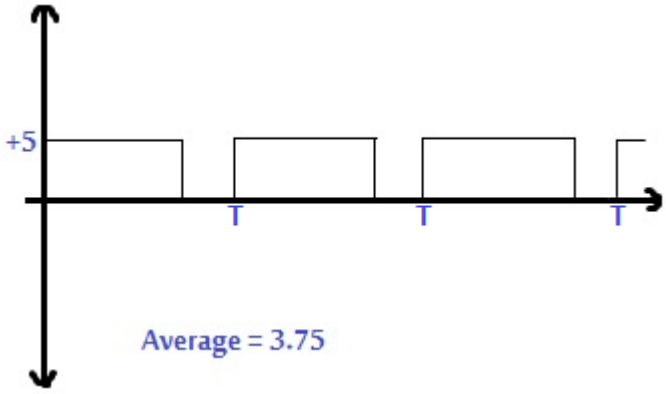
نوع الموتور الذي نتناوله بالتحكم في هذا الفصل يسمي DC Motor، ولكن ليس المقصود بكلمة DC الجهد الثابت للقيمة وإنما المقصود بها هي قيمة الجهد المتوسط للإشارة وفيما يلي بعض الأمثلة التي نفهم منها معنى القيمة المتوسطة Average Value:









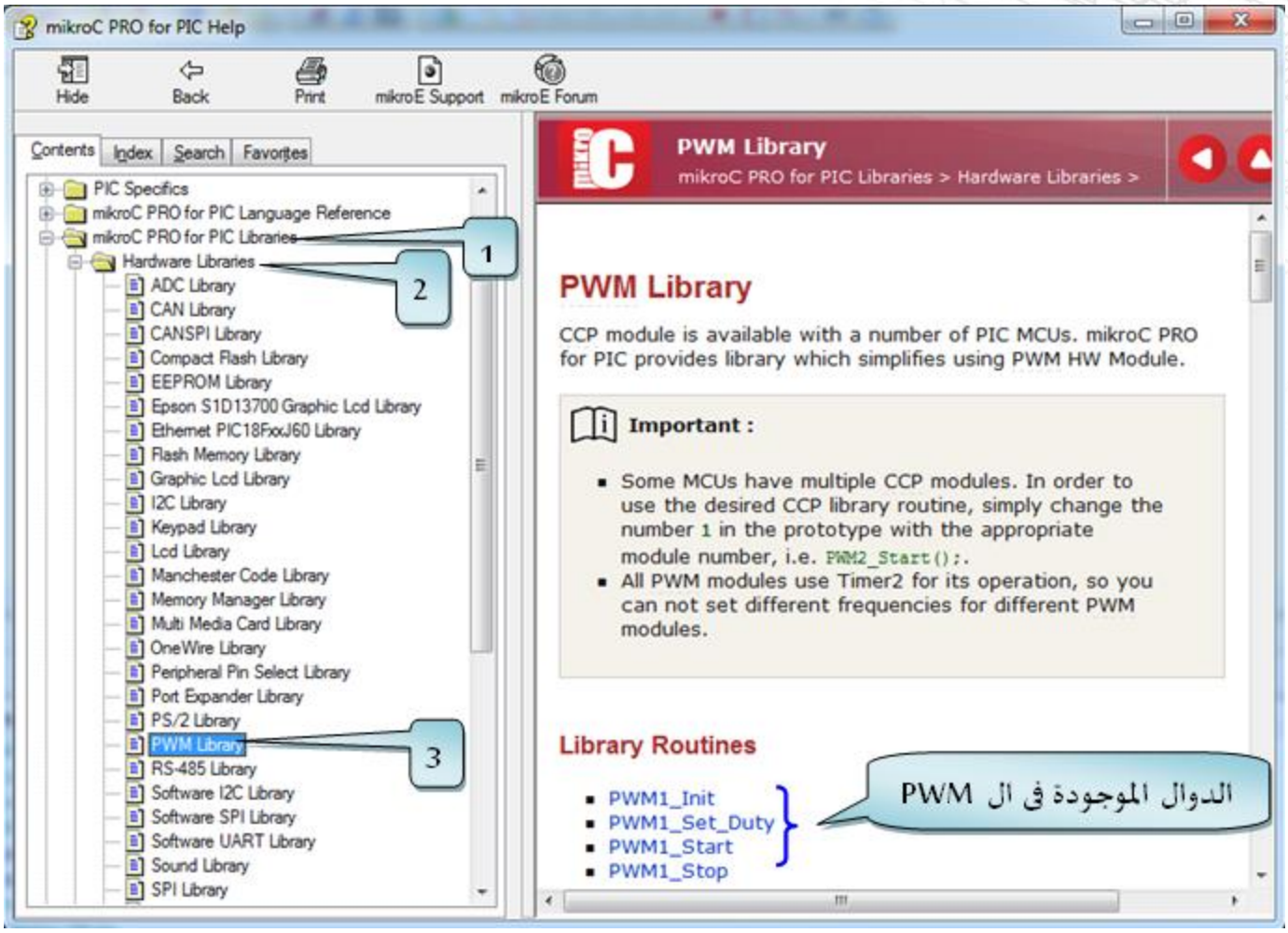


إذا استخدمنا موتور يعمل على ٥ فولت وأدخلنا الإشارة التي قيمتها المتوسطة ٥ فولت فإن الموتور سيدور بسرعته المحددة له، وبتقليل هذه القيمة المتوسطة تقل سرعة دوران الموتور إلى أن تتوقف تماما عند الصفر.

وبالتالي إذا استطعنا الحصول على إشارات مثل السابقة يمكن التحكم في قيمتها المتوسطة من الميكروكنترولر فأننا نستطيع التحكم في سرعة الموتور ... وللحصول على مثل هذه الإشارات يستخدم موديول يسمى Pulse Width Modulation Module أو اختصارا PWM، وإذا قمت بترجمة المصطلح PWM لوجدت أن كلمة Modulation تعني تعديل وكلمة pulse width تعني عرض النبضة وبالتالي تكون الجملة كاملة تعني عمل تعديل وتغيير في عرض النبضة للحصول على جهود ذات قيمة مختلفة وهو ما ظهر في الرسومات البيانية السابقة.

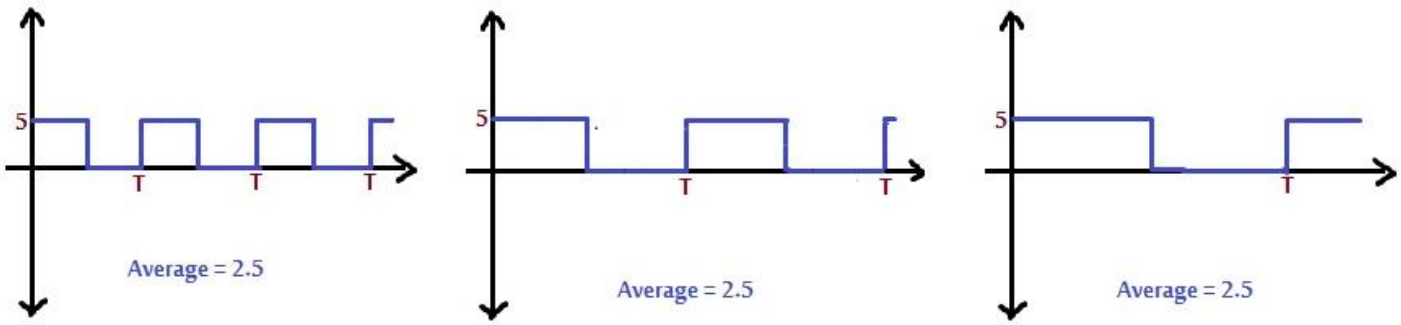
دوال الميكروسي

وكما تعودنا فإنه يمكننا الحصول على الدوال التي تتعامل مع هذا الموديول من خلال مكتبات الهاردوير في نافذة المساعدة ثم البحث عن مكتبة الـPWM Modulation أسفلها كما بالشكل التالي:

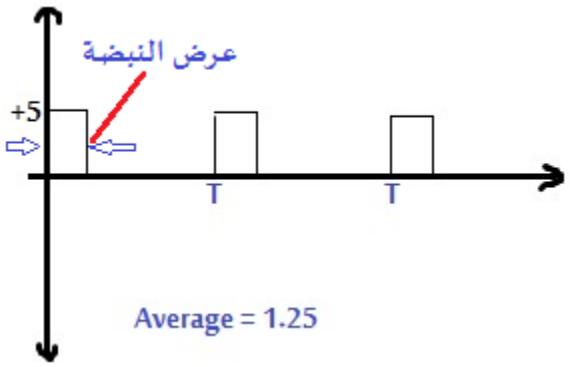


ولكي نفهم الدوال علينا أولا من معرفة الخصائص التي نحتاج لتخصيصها في الإشارة ذات النبض المتغير العرض التي تخرج من الميكرو ... بعض الخصائص قد يؤثر في عمل الموتور ..

أولا: يلزم تحديد تردد الإشارة وهذه خاصية هامة جدا في المواتير ويتم تحديدها عن طريق التجربة حيث تقوم بتجربة عدد مختلف من الترددات على الموتور حتى يتبين لك على أي تردد يعمل، وفي هذه الأشكال تتضح خاصية التردد حيث أن القيمة المتوسطة للخرج متساوية ولكن الفرق في التردد :



ثانيا: يلزم تحديد عرض النبضة الخارجة، والصورة الأتية توضح المقصود بعرض النبضة حيث أنها تمثل الجزء الـ High من الإشارة ...



الدالة الأولى

```
PWM1_Init(5000);
```

تستخدم لضبط تردد الإشارة الخارجة، حيث يكتب التردد بالهرتز بين الأقواس، ففي الأمر السابق كان التردد ٥٠٠٠ هرتز، وتكتب هذه الدالة داخل الدالة الرئيسية ...

ربما لا يمكنك ملاحظة التغير في سرعة الموتور وعندها قم بتغيير قيمة التردد بالزيادة أو النقصان حتى يمكنك ملاحظة التغير في السرعة.

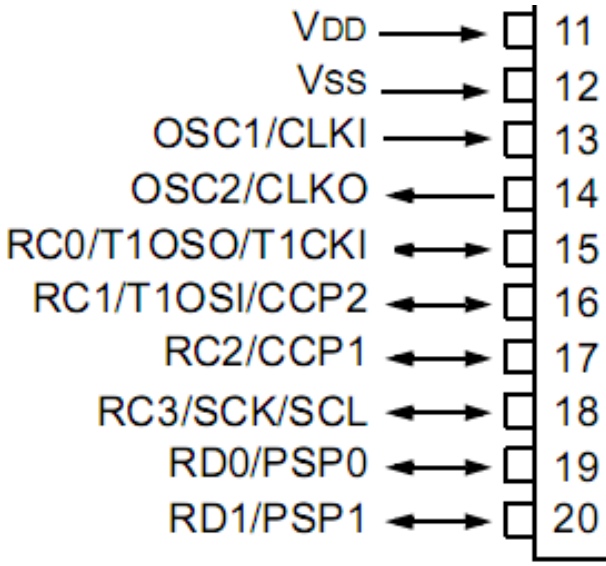
الدالة الثانية

```
PWM1_Set_Duty(192);
```

تستخدم لتحديد عرض النبضة وهى تأخذ قيمة من صفر إلى ٢٥٥، حيث أن القيمة صفر تعنى أن عرض النبضة يساوى صفر أي نستطيع اعتبار أن الإشارة صفر فولت لا نه لا يوجد أي جزء من الإشارة في الحالة high، والقيمة ٢٥٥ تعنى أن النبضة كلها high مما يعنى أن الإشارة – تقريبا – تعتبر DC، وبالتالي القيمة ١٢٨ تعنى أن نصف الزمن تكون الإشارة فيه High، وأيضا القيمة ١٩٢ (ثلاثة أرباع القيمة ٢٥٥) تعني أن ثلاثة أرباع الزمن الدوري تكون فيه الإشارة High، وأخيرا القيمة ٦٤ (ربع القيمة ٢٥٥) تعنى أن ربع الزمن الدوري تكون فيه الإشارة High.

الدالة الثالثة

باستخدام الدالتين الأولى والثانية قمنا بضبط الإعدادات، ثم يأتي الدور على الدالة الثالثة لتقوم بإخراج الإشارة على أحد رجول الميكرو المخصصة لهذا المودويول، وهذا الرجل يكون مكتوب بجوارها CCP1 في الداتا شيت، وهي تمثل الرجل رقم ١٧ في الميكروكنترولر 16F877A كما بالشكل:



وربما لاحظ أيضا أن الرجل رقم ١٦ مكتوب عليها CCP2 أيضا، وهذا يعنى أن هذا الميكرو يحتوي على عدد موديولين وليس موديول واحد فقط ...

وتأخذ الدالة الشكل التالي:

```
PWM1_Start();
```

لاحظ وجود الرقم ١ في اسم الدالة PWM1 وهذا يعني أنا الدلة ستعمل على المودويل الأول أي الرجل رقم ١٧، وإذا أردنا العمل على الموديول الثاني أي الرجل ١٦ سنستبدل الرقم ١ بالرقم ٢ في اسم الدالة السابقة.

الدالة الرابعة

وتقوم بإيقاف إخراج الإشارة، وهي مهمة جدا في بعض الاستخدامات حيث يمكن استخدامها في إيقاف الموتور مثلا كما سيتبين فيما بعد، وتأخذ الدالة الشكل التالي مع مراعاة الرقم كسابقتها:

```
PWM1_Stop();
```

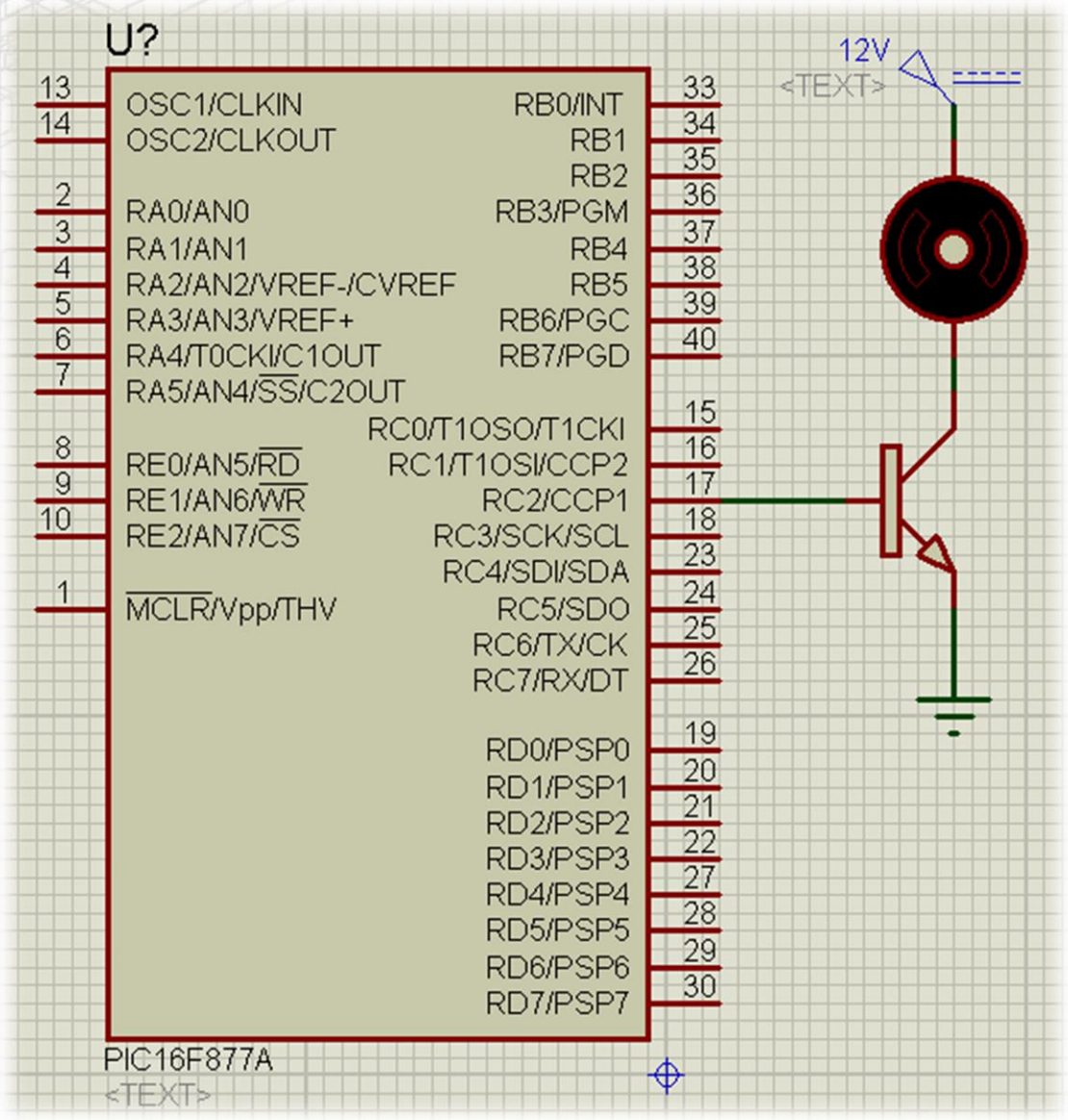
والآن بعد ما عرفناه من دوال يمكننا التحكم في سرعة الموتور بسهولة جدا وذلك بالتحكم في قيمة التردد وعرض النبضة ثم تشغيل الموديول ...

على مستوى الهاردوير

ولكن يتبقى مشكلة بسيطة وهي أن الإشارة التي سيخرجها الميكرو أقصى قيمة لها خمسة فولت كما أكدنا مرارا في السابق، وبالتالي يمكن للميكرو أن يخرج ٥ أو ٢,٥ أو ١,٢٥ فولت مثلا أو غير ذلك من القيم فيما لا يتعدى الخمسة فولت، ولكن الموتور يعمل على جهد عالي كما نعلم وبالتالي لا يمكن توصيله مباشرة بالميكرو، فما الحل ؟؟؟ يمكنك استنتاجه بسهولة من الفصول السابقة.

يتمثل الحل في استخدام الدائرة التي سبق وشرحناها والتي تسمى Transistor as a switch فندخل الإشارة الخارجة من الميكرو على قاعدة الترانزستور فيقوم الترانزستور بتوصيل أو قطع دائرة الجهد العالي الموصل عليها الموتور حسب قيمة الخرج إما خمسة فولت أو صفر.

ويتضح شكل الدائرة في الرسم التالي:



مشروع تطبيقي

فكرة المشروع

الآن نريد عمل مشروع للتحكم في سرعة الموتور عن طريق مقاومة متغيرة، بحيث عندما يتم تغير قيمة هذه المقاومة تتغير سرعة الموتور ...

طريقة العمل

نقوم بتوصيل المقاومة المتغيرة على مصدر جهد خمسة فولت للحصول على جهد متغير منها يتراوح بين صفر وخمسة فولت، وعندما تتغير قيمتها يقوم الميكرو بقراءة الجهد الناتج عنها.

وبناء على قيمة الجهد المقروء يقوم الميكرو بحساب عرض النبضة، ثم يقوم الميكرو بضبط الإعدادات وإخراج الإشارة.

برنامج الميكروسي

الموديولات التي سنحتاجها هي الـ PWM وكذلك الـ ADC، وبالتالي نكتب داخل الدالة الرئيسية دوال التهيئة الخاصة بهما كالآتي ...

```
PWM1_Init(1000);
ADC_Init()
```

ويكون البرنامج الكلى كما بالشكل:

```
int duty;
int V;
void main()
{
    PWM1_Init(1000);
    ADC_Init();

    while(1)
    {
      V = ADC_Read(0);
      V = (V *5)/1023 ;

      duty = (V*255)/5;
      PWM1_Set_Duty(duty);

      PWM1_Start();

    }
}
```

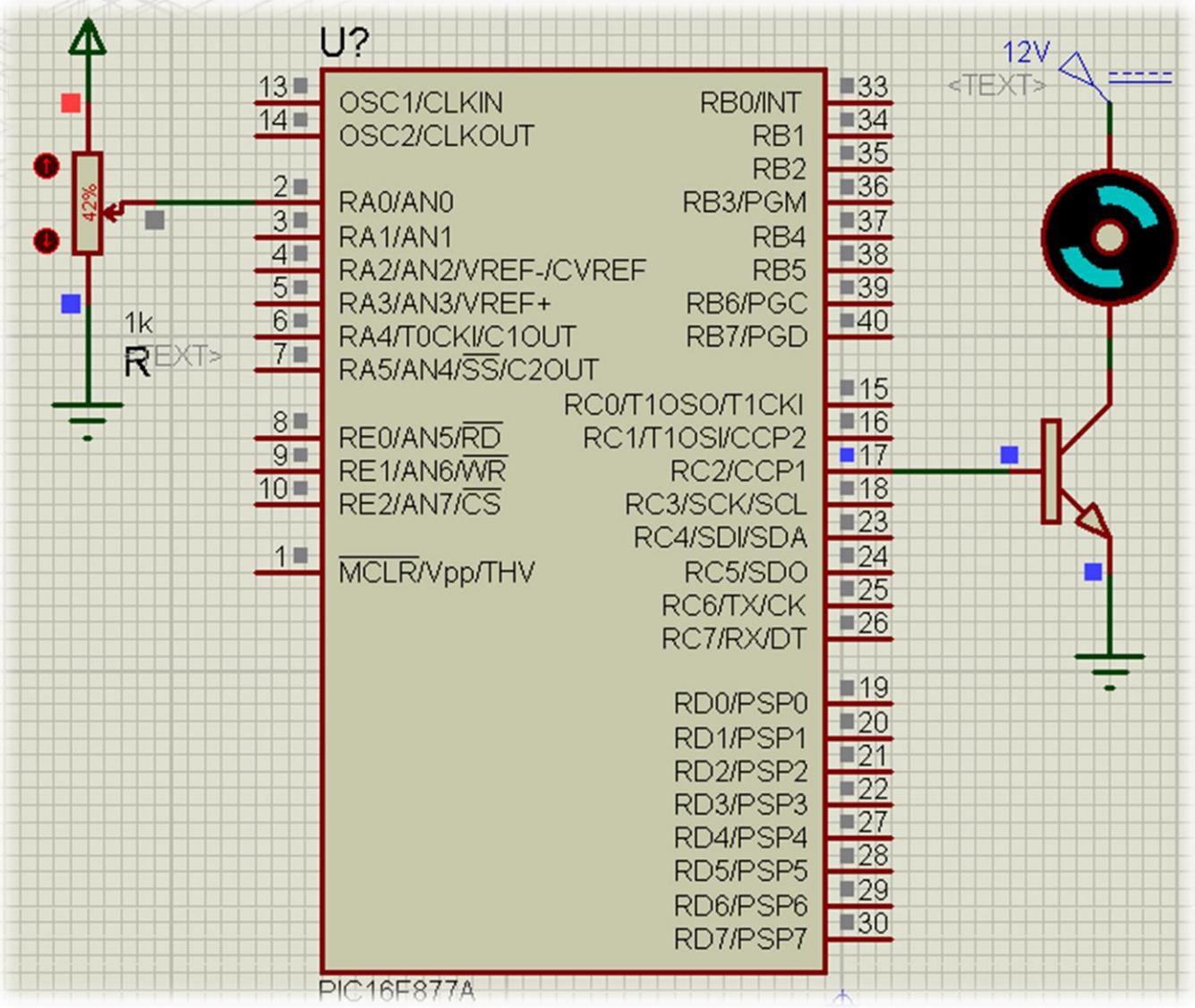
لقراءة قيمة الاشارة الاناولج والتى تتراوح قيمتها من صفر الى 1023

لتحويل القيمة الى قيمة الجهد الفعلى (من 0 الى 5 فولت)

لتحويل قيمة الجهد (من 0 الى 5) الى قيمة تحدد عرض النبضة والتى تتراوح من 0 الى 255 ووضعها فى الدالة

المحاكاة

نقوم بتوصيل المقاومة المتغيرة التي تخرج جهد متغير على أحد رجول الأنالوج وليكن الرجل AN0 وهي الرجل رقم ٢ أو الرجل ذات الاسم RA0، ونقوم بتوصيل دائرة الموتور على أحد رجلي الـ PWM وليكن الرجل CCP1 رقم ١٧ كما بالشكل التالي:



في الشكل السابق قمنا بتشغيل المحاكاة ثم قمنا بتغيير قيمة المقاومة بالضغط على أحد الزرين الموجودين بجوار المقاومة فوجدنا تغير في سرعة الموتور كما سبق ...

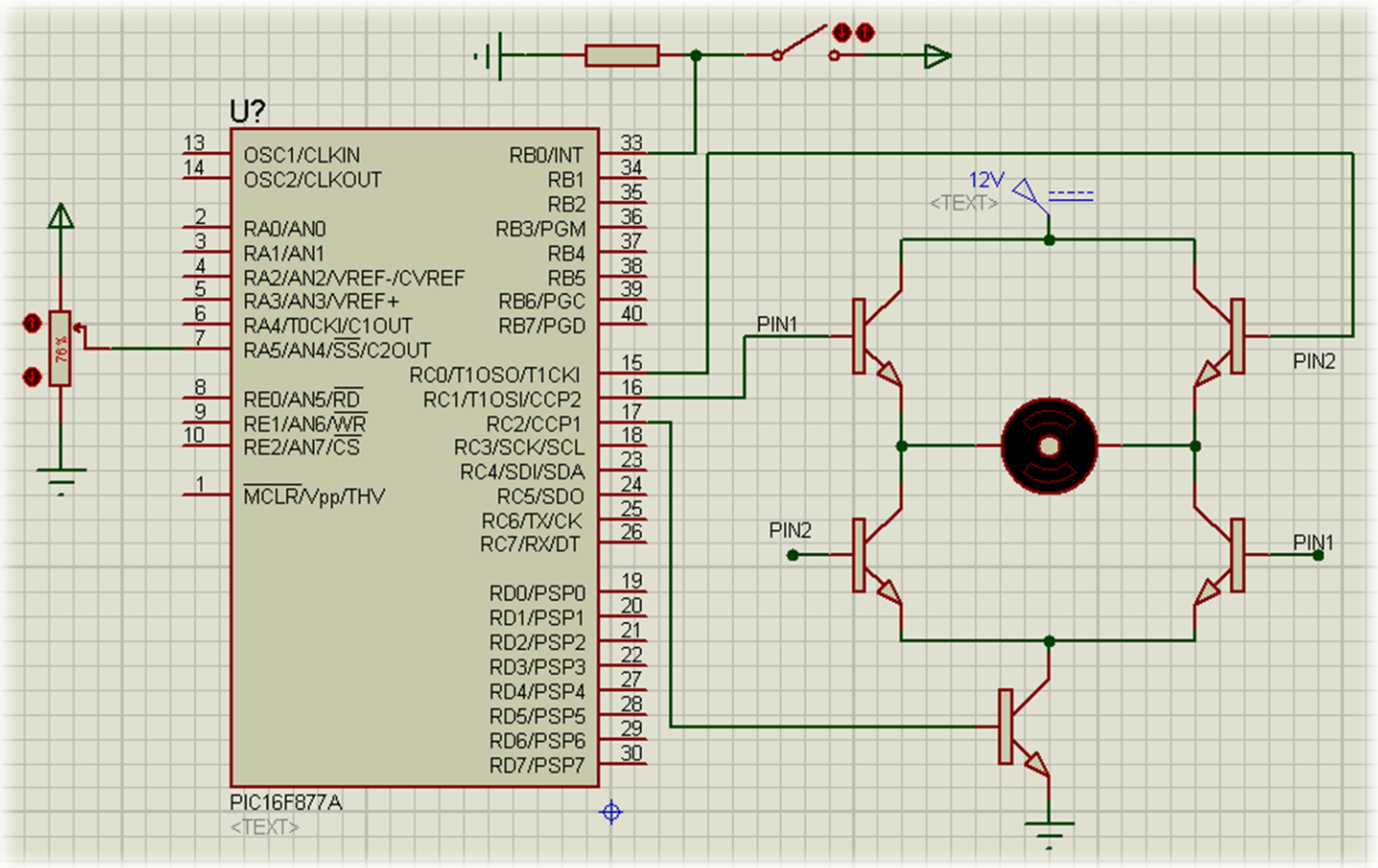
مشروع متكامل

حتى هنا تعلمنا كيف نتحكم في اتجاه حركة الموتور وتعلمنا أيضا كيف نقوم بالتحكم في سرعته كل على حدة، ونريد الآن أن نتعلم كيف نقوم بالتحكم في السرعة والاتجاه معا في مشروع واحد ... كيف ذلك ؟؟؟

بالطبع سيكون ذلك بدمج الدائرة الخاصة بالتحكم في السرعة بالدائرة الأخرى الخاصة بالتحكم في الاتجاه، وهنا يجب عليك أن تحاول أن تتخيل شكل الدائرة النهائية بعد دمج الدائرتين ... حاول جديا قبل أن تنظر للإجابة فيما يلي ...

الدائرة

فيما يلي إجابة السؤال السابق حيث تصبح الدائرة الكلية كالآتي:

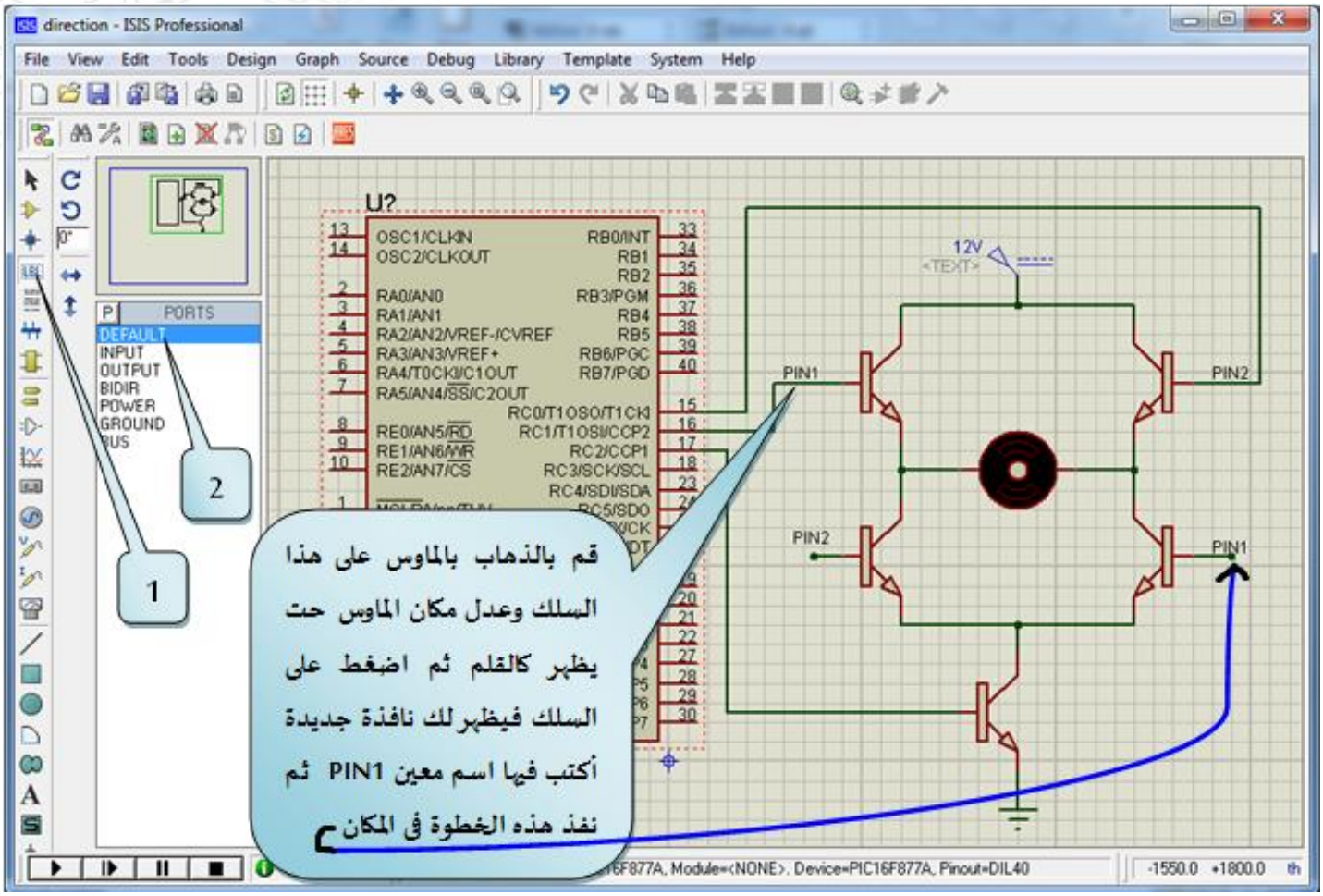


وفيها يجب ملاحظة أين تم وضع الترانزستور الخاص بالتحكم في السرعة ..

تم وضع سويتش لتغيير اتجاه الدوران على الرجل RB0، ومقاومة متغيرة للتحكم في السرعة على أحد رجول الأنالوج وليكن AN4.

وربما تكون قد لاحظت أيضا عدم توصيل الترانزستور الأيمن في الأسفل بالترانزستور الأيسر العلوي وأيضا بالمثل الترانزستور السفلي الأيسر في الـ H-bridge وربما تكون لاحظت وجود كلمة على طرف كل منهما وربما يدور بذهنك الآن بعض الاستفسارات الآن ... دعنا نجيب عليها حالا ...

لقد قمنا بتوصيل كل طرف في مكانة الصحيح ولكن بطريقة خفية دعنا نتعلمها في الشكل الآتي:



ملحوظة:

إذا كان الموتور أو الحمل الذي سيتم التحكم فيه باستخدام الدائرة السابقة يحتاج تيار عالي ينبغي عليك أن تختار ترانزستور يتحمل هذا القدر من التيار ...

أيضا يتم توصيل مقاومة واحد كيلو أوم بين الميكرو وقاعدة الترانزستور 2n2222.

تقليل تعقيد الدائرة

أعتقد أننا بحاجة لتسهيل الدائرة أكثر من هذا، بمعنى أنه بدلا من أن نقوم نحن يدويا بتركيب الترانزستورات وتكوين الدائرة سنتعامل مع IC يحتوي بداخله على دائرتي التحكم في السرعة والاتجاه معا ويخرج لنا فقط الرجول التي نضع قيم الجهد اللازمة لعملية التحكم، يحمل هذا الـ IC

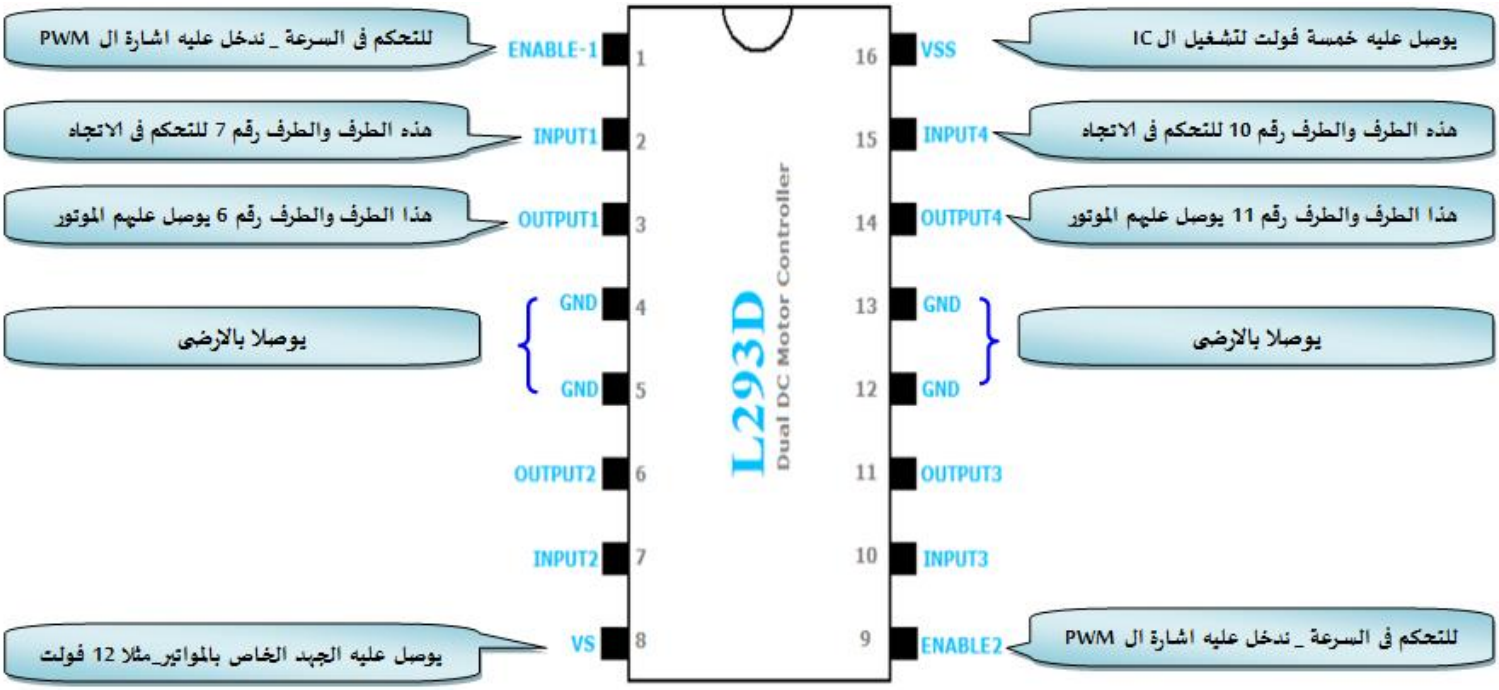
الأرقام L293B أو L293D أو L298 ويكمن الاختلاف بين الثلاثة موديلات في قيم التيار القصوى التي يمكن لكل IC تحملها، وبالتالي يجب تحديد تيار الحمل أو الموتور وبناء عليه يتم اختيار الموديل المناسب من الثلاثة السابقين ...

دعنا نتخيل هذا الـ IC قبل أن نراه، دعنا نتوقع عدد رجوله ...

١) نحتاج رجلين لتوصيل طرفي الموتور نفسه.
٢) نحتاج رجل لتوصيل الجهد الخاص بالموتور (١٢ فولت) وبالتالي نحتاج رجل أخرى لتوصيل الأرضي.
٣) نحتاج رجلين لدخلين الـ H-bridge للتحكم في الاتجاه.
٤) نحتاج رجل أخرى للتحكم في السرعة.
٥) وأيضا رجل إضافية لندخل عليها ٥ فولت لازمة لتشغيل الـ IC حيث أن أي IC لكي يعمل لابد أن يوصل له جهد معين.

بعد قراءة السطور السابقة يمكن توقع ٨ رجول للـ IC، ولكن إذا قلنا أنه يستخدم للتحكم في اثنين موتور في نفس الوقت وليس موتور واحد وبالتالي يكون له عدد ١٦ رجل ...

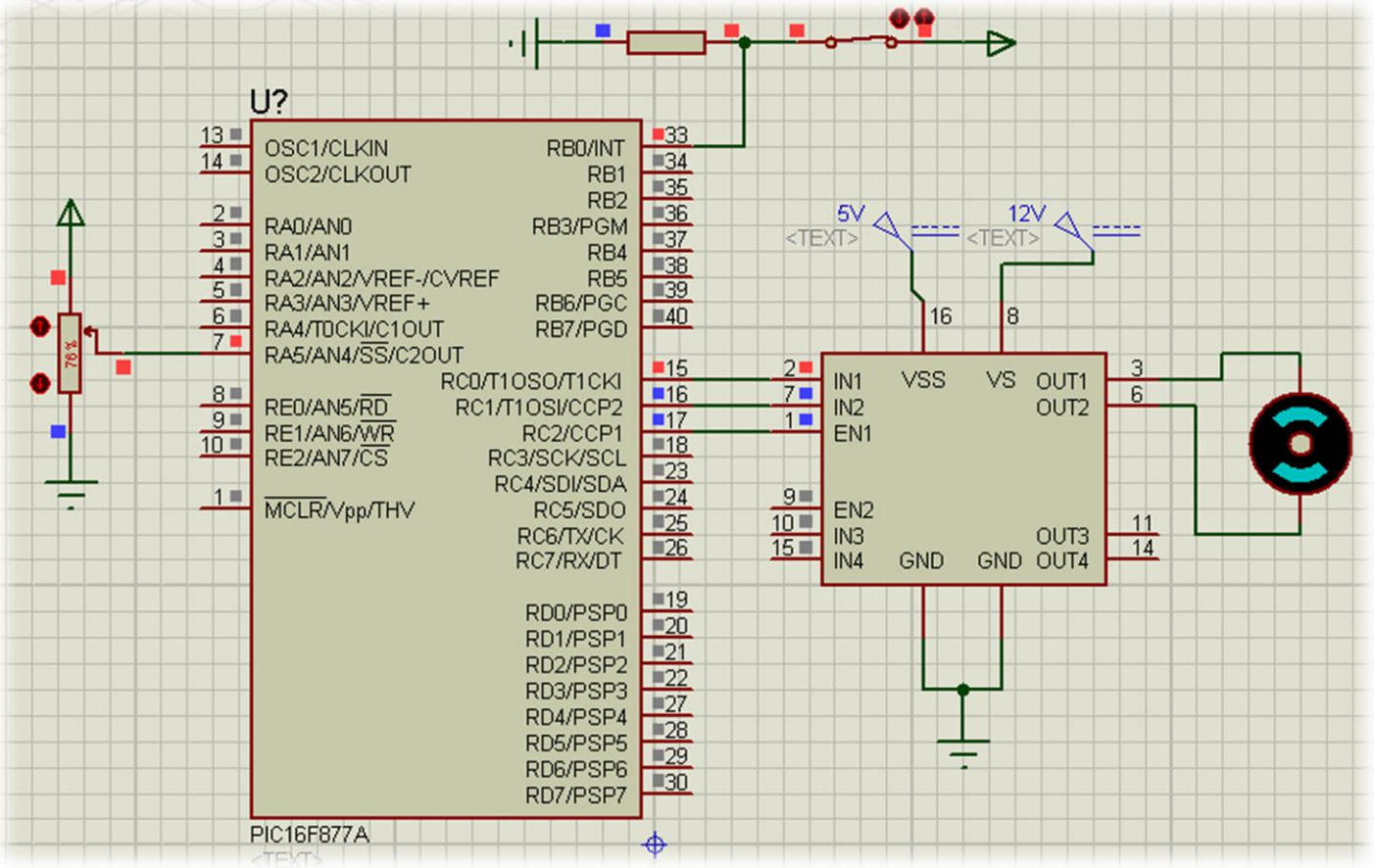
لنفترض أننا سنعمل على الـ IC رقم L293B أو L293D فهما متشابهان جدا مع التأكيد أن التعامل مع الموديلات الأخرى سيختلف في أشياء بسيطة جدا منها التيار ولكن عدد الرجل وترتيبها وشكل الـ IC متشابه تماما ...



هذه الصور مقتبسة من موقع eXtremeElectronics.co.in

في الصورة السابقة يتم توصيل الموتور الأول بالرجول اليمنى والموتور الثاني بالرجول اليسرى ...

وبناء عليها سيصبح تصميم الدائرة كما بالشكل:



وبالتالي فقد تم استبدال الـ IC بالدائرة السابقة على نفس الرجول دون تغيير في الباقي وبالتالي فإن البرنامج سيكون واحد سواء في حال استخدام الـ IC أو في حال عدم استخدامه ...

برنامج الميكروسي

```
int duty;
int V;

void main()
{
  TRISB.B0 = 1;    TRISC.B0 = 0;   TRISC.B1 = 0;

  PWM1_Init(500);     ADC_Init();
  PWM1_Start();
```

لضبط RC0,RC1 كخرج و RB0 كدخل

أوامر تهيئة الموديولات

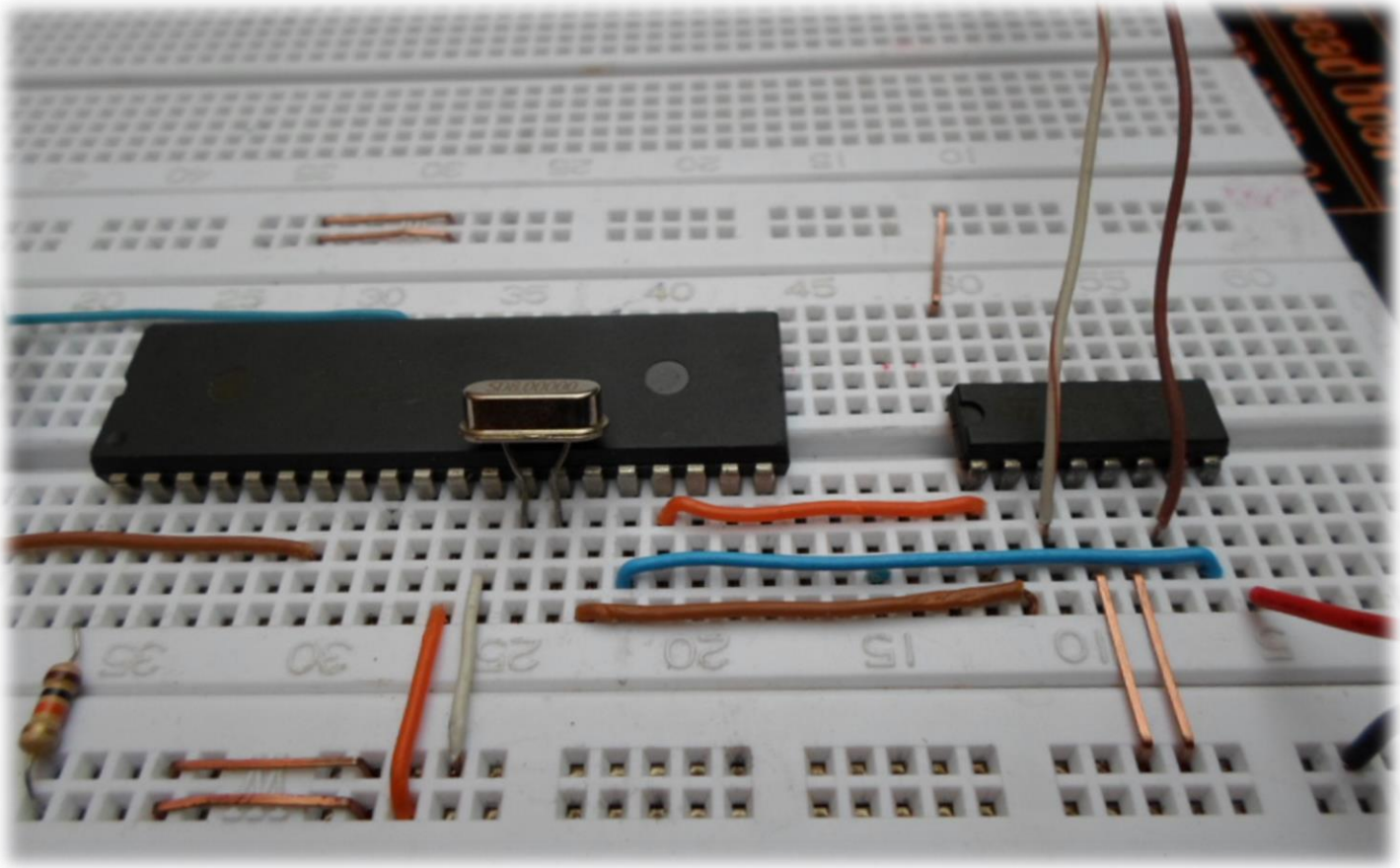
```
while(1)
{
  if(PORTB.B0 == 1)
  {
    PORTC.B0 = 1;
    PORTC.B1 = 0;
  }
  else
  {
    PORTC.B0 = 0;
    PORTC.B1 = 1;
  }
    V = ADC_Read(4);
    V = (V *5)/1023 ;
    duty = (V*255)/5;
    PWM1_Set_Duty(duty);
}
}
```

اذا كان السويتش مغلق يتم الدوران فى اتجاه وان لم يكن مغلق يتم الدوران فى الاتجاه الاخر

لقراءة قيمة الجهد الناتج عن تغير قيمة المقاومة وتحديد قيمة عرض النبضة لتغيير السرعة

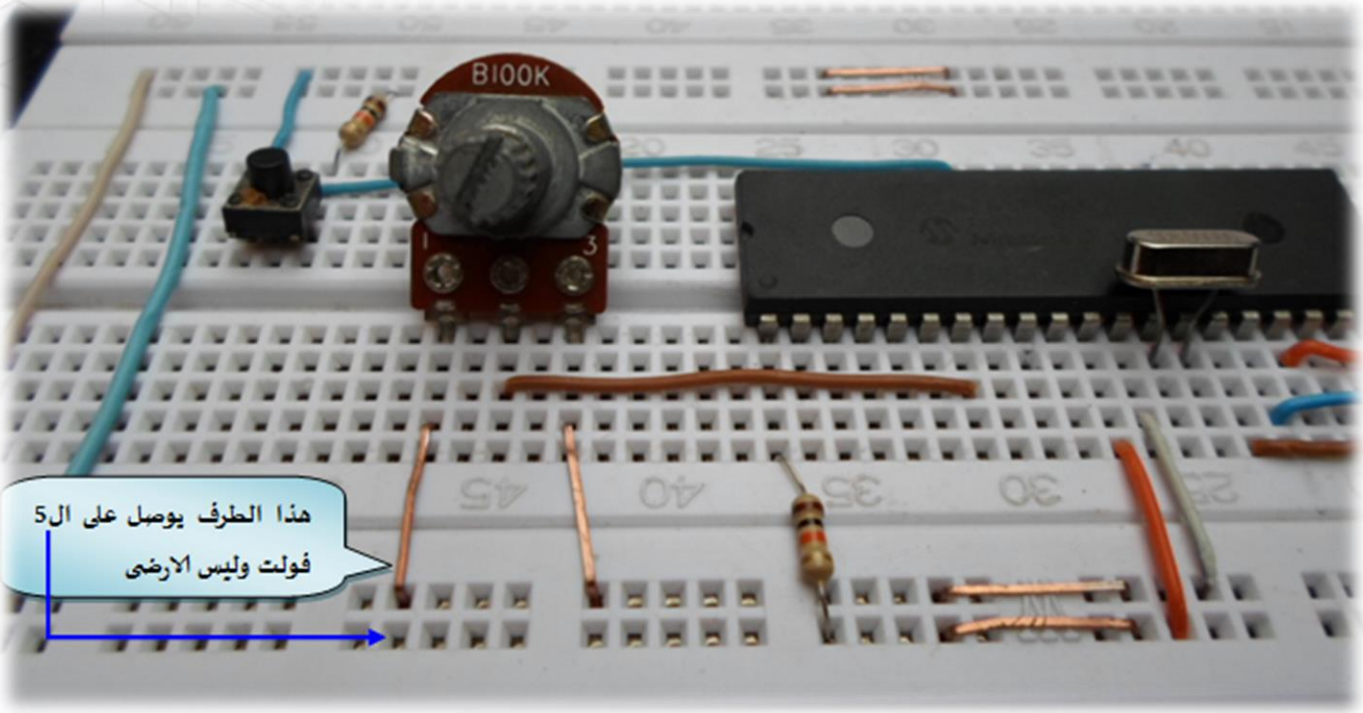
الهاردوير

توصيل الميكروبالـ L293

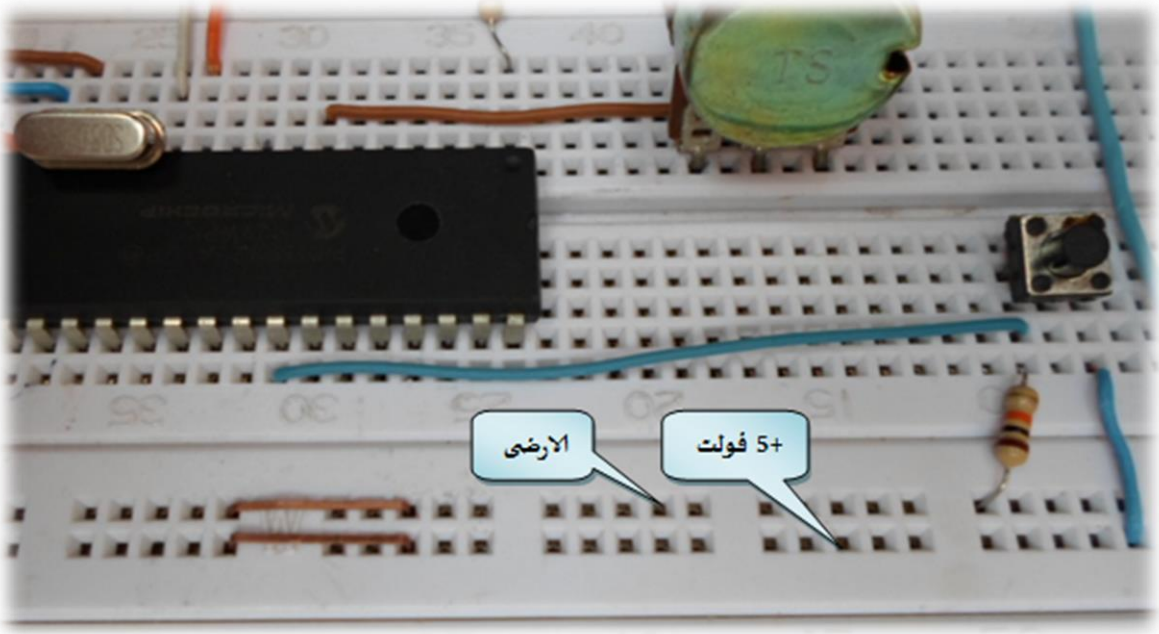


لاحظ أن طرفي الـ L293 رقم ٣ و ٦ خارج منهم سلكين متصلين على الموتور، أيضا الطرف ٨ متصل ببطارية تمثل الجهد اللازم لتشغيل الموتور ... لاحظ جيدا باقي التوصيلات ...

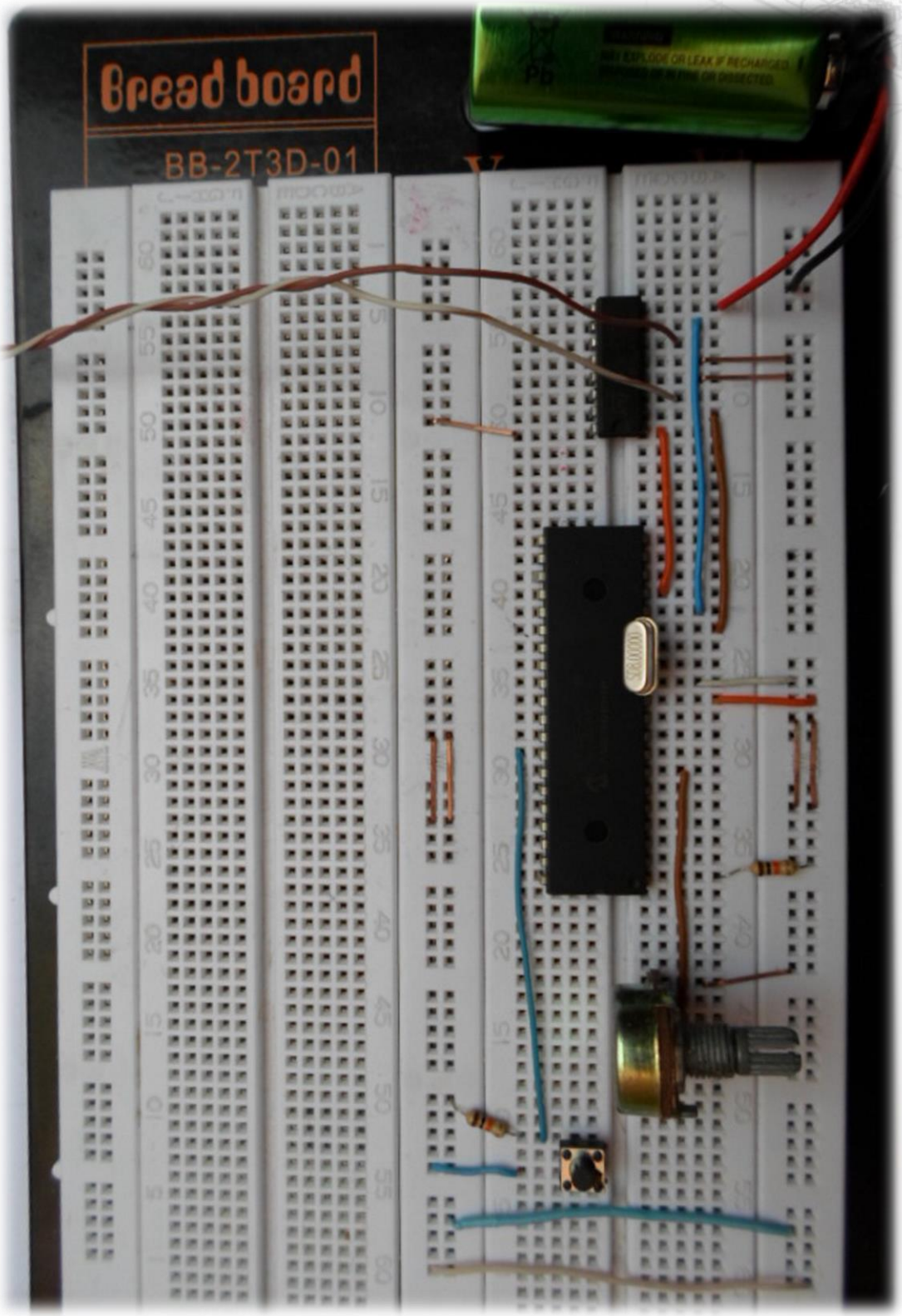
توصيل المقاومة المتغيرة بالميكرو



توصيل السويتش

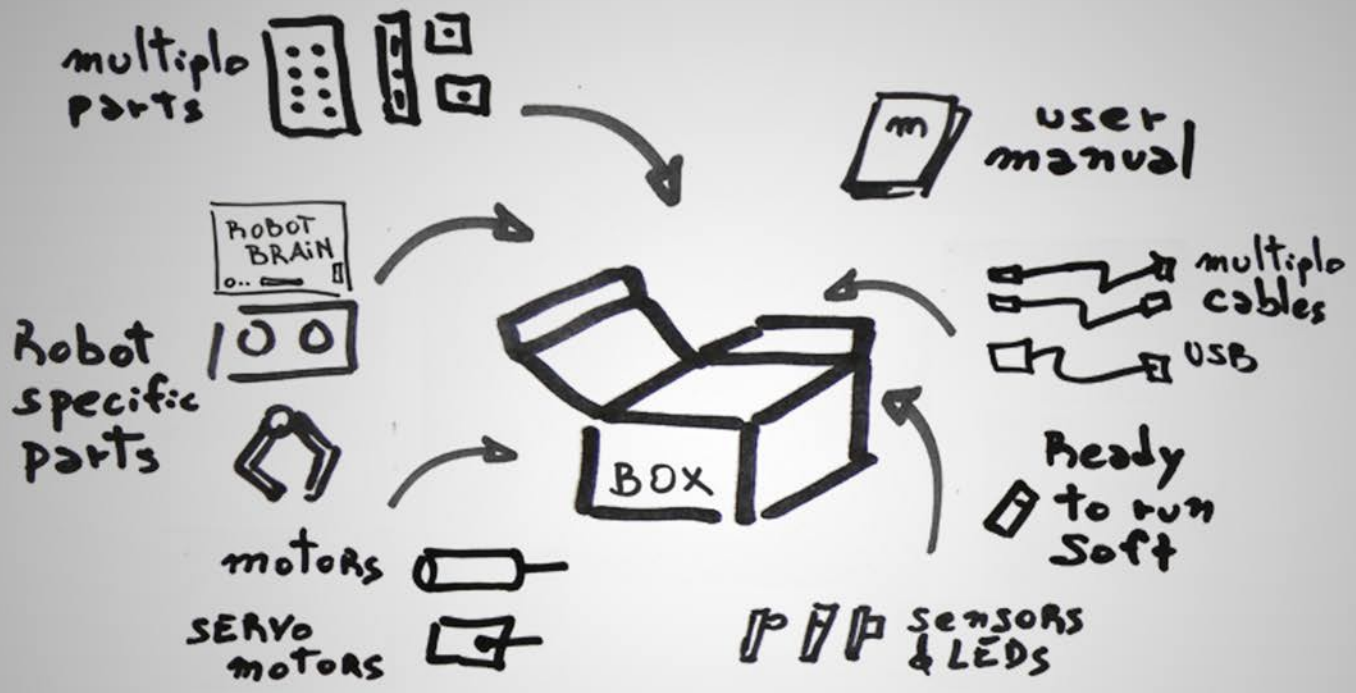


المشروع كاملا



Since 1975

El Gammal Electronics



Main Office
5 Mansheet Almahrany
Altahrer Cairo Egypt
Tel 002 02 27922213
Fax 002 02 27959279

Store 1
1 Boston Ebn Korish
Altahrer Cairo Egypt
Tel 002 02 27943760

Store 2
2 El Amir Kadadar
Altahrer Cairo Egypt
Tel 002 02 27960753

Store 3
23 Abd El Salam Aref
Altahrer Cairo Egypt
Tel 002 02 23961908

www.elgammalelectronics.com
Info@elgammalelectronics.com

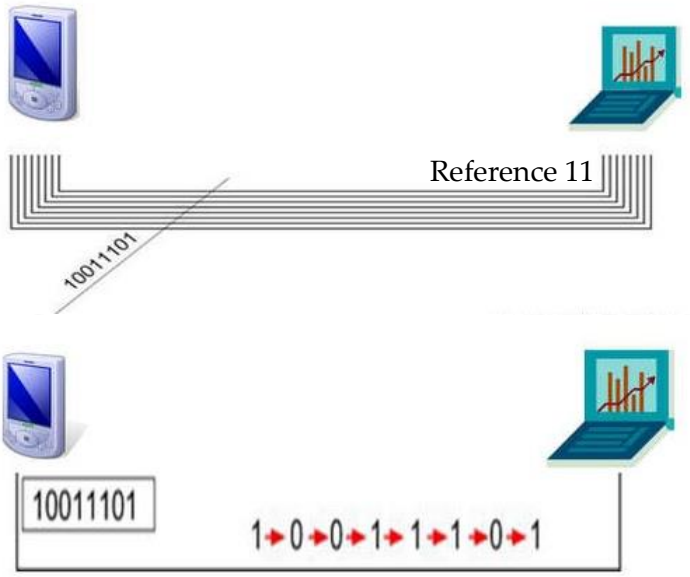
الفصل الثاني عشر

الاتصال التسلسلي

الاتصال التسلسلي هو بروتوكول يستخدم لربط جهازين أو أكثر مثل توصيل الكمبيوتر بالميكرو أو توصيل الميكرو بميكرو آخر أو غير ذلك، وفيه يتم نقل الداتا بطريقة تتابعية (Serial) على وصلة واحدة بين الجهازين يرسل عليها البت الأول يليه البت الثاني ... وهكذا، وهو عكس التوصيل البارالليل والذي فيه يتم نقل كامل البايت مرة واحدة على ٨ وصلات (أسلاك).

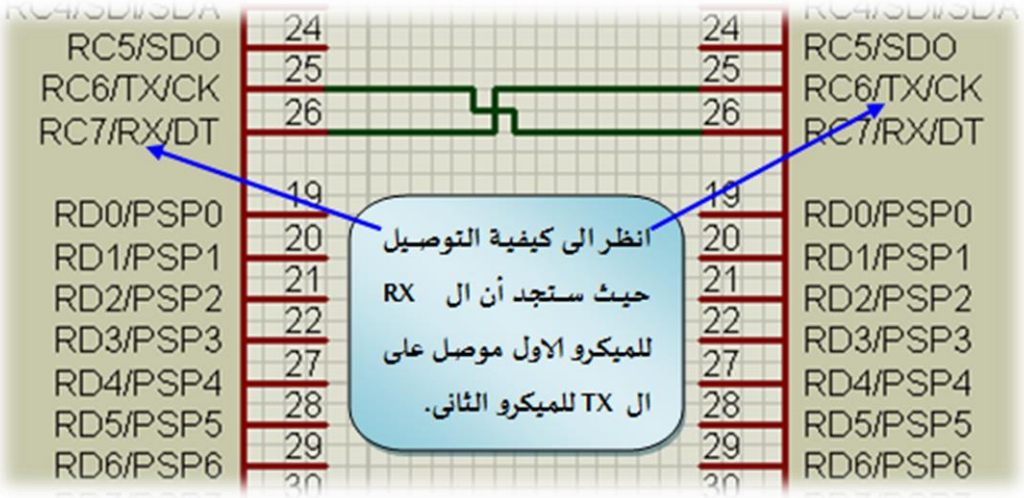
الاتصال التسلسلي Serial Communication

إذا قمت بالبحث عن ترجمة كلمة Communication لوجدت أنها تعنى اتصال أو طريقة اتصال وأما كلمة Serial فهي تعنى تسلسلي أو تتابعي وبالتالي فإن الجملة كاملة Serial Communication تعني طريقة اتصال لنقل الداتا بين جهازين أو أكثر على أن يتم نقل الداتا بطريقة تتابعية بمعنى أن يتم إرسال البت الأول من البايت ثم بعده البت الثاني ثم الثالث ... وهكذا حتى يتم نقل البايت كاملا، والشكل الاتي يوضح الفرق بين نقل الداتا بطريقة تتابعية (بالأسفل) ونقلها بالتوازي (بالأعلى).



وبالنظر إلى الشكل السابق سنجد انه في حالة السريال استخدمنا سلك (Line) واحد فقط، وهذا ينتج عنه مشكلة وهي أننا لا يمكننا الإرسال والاستقبال في نفس الوقت على سلك واحد، فإما أن تقوم بالإرسال أو أن تقوم بالاستقبال ...

والحل البسيط لهذه المشكلة هو أن يتم وضع سلكين بين الجهازين يستخدم أحدهما للإرسال والآخر للاستقبال، وعندها سيكون التوصيل بين اثنين ميكروكنترولر مثلا كما بالشكل الآتي:

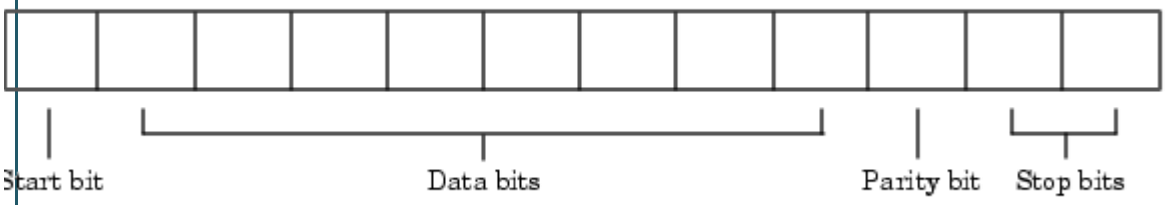


لاحظ في الشكل السابق أن طرف الإرسال للجهاز الأول متصل على طرف الاستقبال للجهاز الثاني والعكس بالعكس في الطرفين الآخرين، وهذا منطقي إذ أنه في الوقت الذى يكون فيه الجهاز الأول يقوم بالإرسال عندئذ يقوم الجهاز الثاني بالاستقبال.

معلومة إضافية: كل بايت يتم إرساله يرسل معه three bits إضافية:

- أولهم يسمى الـ Start Bit وهو يستخدم لكي يتعرف من خلاله المستقبل على بداية البايت وبالتالي يرسل قبل هذه البداية ...
- وثانيهم يسمي الـ Stop Bit وهو يستخدم لكي يتعرف المستقبل من خلاله على نهاية البايت وبالتالي يتم إرساله بعد هذه النهاية
- والبت الثالث والأخير يسمى الـ Parity Bit وهو يستخدم لكي يستخدمه المستقبل ليختبر حدوث تلف للداتا المخزنة في البايت أثناء الإرسال من عدمه.

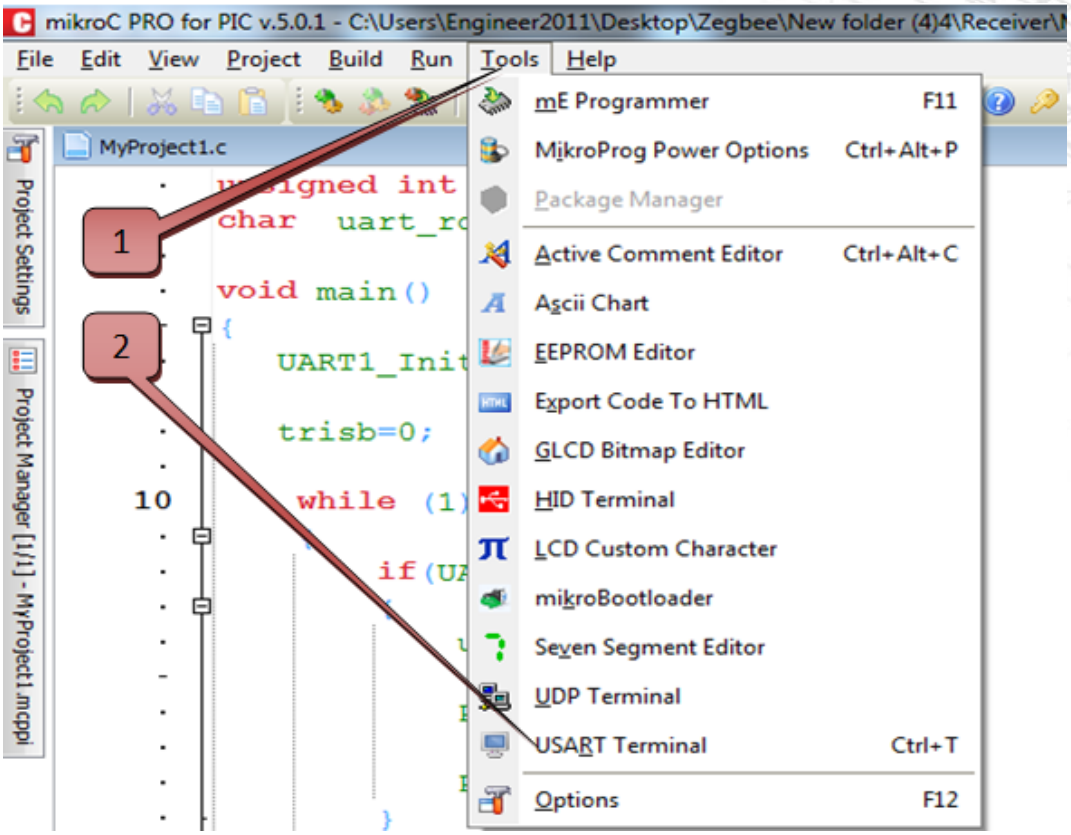
وبالتالي يصبح شكل كل بايت يتم نقله كالآتي:



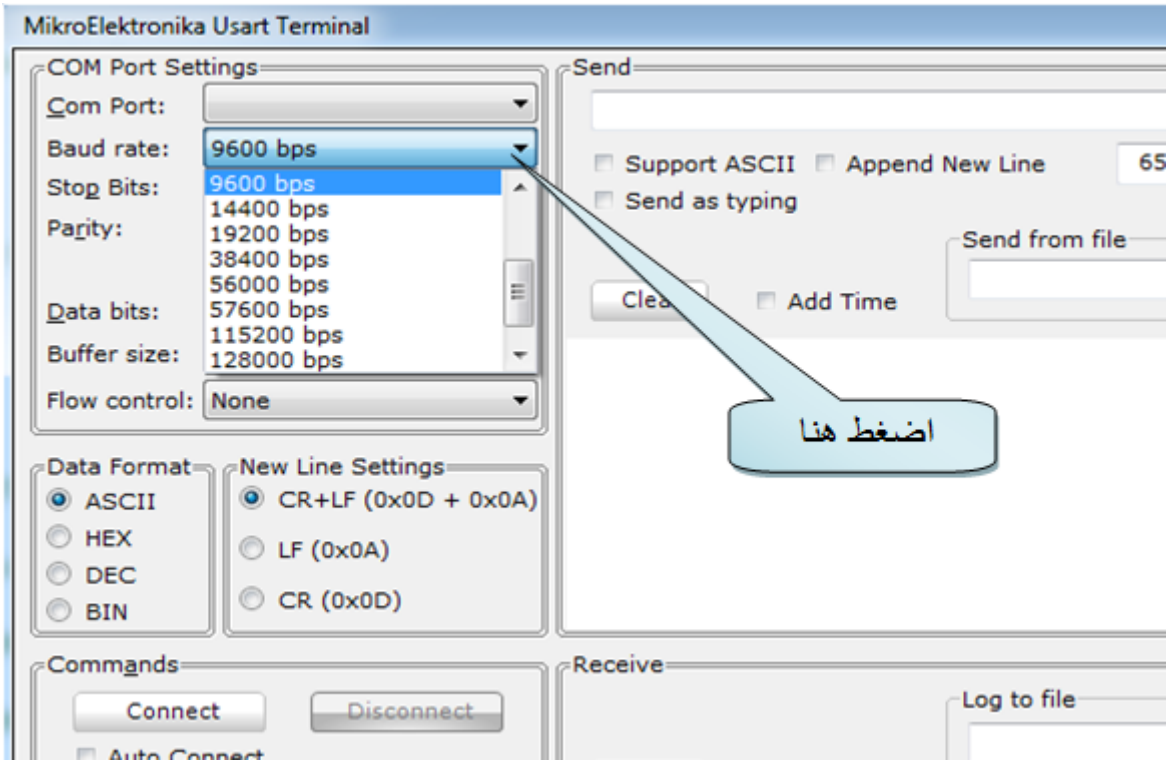
ولكن هذه التفاصيل لا تهمنا كثيرا في الشغل العملي.

برنامج الميكروسي

لنقل الداتا بطريقة تتابعية لابد من تحديد مجموعة من العناصر أهمها سرعة النقل، والمقصود بالسرعة هنا هو عدد الـ Bits التي سيتم إرسالها في كل ثانية، والتي لابد من تحديدها حيث إن كان المرسل يرسل البيانات بسرعة معينة والمستقبل يستقبل بسرعة أقل فسوف تضيع بيانات لن يستقبلها المستقبل، وبصفة عامة فإن عدم تطابق السرعة عند المرسل والمستقبل يؤدي إلى أن المستقبل لن يقرأ الداتا بالشكل الذي كما أرسلت له به، والسرعة يطلق عليها الـ Baud rate والسرعة محددة افتراضيا في أي جهاز هي 9600 بت في الثانية الواحدة، وبالطبع يمكنك تغيير هذه القيمة، ولكن لا يتم تحديد أي قيمة عشوائية فهي قيم قياسية محددة يمكنك تحديدها بالطريقة التالية:

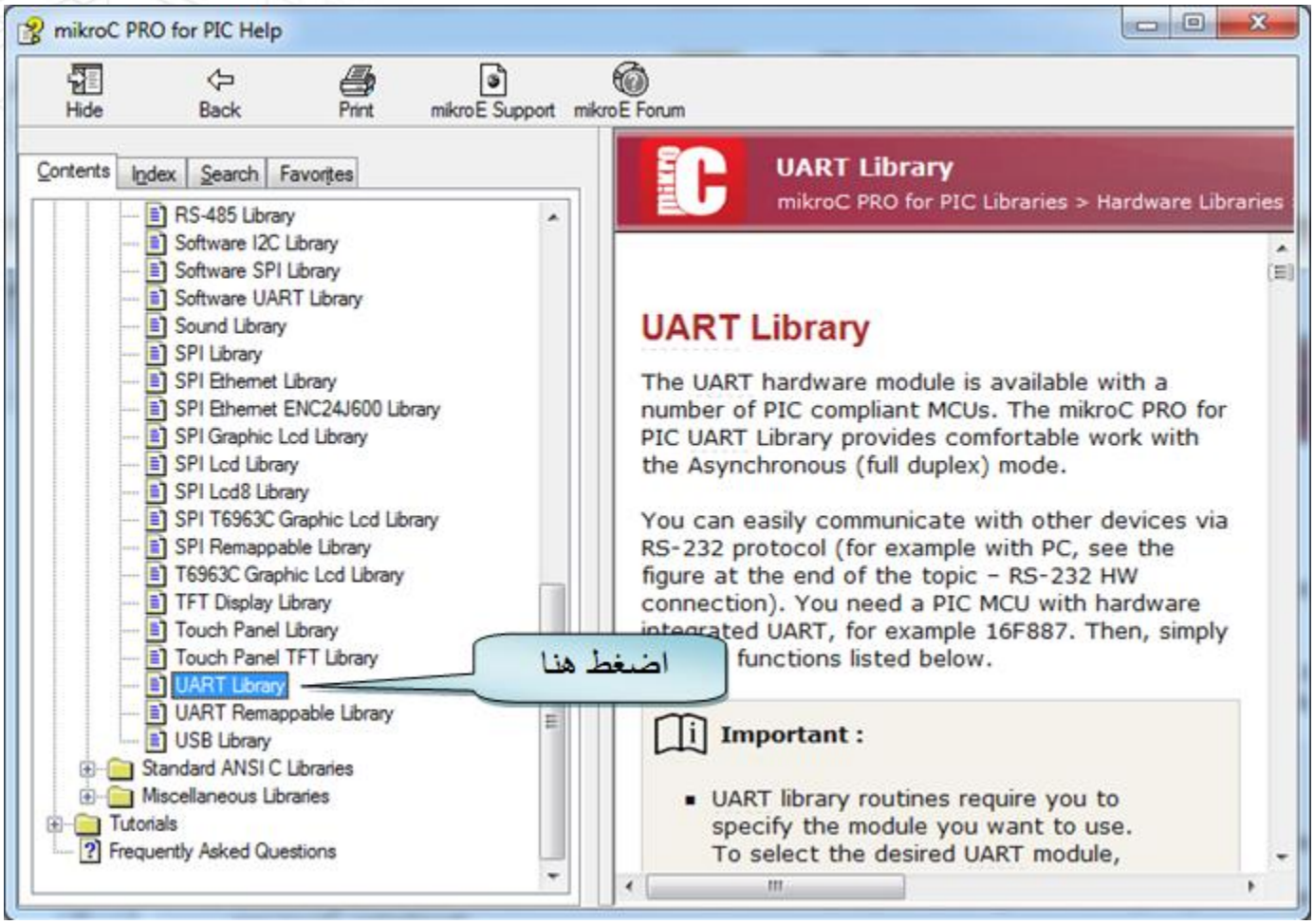


فتظهر لك النافذة الآتية ومنها يمكن اختيار قيمة السرعة من بين القيم المحددة كما يلي:



دوال الميكروسي

سنتناول الآن دوال الميكروسي التي تستخدم مع الـ Serial Communication، وهي أيضا بالطبع يمكن الحصول عليها من نافذة المساعدة كما تعودنا وذلك كما بالشكل الآتي:



الدالة الاولى

من خلال فهمنا لما سبق فإننا نحتاج دالة لتحديد السرعة وهي كالآتي:

```
Uart1_Init(9600);
```

حيث يتم كتابة السرعة بين أقواسها كما هو مبين، ويتم كتابة هذه الدالة داخل الدالة الرئيسية.

الدالة الثانية

وهي ودالة يستخدمها الجهاز لكي يقوم بإرسال البيانات للجهاز الآخر حيث تستخدم لإرسال حرف (أو بايت) فقط كالآتي:

```
Uart1_Write(Data);
```

حيث أن المتغير Data هو متغير لا تزيد قيمته عن ٨ بت إذ أن هذه الدالة مصممة على هذا الأساس، وسيتبين في مشروع هذا الفصل كيفية إرسال بيانات حجمها أكبر من ٨ بت ...

الدالة الثالثة

وهي دالة تستخدم لاستقبال البيانات التي تأتى إليه من الجهاز الآخر، حيث تستخدم لقراءة حرف واحد بالشكل التالي:

```
Uart1_Read( );
```

وحيث هذه الدالة ترجع حرف فلابد من تخصيصها لمتغير – وليكن مثلا من النوع الحرفي – يتم تخزين فيه ما ترجعه هذه الدالة.

وهنا ملحوظة لابد من ذكرها وهي أنه عندما يقوم الجهاز الأول بإرسال بايت للجهاز الثاني فإن هذا البايت يتم تخزينه في Buffer إلى أن يتم قراءته، وبالتالي لضمان وجود داتا في الـ Buffer قبل دالة القراءة لابد من اختبار وجود حرف في الـ Buffer أولا قبل عملية القراءة وإن لم يكن به داتا لا نقوم بعملية القراءة من الأساس وهذا يتم من خلال الكود الآتي:

```
if(UART1_Data_Ready())
{

  uart_rd = UART1_Read();

}
```

اذا كانت فيه داتا وصلت ينفذ ما بداخل ال if

ملحوظات

- كتابة الدالة كالآتي Uart1_Read() أو كالآتي UART1_Read() صحيحا ويمكنك تجربة ذلك بنفسك ...
- توجد أيضا دوال أخرى يمنك الاطلاع عليها ومنها على سبيل المثال دالة تستخدم لإرسال جملة كاملة مثل:

```
Uart1_Write_Text("Engineer");
```

- نلاحظ أيضا في كل الدوال السابقة وجود الرقم ١ وهذا معناه انه يمكن أن يكون هناك دوال ملحق بها الرقم ٢ أيضا إذا كان الميكرو الذي نتعامل معه يحتوي على موديولين يتعاملان مع الـ Serial Interface ...

مشروع تطبيقي

يحتوي المشروع على اثنين من الميكروكنترولر، الأول سيقوم بقراءة قيمة مقاومة (باعتبارها سينسور مثلا) ويقوم بإرسال هذه القيمة إلى الميكرو الثاني، ويقوم الميكرو الثاني بالتحكم في سرعة موتور على أساس القيمة التي استلمها من الميكرو الأول ...

قبل البدء في المشروع قد يتبادر إلى الذهن سؤال وهو: لماذا استخدمنا اثنين ميكروكنترولر في حين أن استخدام واحد فقط يغني؟؟ ... وهو فعلا سؤال منطقي وسنجد إجابته في نهاية المشروع ...

الميكرو الأول

سيقوم بقراءة المقاومة المتغيرة – وبالتالي نحتاج موديول الـ ADC Interface – ثم يقوم بإرسالها إلى الميكرو الثاني وبالتالي سنحتاج إلى الـ Serial Interface ومن ذلك سنستخدم دالتي الـ initialization الآتيتين داخل الدالة الرئيسية:

```
ADC_Init();
UART1_Init(9600);
```

وعندئذ سيكون البرنامج الذي سينفذه الميكرو الأول المرسل (Transmitter) كالآتي:

```
int x;
char y;
void main()
{
   UART1_Init(9600);
   ADC_init();

    while (1)
    {
      X = Adc_Read(0);
      y = X/4;
      UART1_Write(y);
    }
}
```

لتعريف متغير من النوعية الصحيحة لان ال ADC بيقرأ القيمة فى 10 بت

لتعريف متغير من النوعية الحرفية حجمه واحد بايت لان دالة الارسال تقوم بارسال 1بايت

هذا الامر يستخدم لقراءة الجهد المتغير الناتج عن تغير قيمة المقاومة

هذا الامر يقسم القيمة التى قرأناها على 4 وسنعرف لماذا فى الشرح

هذا الامر يستخدم لارسال القيمة – بعد القسمة - للميكرو الثانى

وهو برنامج سهل وبسيط لكن به سؤال يقول: لماذا قمنا بقسمة القيمة على ٤ ... والإجابة تتلخص في أن الـ ADC يقرأ القيمة في ١٠ بت وبالتالي أقصى قيمة يمكنه قراءتها هي ١٠٢٣، ولكن الدالة `uart1_write` ترسل واحد بايت فقط أي ٨ بت فقط في كل مرة وأقصى قيمة لهذه الـ ٨ بت بالنظام الثنائي هي ٢٥٥، وبالتالي لا يمكننا قراءة القيمة من الـ ADC ثم إرسالها مباشرة، وبالتالي

كانت القسمة على ٤ حيث أنه لو كانت القيمة المقروءة هي القيمة العظمى ١٠٢٣ ثم قسمناها على ٤ فستكون قيمتها – تقريبا – ٢٥٥ ولن تزيد عن ذلك، أي أن القيمة بعد القسمة أصبحت مخزنة في ٨ بت وليس ١٠ كما كانت قبل القسمة وعندئذ يمكننا إرسالها ...

حتى هنا يكون الميكرو الأول قد قام بقراءة قيمة المقاومة المتغيرة وإرسالها والان جاء دور الميكرو الثاني ليقوم بوظيفته ...

الميكرو الثاني

يقوم الميكرو الثاني باستقبال القيمة المرسلة له من الميكرو الأول من خلال الـ Serial Interface وبالتالي سيقوم بقراءة القيمة من الـ Buffer ثم يقوم على أساسها بتغيير سرعة الموتور وهذا ما يجعلنا نحتاج إلى الدالتين الآتيتين في بداية البرنامج:

```
UART1_Init(9600);
PWM1_Init(500);
```

ويرجى مراجعة كيفية التحكم في سرعة المواتير من الفصول السابقة حتى يتسنى لك فهم المشروع جيدا، فقد كنا نتحكم في سرعة الموتور عن طريق التحكم في عرض النبضة باستخدام الدالة:

```
PWM1_Set_Duty();
```

وهذه الدالة يرسل إليها قيمة من ٠ إلى ٢٥٥ كما سبق وشرحنا وبالتالي سيكون البرنامج كالآتي:

```
char  value;

void main()
{
   UART1_Init(9600);
   PWM1_Init(500);

   while (1)
    {
        if(UART1_Data_Ready())
        {
            value = UART1_Read();

            PWM1_Set_Duty(value);

            PWM1_Start();
        }
    }
}
```

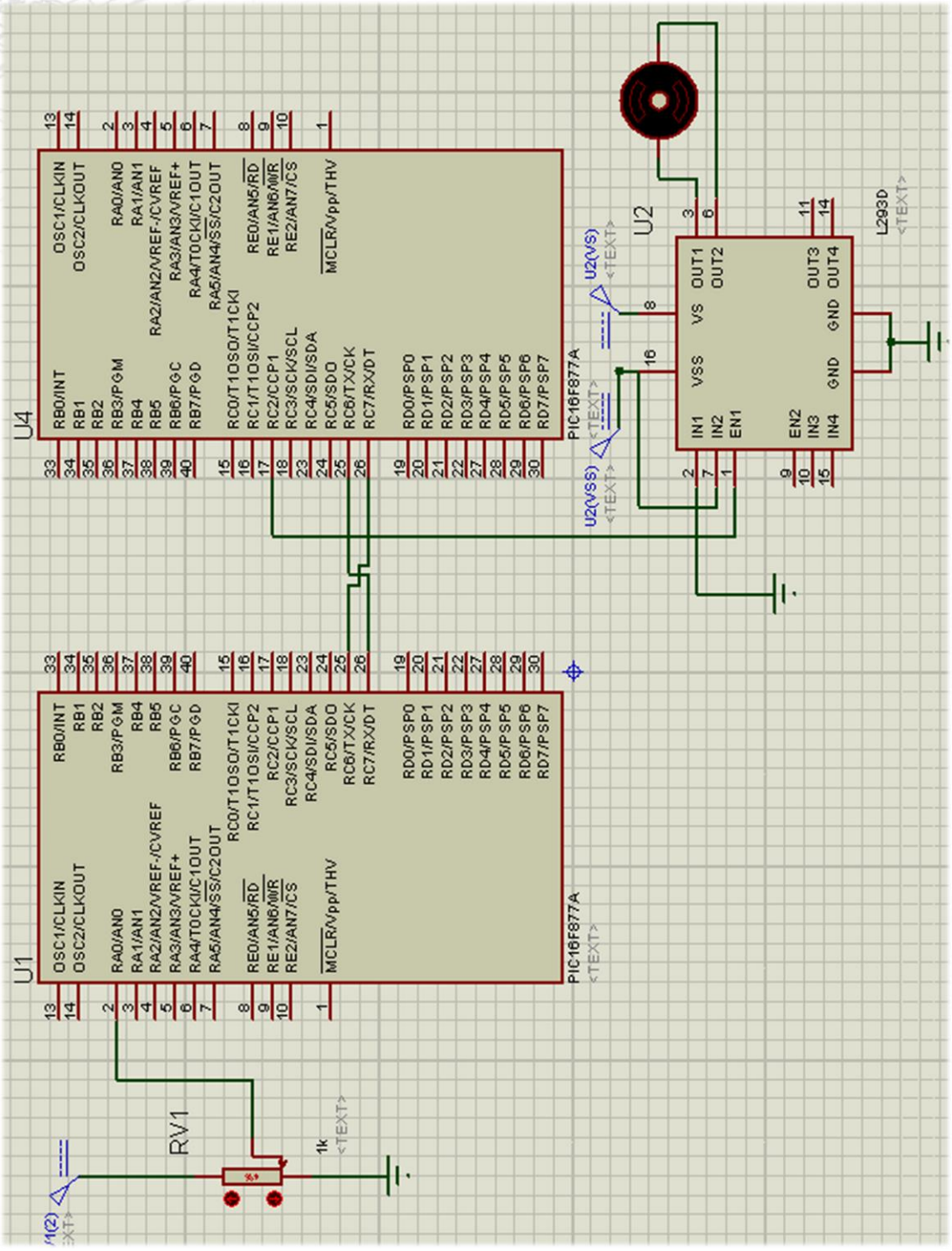
لعمل تهيئة للموديولات

لقراءة القيمة وتخزينها فى المتغير value

نعطى القيمة الى هذه الدالة

لاحظ أن القيمة التي تم استقبالها تتراوح من ٠ إلى ٢٥٥ لأننا قسمنها على ٤ عند المرسل، والدالة `set_duty` تأخذ قيما من ٠ إلى ٢٥٥، وبالتالي أخذنا القيمة وأعطيناها مباشرة للدالة ...

المحاكاة

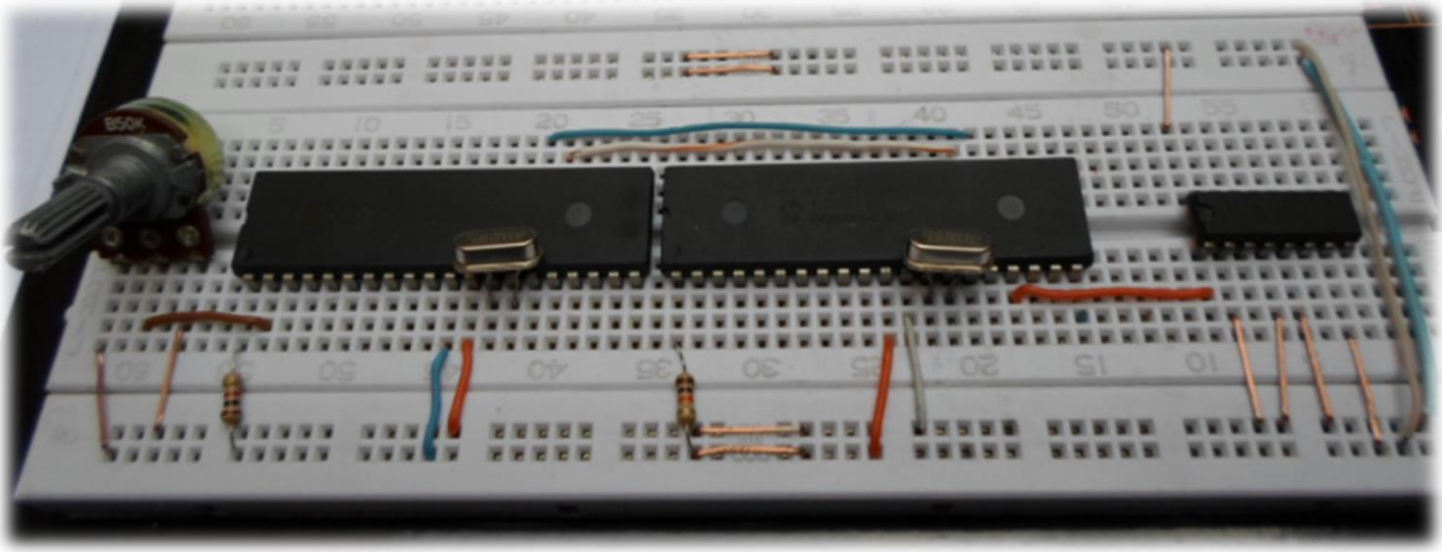


ويمكنك تشغيل المحاكاة وزيادة أو تقليل قيمة المقاومة وتلاحظ التغيير في سرعة الموتور.

ملحوظة: فيما سبق قمنا بتوصيل ميكرو بآخر باستخدام سلكين فقط ولكن توصيل الميكرو بالكمبيوتر لن يكون بهذه البساطة بل سنحتاج إلى الـ IC MAX232 وهذا ما سنتعرف عليه في الجزء الثاني من الكتاب إن شاء الله ...

الهاردوير

فيما يلي الصور الخاصة بالهاردوير، ولكن فيها لم أقم بتوصيل مصدر الجهد الخاص بالموتور والذي يوضع على الطرف ٨ من الـ L293D، أيضا يجب أن نؤكد أن الطرفين ٣ و ٦ هما اللذان يتصل عليهما الموتور كما سبق وتعلمنا ...



إجابة على السؤال المذكور في بداية المشروع بشأن الحاجة لتوصيل ٢ ميكروكنترولر ببعضهما فإننا في الواقع لا نحتاج كثيرا لتوصيل ميكروكنترولر بآخر، ولكننا قد نضطر إلى هذا في بعض الظروف مثل أن يكون هناك ميكروكنترولر في مكان ميكروكنترولر آخر في مكان آخر ويتم نقل البيانات بينهما باستخدام بعض الموديولات مثل (البلوتوث أو ZegBee أو غيرهم) وفى هذه الحالة يكون الميكرو الأول موصل بالمودويل المرسل والميكرو الآخر موصل بالمودويل المستقبل، ثم يتم نقل البيانات بين كل ميكرو والمودويل الموصل معه بطريقة السيريال ... وبالتالي لن يحدث تغيير كبير في أكواد المشروع السابق عند استخدام مودويل البلوتوث ..

هذا الجزء قمت بتخصيصه للتعريف ببعض المصادر الهامة التي يمكنك أن تستعين بيها في هذا المجال.

وبالطبع لابد أن أبدأ بأهم المصادر المتاحة باللغة العربية وهي كتب م. عبد الله علي والذي قام بتأليف سلسة من الكتب وهي:

- كتاب أردوينو ببساطة.
- كتاب دليل محاكاة أردوينو.
- كتاب راسبيري ببساطة.
- كتاب تقنيات الاختراق المادي.

وهي مجموعة من الكتب باللغة العربية وتعتبر الوحيدة في المجالات التي كتبت فيها وهي فعلا إثراء للمحتوى العربي، وتمنياتي بأننا جميعا نتخذ خطوات مثل هذه ليصبح لدينا محتوى عربي مؤثر، هذا بالإضافة إلى أن هذه الكتب مجانية يمكن تحميلها بل والتعديل فيها وستجد روابط هذه الكتب في نهاية الموضوع.

كتاب أردوينو ببساطة:

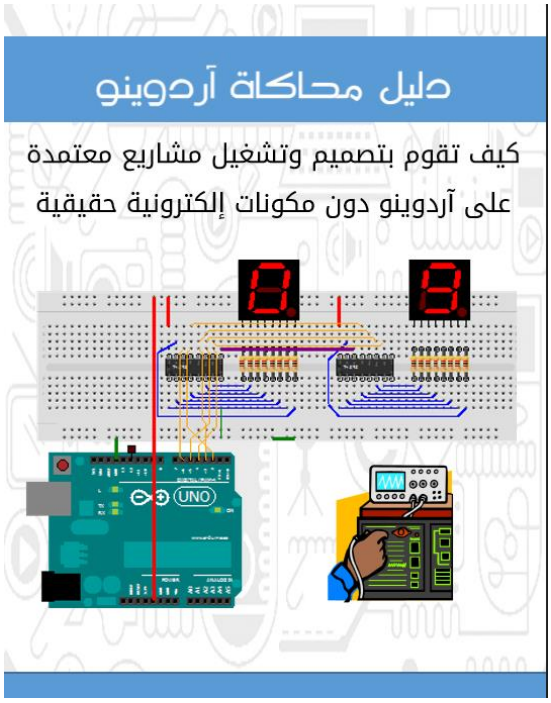
اصبحت الإلكترونيات التفاعلية تحيط بنا من كل جانب، ستجدها ترافقك اينما ذهبت ... في السيارة، في بيتك، في العمل، أو حتى في جيبك الصغير، هناك دائما آداه إلكترونية ذكية تتفاعل معك في أي مكان تذهب اليه؟

ما هو أردوينو Arduino؟؟ أردوينو هو مشروع كبير مفتوح المصدر يهدف الى توفير برمجيات مجانية ولوحة تطوير تفاعلية مفتوحة المصدر Open source Development Board تستخدم في بناء

دوائر إلكترونية ذكية وتستطيع التفاعل مع البشر بسهولة ويسر ويمكن لأي فرد استخدامها لعمل مشروعة الخاص دون الحاجة لمعرفة مسبقة بعلم الإلكترونيات ودون دراسة تعقيدات الدوائر الإلكترونية، وتتكون هذه اللوحة من دائرة إلكترونية تحتوي على متحكم دقيق قابل للبرمجة عن طريق برنامج Arduino IDE والذي يمكن تحميله مجانا لجميع انظمه التشغيل.

كتاب دليل محاكاة أردوينو الشامل:

الآن يمكنك تنفيذ تجارب ومشاريع إلكترونية رائعة دون الحاجة لشراء أي مكونات إلكترونية وذلك عبر محاكاة المتحكمات الدقيقة وبرامج أردوينو على الحاسب الآلي مباشرة.



جاء هذا الكتاب كمحاولة لتسهيل تعلم أردوينو لأبناء الوطن العربي، حيث يواجه الكثير صعوبة في شراء القطع الإلكترونية لعدم توافرها في بعض المناطق أو لغلاء سعرها في مناطق أخرى لذلك جاء هذا الكتاب يشرح حل مفصل للتغلب على هذه المشاكل، فخلال صفحات هذا الكتاب ستتعرف على ٨ تقنيات لمحاكاة أردوينو على مختلف أنظمة التشغيل مثل Windows – Linux – Mac – Android – IOS وجميع الأنظمة المشتقة منها

شكر خاص للأخت سنا حواصلي من مجموعة (فابلوجيا) في هندسة دمشق بسوريا على مقالاتها الرائعة والمساهمة بشكل أساسي في هذا الكتاب

كتاب راسبيري باي ببساطة

ما هو الراسبيري باي Raspberry Pi؟ هو حاسوب صغير في حجم كف اليد بسعر ٢٥ دولار واستهلاك طاقة أقل من ٣ وات، تم تصميمه في جامعة كامبريدج للمساعدة في تدريس علوم الحاسب بحيث يدمج ما بين أنظمة لينكس وعلم البرمجة والإلكترونيات وأنظمة التحكم الذكية في ذات الوقت، مما جعل هذا الحاسوب الصغير يحقق نجاح مذهل في كلا المجالين التعليمي والتطبيقي.

ماذا أستطيع أن أفعل بالراسبيري؟ يمكنك استخدام الراسبيري كأي حاسوب تقليدي لتصفح الإنترنت وإرسال البريد الالكتروني وحتى تحرير الملفات والوثائق عبر حزمة LibreOffice المكتبية، أيضا تستطيع تحويل أي تلفاز عندك إلى نظام ترفيه منزلي متصل بالإنترنت، وكذلك

يمكنك عمل مشاريع تحكم إلكترونية مذهلة واستخدام الراسبيري كبديل متطور جدا عن المتحكمات الدقيقة Microcontrollers.

كتاب تقنيات الاختراق المادي:

الأمن المادي Physical Security: تعرف تقنيات الأمن المادي للمعلومات بأنها جميع الإجراءات والأجهزة ووسائل الحماية المستخدمة في الحفاظ على البيانات من السرقة أو التشويه او الاطلاع عليها لغير المخولين لهذا الأمر، حيث يتم استخدام عدة تقنيات مثل أنظمة التحكم في الوصول للمباني والغرف Access control والبوابات الذكية Smart doors وأنظمة تحديد الهوية الرقمية RFID و ذلك لضمان وصول أشخاص معينين لهذه البيانات دون غيرهم، كذلك يتم استخدام تقنيات التشفير و التحقق من الهوية برمجيا في أنظمة التشغيل لنفس الغرض و لحماية البيانات.

يناقش كتاب تقنيات الاختراق المادي التقنيات المستخدمة بواسطة اللصوص والمتسللين Black hackers للوصول الى المعلومات بصورة مادية واختراق الحواجز الأمنية المادية والرقمية وكذلك يعرض الإجراءات المضادة للاختراق والمخصصة لتوفير الحماية ضد هذا النوع من الهجمات.

الهدف من وراء نشر هذا الكتاب هو نشر الوعي الأمني في أحد أكثر المجالات الخطيرة والمهملة في الوطن العربي، حيث نجد الشركة تهتم بالحماية البرمجية مثل برامج مضادات الفيروسات والجدران النارية firewalls وتهمل الإجراءات الأمنية المادية مما يتسبب دائما في تجاوزات أمنية خطيرة.

ويكفي أن نعرف ان أشهر طرق الاختراق الإلكترونية التي حدثت على مستوى العالم تعتمد بالأساس على الاختراق المادي Physical Hacking. ينقسم محتوى الكتاب إلى جزئين منفصلين بمجموع ٧ فصول حيث يناقش كل فصل أحد تقنيات الاختراق المادي وكذلك الإجراءات المضادة للحماية من هذا النوع.

يمكنك تحميل المجموعة الكاملة من الكتب السابقة من خلال الذهاب لهذا الرابط:

http://simplyarduino.com/?page_id=889

أيضا يمكنكم الاستعانة بالموقع الآتي للحصول على الكتب المتاحة باللغة العربية في أكثر من مجال:

librebooks.org

ولأن هذا الموقع هو موقع كتب عربية حرة هو منصة للكتب الحرة باللغة العربية، يهدف الموقع لإثراء المحتوى العربي والتعريف بالكتب والثقافة الحرة وأهميتها عربيا، بالإضافة إلى التشجيع على إنتاج المزيد من الكتب الحرة ذات جودة عالية. الموقع مفتوح لكافة المجالات المتنوعة ويمكن نشر أي كتاب طالما كان حرا.

كما أنه يوجد موقع أجنبي يحتوي على كورس في الميكروكنترولر ومجموعة كبيرة من المشاريع وهو الموقع الآتي:

http://embedded-lab.com/blog/?cat=38
http://embedded-lab.com/blog/?cat=4

ومدونة عربية أخرى تحتوي على العديد من المقالات في مجال الـ Embedded system ومقالاتها رائعة جدا:

www.genotronex.com

وموقع (اصنعها)، والذي يحتوي على شروحات كثيرة جدا لكيفية صنع العديد من الاجهزة بصورة مبسطة تمكنك أنت من القيام بذلك بأقل التكاليف:

http://isnaha.com/

وأيضا يمكنك البحث على الانترنت عن كتب م. أحمد سمير فايد حيث أنه له كتابين في مجال البك ميكروكنترولر

وأحد أهم المواقع الذي يحتوي على شرح خاص بالبك ميكروكنترولر وهو من عمل شركة MiKroe والتي قامت بعمل برنامج الميكرو سي:

www.mikroe.com/chapters/view/1/introduction-world-of-microcontrollers/
www.mikroe.com/chapters/view/14/chapter-1-world-of-microcontrollers/

وهذه أسماء مجموعة من المراجع يمكنكم شراؤها أو تحميلها للاستفادة منها:

Advanced PIC Microcontroller Projects in C
Interfacing PIC Microcontrollers (16F877A)
PIC Microcontrollers for absolute beginners
PIC Microcontrollers program in C
The PIC Microcontroller - Your Personal Introductory Course, 3rd Edسلطان

ومجموعة اخرى من المواقع:

www.instructables.com
http://www.engineersgarage.com/embedded/pic-microcontroller-projects
http://www.best-microcontroller-projects.com/pic-projects.html
http://www.vlsiacademy.org/video-library.html
www.eletorial.com

وإن كنا قد وصلنا إلى نهاية هذا الكتاب إلا أننا لم نصل إلى نهاية المجال نفسه فهذا هوا المجلد الأول فقط في مجال البك ميكروكنترولر ... وفيما يلي مجموعة من العناوين المتضمنة في المجلد الثاني بإذن الله:

- التحكم عن بعد باستخدام الريموت كنترول.
- توصيل الميكرو بالإنترنت.
- التعامل مع الذاكرة EEPROM.
- كيفية عمل بروجرامر للميكرو 16F877A.
- التحكم في الـ Stepper motor.
- المقاطعات والتايمر.

وغيرها من المواضيع المتقدمة الأخرى ... والله الموفق المستعان.

المراجع ...

١) كتاب احترف برمجة الميكروكنترولر للمهندس احمد سمير فايد.

٢) موقع القرية الإلكترونية.

3) http://www.allaboutcircuits.com/vol_4/chpt_13/1.html

4) http://www.mikroe.com/chapters/view/74/pic-basic-book-chapter-1-world-of-microcontrollers/

5) http://www.microcontrollerboard.com/pic_memory_organization.html

6) http://www.aliexpress.com/item-img/10-inch-88-88-Digital-LED-Seven-Segment-Time-and-Temperature-Display-Blue-Color-RF-Remote/642679401.html#

7) http://www.thelearningpit.com/lp/doc/7seg/7seg.html

8) http://quickstartkitforarduino.blogspot.com/2012/05/simple-labs-quick-start-kit-for-arduino_7820.html

9) http://www.8051projects.net/keypad-interfacing/introduction.php

10) http://arduino.cc/playground/Main/KeypadTutorial

11) http://www.microcontrollerboard.com/pic_serial_communication.html