# الأكاديمية العربية الدولية

## الأكاديمية العربية الدولية

## المقررات الجامعية

www.aiacademy.info | care@aiacademy.info

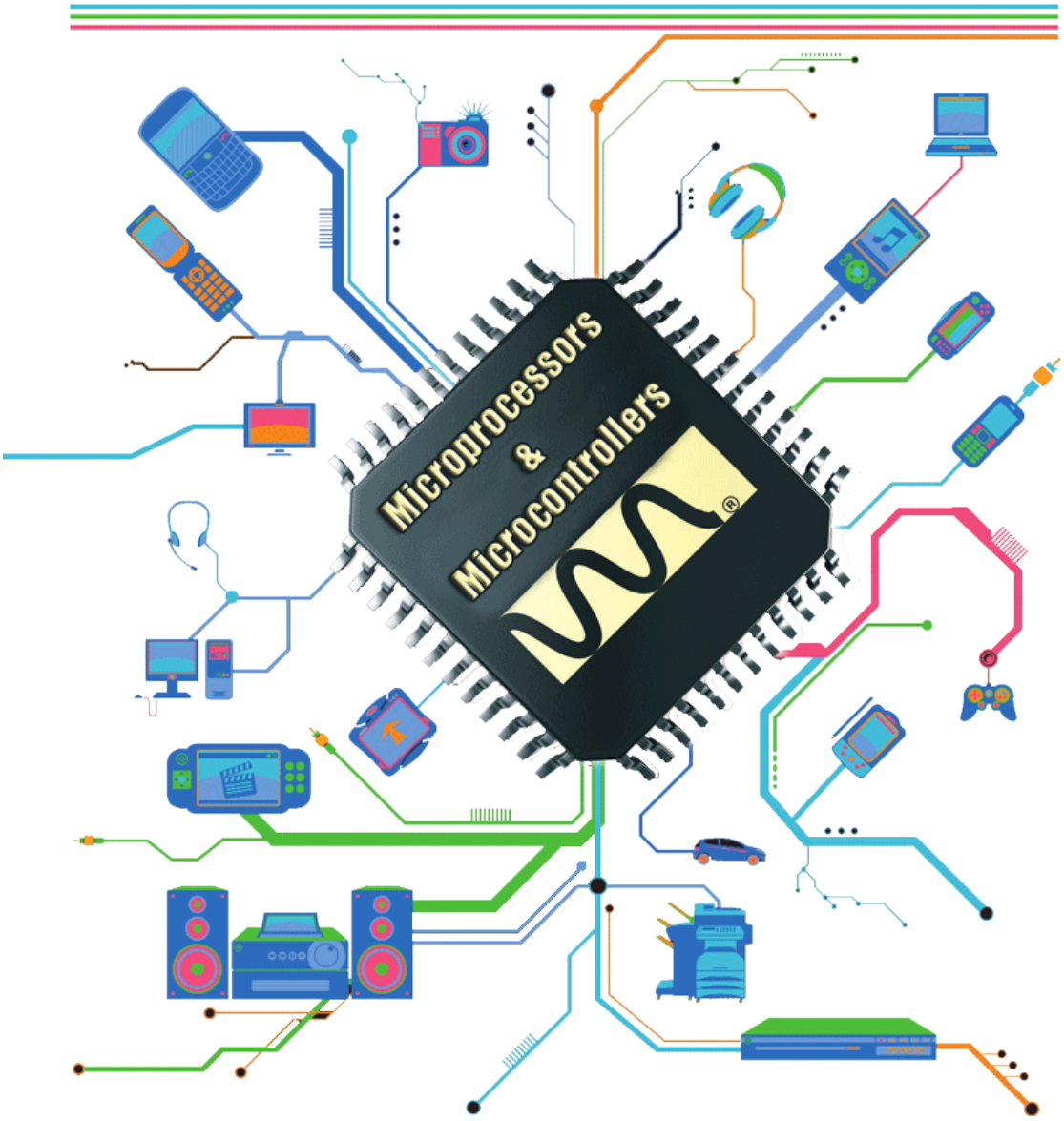
جامعـــــــــــة حلـــــــــــب
كلية الهندسة الكهربائية والإلكترونية
قسـم هندسـة التحكـم والأتمتـة

# ﴿الجلسات العملية لمادة المعالجات والمتحكمات المصغرة﴾

# Microprocessors & Microcontrollers Lab Sessions

## السنة الثالثة | قسم اتصالات

## ﴿الجلسة العملية الأولى﴾




م. وليد بليد

Copyright © 2012 Walid Balid - All rights reserved.

Wednesday, February 22, 2012

# ﴿الجلسة العملية الأولى﴾

## نظرة عامة (Overview):

هـذه الجلسـة تقـدم مـدخلاً هامـاً إلى الأنظمـة المدمجـة وموضـوعاتها وتطبيقاتهـا ومراحـل وأسـس تصـميمها. حيـث يـتم التطـرق فيهـا إلى بنيـة النظـام المـدمج، العوامـل المـؤثرة في تصـميمه، تطبيقاتـه الصـناعية وحلولـه التكنولوجيـة. ثم نتفـرع إلى أصـناف الـدارات المتكاملـة الرقميـة ونفصـل في المتحكمـات والمعالجـات وبـنى مسـجلات التعليمـات ومعيارية تصميم البنية.

## 1-1 تمهيد (Preface):

إن من أهم وأشمل الفروع المعرفية التكنولوجية الهندسية التي تتسابق المجتمعات المتقدمة ومخابر الأبحاث في الجامعات إلى تطويرها، وتشغل حياتنا اليومية بتطبيقاتها المتعددة دون أن ندرك ذلك، هي ما يطلق عليه بـ Embedded Systems.

في الحقيقة تتعدد الترجمات العربية لمصطلح الـ(Embedded Systems (ESs، فيطلق عليها: " الأنظمة المدمجة"، "والأنظمة المضمنة"، "والأنظمة المطمورة"، إلى ما هنالك من ترجمات أخرى، غير أنها جميعاً لا تقارب المعنى الحقيقي، وكيف لا؟! ولا يوجد إلى الآن تعريف معتمد باللغة الإنكليزية للESs، إذ أنها تعرف وفقاً للتطبيق الذي تشغله، وهناك آلاف التطبيقات التي قبلها النابض هو نظام مدمج.

الانطلاقة الأولى نظرياً كانت مع ظهور أول حاسب مصغر (12 bit PDP-8 Minicomputer) في عام1965، حيث أطلق عليه مصطلح الـEmbedded Computer، وتلاه ظهور أول معالج مصغر (Intel.4004،4-bit ) في عام 1971، إلا أن المفهوم كان بعيداً جداً عن المضمون الذي يحمله المصطلح، حتى عام 1977 وظهور أول متحكم مصغر (Intel.8048)، وعام 1979 وظهور أول معالج إشارة رقمية (Bell Labs' DSP-1)، ثم كانت الثورة الأولى لظهور مصفوفات البوابات الحقلية القابلة للبرمجة (FPGAs) في عام 1984. في عام 1988 ظهر مصطلح ESs في العدد الأول لمجلة "Embedded Systems Programming".

سابقاً كان استخدام الأنظمة المدمجة (ES's) مقتصراً على التطبيقات العسكرية وأبحاث الفضاء، واليوم تستخدم هذه الأنظمة في جميع الميادين الهندسية، مثل: الأجهزة الكهربائية والإلكترونية المنزلية، أجهزة الاتصالات، الأتمتة الصناعية، صناعة السيارات، أنظمة التحكم الرقمي، الروبوتات، التطبيقات العسكرية وأبحاث الفضاء، والعديد مما لا ينتهي ذكره من التطبيقات، إذ أنها غدت نواةً لـ99.99% من التطبيقات والأجهزة الإلكترونية، وهذا ما يجعلها محوراً أساسياً للبحث والتطوير.

مؤخراً، تعتبر دراسة تصميم الأنظمة المدمجة (ES's) من أهم المقررات الدراسية في الكليات الهندسية عالمياً، حيث تعطى الاهتمام الأكبر في مراحل مبكرة، ويُؤسَّس لها من السنة الدراسية الأولى، وتوظف معظم الأبحاث الجامعية في تطوير الصناعة وإيجاد الحلول التكنولوجية.

## 2-1 مقدمة (Introduction):

في عام 1969 طلبت شركة Busicom اليابانية من شركة Intel تصنيع مجموعة دارات تكاملية خاصة لإحدى آلاتها الحاسبة الجديدة. في عام 1971 كانت استجابة شركة Intel بتصنيع المعالج 4004 والذي هو أول رقاقة معالج يستخدم شريحة واحدة (Single Chip)، وبالتالي بدلاً من تصميم نظام لكل نموذج آلة حاسبة جديد، اقترحت Intel معالجاً ذا أغراضٍ عامةٍ يمكن أن يستخدم في أي نموذج من الآلات الحاسبة.

المعالج 4004 صمم لينفذ مجموعة من التعليمات البرمجية المخزنة في شريحة ذاكرة خارجية، وبالتالي يكفي تغيير برنامج الذاكرة الخارجية ليتناسب مع نموذج الآلة الحاسبة وميزاتها. هذا المعالج لقي نجاحاً باهراً، واستخدم على أصعدة عدة لعقد من الزمن، حيث – وللمرة الأولى – أصبح من الممكن بناء نظام معقد نسبياً باستخدام شريحة واحدة. التطبيقات المتقدمة الأولى في مجال الأنظمة المدمجة تضمنت: مسابر فضائية بغير ملاحين، إشارات المرور المتحكمَ بها حاسوبياً، أنظمة التحكم بالطائرات.

في الثمانينيات والتسعينيات كانت بداية عصر انتشار المعالجات الدقيقة (Microprocessors)، حيث انتشر استخدام المعالجات والمتحكمات المصغرة في معظم التطبيقات والأجهزة الإلكترونية الموجودة في حياتنا اليومية – في المطبخ (الميكروويف، آلة تحضير القهوة..)، في غرفة المعيشة (أجهزة العرض والتحكم والصوت والتكييف..)، في المكتب (الهاتف، الفاكس، الطابعة، آلة عد النقود..)، وجميع هذه التطبيقات هي أنظمة مدمجة (ESs).

العقد الأخير شهد تطوراً كبيراً وسّع آفاقاً جديدة ذات إمكانيات واعدة في تطبيقات الأنظمة المدمجة والتي منها: أنظمة التحكم عن بعد والتي تستخدم في المنازل الذكية، أنظمة الأكياس الهوائية الذكية في السيارات، أجهزة المراقبة الطبية الذكية التي تُعْلم الطبيب بالحالة الفيزيولوجية والمستويات الحرجة للمريض، أنظمة الملاحة والتوجيه في السيارات.

اليوم، يستخدم أكثر من 6-Bilion معالج/متحكم مصغر في كل عام في تطبيقات الأنظمة المدمجة، في حين أن 2% فقط من هذه المعالجات تستخدم في الحواسب الشخصية والمحمولة، وتشير الإحصاءات إلى أن عدد الأنظمة المدمجة يزداد بشكل متسارع، وأن الطلب متزايد على المهندسين اللذين يمتلكون مهارات تصميم الأنظمة المدمجة المستقبلية.

## 3-1 تعريف النظام المدمج (What is an Embedded System?):

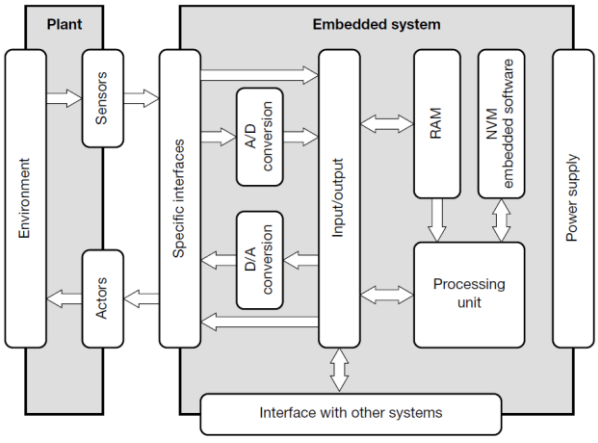
إن مصطلح الـ"Embedded system" هو أحد المصطلحات الشاملة التي لا تعبر بالضرورة عن معنى محدد لتوصيفها، فهي تغطي طيفاً واسعاً من التطبيقات والأنظمة، نذكر منها: الأجهزة الخلوية، أنظمة التحكم بالسكك الحديدية، أنظمة التوجيه والمراقبة العسكرية، التجهيزات الكهربائية والإلكترونية المنزلية والمكتبية...

تعرّف الأنظمة المدمجة على أنها: نظام مخصص لأداء وظيفة محددة يحوي على كيان صلب (HW) وبرمجية خاصة – برنامج عمل المعالج – (SW) إضافة إلى أجزاء أخرى (ميكانيكية، إلكترونية).

غالباً تعتبر العناصر القابلة للبرمجة القلب النابض في الأنظمة المدمجة، مثل: المتحكِّم المصغر (MCU)، المعالج المصغر (MPU)، المصفوفات الحقلية القابلة للبرمجة (FPGAs). عشرات الملايين من هذه العناصر تستخدم يومياً في الأنظمة المدمجة التي تغطي معظم التطبيقات المحيطة بنا، وتساهم في تحضير طعامنا دون أن ننتبه إلى ذلك.

على نحو خاص فإن النظام المدمج يشكل جزءاً أو عنصراً من نظام أكبر، مثاله: السيارات والحافلات الحديثة التي تحوي على العديد من وحدات الأنظمة المدمجة والتي منها: نظام مدمج مسؤول عن منع الانزلاق عند الكبح (ABS)، نظام مدمج مسؤول عن لوحة العدادات (Dashboard)، نظام آخر مسؤول عن التوجيه الملاحي (GPS)... حتى أنه في بعض السيارات الفاخرة (مثل: PMW) وصل عدد المعالجات إلى أكثر من 100 معالج يوصل من خلالها أكثر من 3000 حساس، مرتبطة عبر شبكة CAN.

رغم طيف التطبيقات الواسع جداً للأنظمة المدمجة، إلا أن لها ميزات مشتركة فيما بينها، وهي أنها تتفاعل مع العالم الخارجي، وتتحكم بالأجهزة المرتبطة. الشكل1 يبين مخططاً صندوقياً عاماً للمكونات الأساسية التي تشترك فيها جميع الأنظمة المدمجة. إن عملية التخاطب بين النظام المدمج والعالم الخارجي، هي من خلال قراءة إشارات الحساسات الموصولة إلى أقطاب الدخل، ومن ثم تقوم وحدة المعالجة المركزية (Processing Unit) باستخدام الذاكرة RAM بمعالجتها بعد تحويلها إلى إشارات رقمية عن طريق وحدة التبديلADC. يتم إصدار نتائج المعالجة كإشارات تحكم رقمية على أقطاب الخرج الرقمية، أو إشارات تحكم تشابهية عن طريق وحدة التبديل DAC. ثم يتم ترجمة (Compile) برنامج النظام المدمج (ES.SW) من أجل معالج محدد، ويتم تخزين البرنامج في ذاكرة دائمة (NVM) تدعى بالذاكرة ROM.

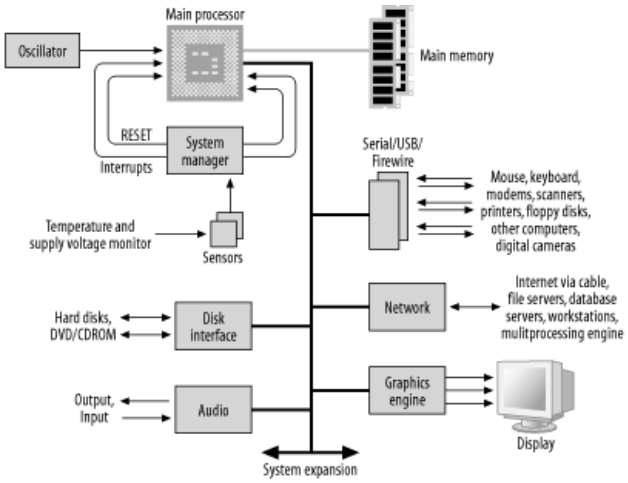



الشكل1 المكونات العامة للأنظمة المدمجة

### 4-1 بنية النظام المدمج (Embedded System Architecture):

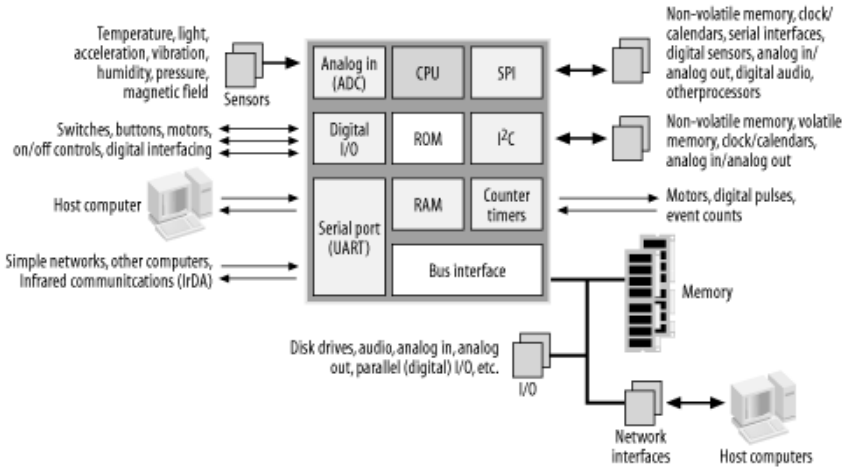
تتميز الحواسب العامة (PC: Personal Computers) بأنها تمتلك ذاكرة كبيرة تحوي على نظام التشغيل والتطبيقات البرمجية والبيانات، بالإضافة إلى إمكانية وصل وحدات تخزين ذات سعة كبيرة مثل: الأقراص الصلبة والرقمية. كذلك تمتاز بأنها تمتلك مجموعة متنوعة من أجهزة الإدخال (لوحة المفاتيح، الفأرة، مدخل صوتي) والإخراج (الشاشة، مخرج صوتي) إضافة إلى وحدات اتصال محيطية (الطابعة،

الفاكس، الشبكة، الماسح الضوئي، ...). إن وجود هذه الميزات يتطلب وجود معالج ذي أداء عالٍ وسرعة كبيرة، والذي ينتج عنه استهلاك كبير للتغذية، كما أن حجم النظام سيكون كبيراً جداً وسعره مرتفع جداً. الشكل2 يبين المخطط الصندوقي للحواسب العامة (PC).




الشكل2 مخطط بنية الحواسب العامة PC

على خلاف الحواسب العامة، فإن الأنظمة المدمجة تستخدم المتحكمات أو المعالجات المصغرة، والتي تمتاز بأن وحدة المعالجة المركزية سوف تكون مدمجة مع جميع المحيطيات والذواكر على شريحة واحدة. يبين الشكل3 المخطط الصندوقي العام لبنية الأنظمة المدمجة.




الشكل3 المخطط الصندوقي العام للأنظمة المدمجة

تمتلك معظم المتحكمات المصغرة الوحدات المحيطية الرئيسية التالية: وحدة معالجة مركزية (CPU)، ذاكرة برنامج (ROM)، ذاكرة معطيات دائمة (EPROM)، ذاكرة عشوائية (RAM)، أقطاب الدخل والخرج (I/O)، وحدات التوقيت والعد (T/C)، وحدات اتصال تسلسلي ($I^2C$، SPI، UART). كما أن بعض المتحكمات المصغرة المتقدمة تمتلك نوافذ اتصال تسلسلي عالية السرعة مثل: CAN، USB، Ethernet. الجدول1 يبين بعض أوجه الاختلاف العامة بين خصائص الأنظمة المدمجة والحواسب الشخصية.

| الأنظمة المدمجة (Embedded Systems) | الحواسب العامة (PC Computers) |
|---|---|
| مكرسة لمهام محددة. | تستخدم في أغراض عامة. |
| تملك مجموعة واسعة جداً من المعالجات تبلغ 140 عائلة مصنعة من قبل أكثر من 40 شركة متخصصة | محدودة في عائلتين من المعالجات (AMD،Intel ). |
| كلفة النظام تعتبر من العوامل الأساسية. | لا يوجد اعتبار للكلفة. |
| يوجد قيود لشروط العمل في الزمن الحقيقي (RTS). | لا يشترط عملها في الزمن الحقيقي. |
| أنظمة التشغيل تعمل في الزمن الحقيقي (RTOS). | أنظمة التشغيل لا تعمل في الزمن الحقيقي. |
| إن نتائج فشل النظام خطيرة جداً ويمكن أن تكون قاتلةً. | فشل النظام لا يشكل خطراً. |
| يوجد قيود لاستهلاك الطاقة الكهربائية. | لا يوجد قيود حول استهلاك الطاقة. |
| يجب أن تعمل في ظروف بيئية قاسية أحياناً. | غالباً توجد في ظروف العمل الطبيعي. |
| مصادر النظام محدودة. | مصادر النظام لانهائية (PCI، ISA، AGP، LPT، ...). |
| يتم تخزين كامل برنامج المعالج في ذاكرة ROM. | يتم تخزين نظام التشغيل والبرامج الخدمية في HDD. |
| تتطلب أدوات وطرقاً خاصة ليتم تصميمها بكفاءة. | الأدوات المستخدمة عامة. |
| مزودة بدارات Debugger مخصصة على نفس الشريحة. | لا تملك أي دارات ذات وظائف تتبع الأخطاء. |

الجدول10 مقارنة بين خصائص الأنظمة المدمجة والحواسب العامة

## 5-1 العوامل المؤثرة في تصميم الأنظمة المدمجة (Requirements Affect in ESs Design):

عند تصميم أي نظام مدمج فإنه يجب مراعاة مجموعة من المتطلبات والاعتبارات يتم تحديدها في الدرجة الأولى وفقاً لعامل الكلفة المطلوب، على سبيل المثال: إذا تتطلب إنتاج نظام تحكم مدمج بكلفة لا تتجاوز 1000 ليرة سورية؛ فإنه ربما من الضروري الاستغناء عن بعض الميزات الكمالية للوصول إلى الكلفة المطلوبة. الاعتبارات الأساسية في تصميم الأنظمة المدمجة هي:

1. سعة المعالجة (Processing Power): وهي عدد التعليمات التي يمكن تنفيذها خلال ثانية واحدة (MIPS)، وبازديادها تزداد سعة (قوة) المعالجة للمعالج.
2. عرض الناقل الداخلي (Data-Bus): وهي عرض ناقل البيانات بين وحدة المعالجة والذاكرة ويتراوح 4-bit ~ 64-bit، وبازدياد عرض الناقل تزداد سرعة تناقل البيانات بين المعالج والذاكرة.
3. حجم الذاكرة (Memory Space): وهي المساحة المطلوبة لتخزين برنامج تنفيذ التعليمات (ROM) والبيانات (المعطيات) التي يتم معالجتها آنياً (RAM). عموماً، فإن مساحة الذاكرة المطلوبة تتعلق بالمعالج المستخدم والميزات المحيطية المترافقة معه وحجم البرنامج.

4. استهلاك الطاقة (Power Consumption): هي من أهم الاعتبارات خصوصاً في الأجهزة النقالة التي تعمل على المدخرات، وتستعمل وحدة القياس mW/MIPS لتحديد كمية الطاقة المطلوبة تبعاً لسعة المعالجة، حيث أنه بازدياد سعة المعالجة تزداد كمية الطاقة المطلوبة لعمل المعالج. عملياً، فإن الأنظمة التي تستهلك طاقة منخفضة تتميز بخصائص مرغوبة جداً مثل: حرارة أقل، وزن أقل، حجم أصغر، تصميم ميكانيكي أبسط. لذلك تستخدم المعالجات متعددة النوى ( Multi-core Processors) في الأنظمة المدمجة التي تتطلب نظاماً منخفض الاستهلاك والحجم وذا سعة معالجة عالية.

5. كلفة التطوير (Development Cost): هي كلفة تصميم الكيان الصلب (ES.HW) والبرمجيات المترافقة (ES.SW)، وتعرف أيضاً بالمصطلح NRE (Non-Recurring Engineering)، وهي كلفة ثابتة تدفع لمرة واحدة فقط أثناء مرحلة تصميم النظام – هذه الكلفة يتم توزيعها على عدد قطع الإنتاج.

6. كمية الإنتاج (Number of Units): إن الموازنة بين كلفة الإنتاج وكلفة التطوير تتعلق مباشرة بكمية الإنتاج المطلوبة، إذ يتم توزيع الكلفة الثابتة على عدد العناصر المطلوبة. أما من أجل تصميم ذي كمية محدودة من القطع؛ فإن كلفة التطوير لمثل هذا النظام ستكون كبيرة جداً.

7. حياة المنتج (Lifetime): وهو العمر الافتراضي المتوقع لبقاء المنتج في الاستخدام الفعال. إن هذا الاعتبار يؤثر مباشرة في جميع قرارات التصميم انطلاقاً من اختيار عناصر الكيان الصلب وصولاً إلى كلفة التطوير.

8. الوثوقية (Reliability): وهي مقدرة النظام على الاستجابة في مختلف الظروف، وتتناسب الوثوقية طرداً مع كلفة النظام.

إضافةً إلى هذه المتطلبات الأساسية الثمانية، فإن لكل نظام مدمج متطلبات وظيفية أخرى خاصة تتعلق بهوية النظام وتوظيفه (مايكروويف، منظم دقات القلب، نظام الطيران الآلي، نظام التوجيه الملاحي...)، ومن هذه المتطلبات: المعالجة في الزمن الحقيقي (Real-time Processing).

### 6-1 صناعة الأنظمة المدمجة (Embedded System Industry):

في السنوات الأخيرة أصبح قطاع صناعة الأنظمة المدمجة في العديد من البلدان الصناعية القطاع الأكثر ازدهاراً وتطوراً، حيث تعتبر صناعة الأنظمة المدمجة عالمياً الجزء الأكبر والأسرع نمواً، وخصوصاً صناعة المتحكمات المصغرة التي تشكل تقريباً 99.99% من الناتج العالمي من المعالجات (MPU،MCU ) التي يتم إنتاجها سنوياً.

إن سبب هذا التكاثر المتزايد يعود إلى أن عدد المعالجات المستخدمة في الحواسب الشخصية يعتبر صغيراً جداً مقارنةً مع المعالجات المستخدمة في الأنظمة المدمجة. التقرير الأخير يشير إلى أن المنزل الواحد يحوي على الأقل 100~40 متحكم مصغر، في حين يمكن أن يوجد ثلاثة أو أقل في الحاسب الشخصي.

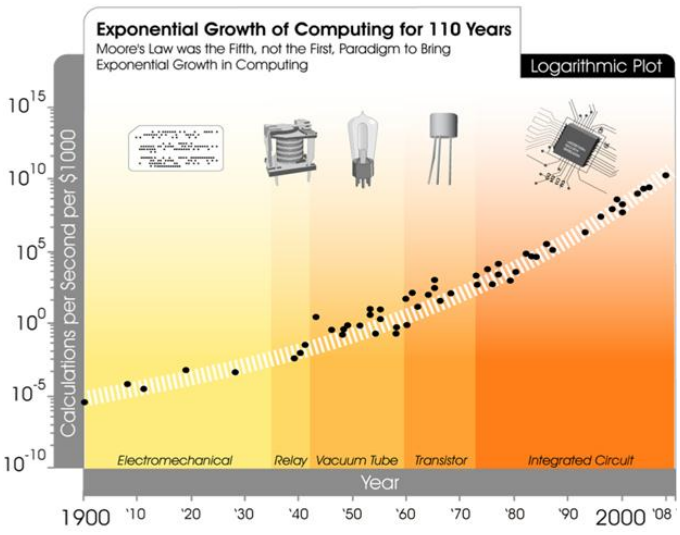
المتحكمات المصغرة يمكن أن توجد في جميع التجهيزات المنزلية، مثلاً: التلفاز، المشغل الرقمي، ألعاب الأطفال، الموقد. إضافةً إلى ذلك فإن الأنظمة المدمجة تمثل القطاع الرئيسيَّ في سوق الأنظمة الرقمية، والحلول التكنولوجية لمعظم تطبيقاته الصناعية، والتي منها: وسائل النقل (Automotive)، إلكترونيات المستهلك (Consumer Electronics)، الأتمتة الصناعية (Industrial Automation)، التطبيقات العسكرية (Military)، تناقل البيانات (Data-Transmission)، الاتصالات (Communication)، الفضاء (Aerospace).

حالياً، أكثر من 98% من متحكمات 8/32-bit تستخدم في الأنظمة المدمجة. طبقاً للدراسة الإحصائية التي نشرتها شركة SEMICO في عام 2006 فإن 55% من المتحكمات التي تباع حول العالم هي 8-bit، وأكثر من 4-billion متحكم 8-bit بيع في عام 2006.

المحللون الاقتصاديون يتوقعون أنه مع انطلاقة عام 2010 فإن أكثر من 90% من البرامج التي تم تطويرها ستكون مخصصة للأنظمة المدمجة، كما أنّ عدد مبرمجي الأنظمة المدمجة سيزداد بمقدار عشرة أضعاف مقارنةً مع مبرمجي الأنظمة الأخرى. على الرغم من هذه الإحصاءات؛ فإن معظم مناهج هندسة الحاسبات والتحكم في العديد من الجامعات الغربية على وجه عام، وجامعاتنا المحلية على وجه التخصيص، ما تزال تعلم مهارات البرمجة والتصميم المتعلقة بلغات برمجة الحواسب العامة فقط، بدلاً من برمجة الأنظمة المدمجة الأكثر تخصصاً.

## 7-1 الحلول التكنولوجية للأنظمة المدمجة (E.Systems Technologies & Approaches):

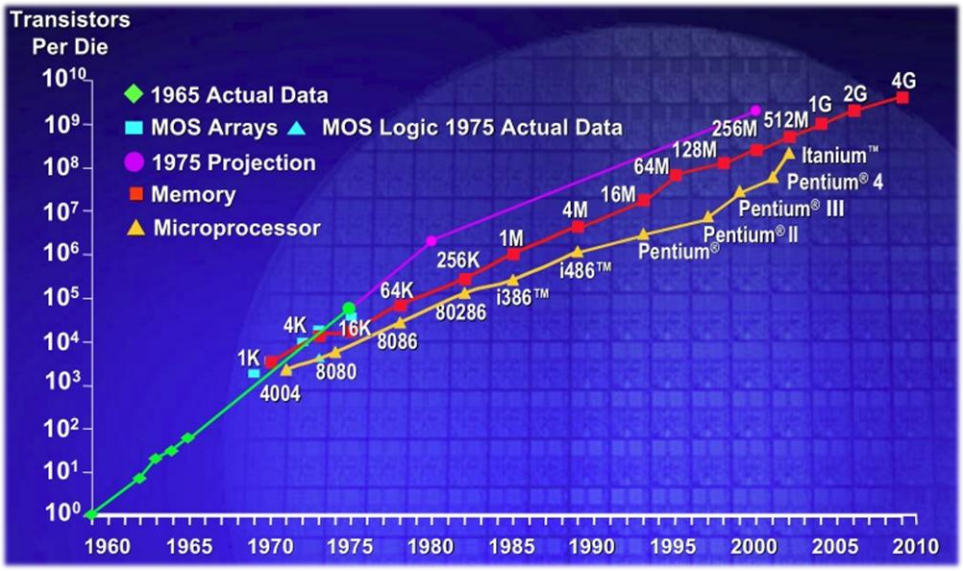
قبيل ظهور العناصر الإلكترونية كانت الحاسبات تبنى باستخدام عناصر كهروميكانيكية حتى العقد الخامس من القرن التاسع عشر وظهور الصمامات الإلكترونية والدارات المنطقية – التي تبنى من الترانزستورات والمقاومات (RTL)، ثم تلاه ظهور الدارات المتكاملة، وأصبح بالإمكان تصنيع دارة منطقية على شريحة سيليكونية واحدة. الشكل4 يبين مخططاً زمنياً للتطور التكنولوجي للحاسبات.




الشكل4 التطور الزمني لتكنولوجيا الحاسبات

في عام 1965 لاحظ Gordon Moore مدير شركة Intel أن تكنولوجيا الدارات المتكاملة تتطور بمعدل مذهل بحيث أن عدد الترانزستورات التي يمكن أن توضع على نحو رخيص (بدون أي كلفة زائدة) على دارة متكاملة – أي أن تعقيد الدارات المتكاملة مع اعتبار

الكلفة الأخفض للعناصر – يتضاعف كل سنتين تقريباً. إن قابلية ودرجة تطور العديد من الأجهزة الإلكترونية الرقمية (سعة وسرعة المعالجة، حجم الذاكرة، ...) يرتبط بشكل وثيق بهذا القانون حيث أن هذه الملاحظة أدت على نحو كبير جداً إلى زيادة فائدة استخدام تكنولوجيا الإلكترونيات تقريباً في جميع قطاعات الاقتصاد العالمي وقد تم تسمية هذا القانون علمياً باسم "قانون مور" (Moore's Law) نسبةً إلى Gordon E. Moore الذي قدم هذا القانون ومنذ ذاك الوقت يعتبر هذا القانون محور التخطيط والتوجيه طويل الأمد في وضع أهداف البحث والتطوير في صناعة أنصاف النواقل ويتوقع أن يستمر العمل بهذا القانون إلى ما بعد عام 2020. الشكل5 يبين مخططاً لوغارتمياً لعدد الترانزستورات على شريحة واحدة في مراحل زمنية متعددة (1960-2010) ويلاحظ بأن معدل التزايد يتضاعف كل سنتين.




الشكل5 المنحني الزمني لازدياد عدد الترانزستورات على شريحة متكاملة

تعتبر مسألة تحديد التقنية المستخدمة في تصميم الأنظمة المدمجة من الأمور الهامة والأساسية في المراحل المبكرة للتصميم وهي تستند إلى الوظائف الأساسية المطلوبة من النظام، إذ هناك العديد من الخيارات المتاحة لتصميم الأنظمة المدمجة بدءاً من المعالجات المصغرة (MPUs) والمتحكمات المصغرة (MCUs) والعناصر المنطقية القابلة للبرمجة (PLDs) والدارات المتكاملة ذات التطبيقات الخاصة (ASICs)، كما أن الاختيار بين هذه العناصر المختلفة يعتمد على متطلبات النظام المطلوب تصميمه أكثر من كونه معتمداً على اعتبارات وميول شخصية للمصمم.

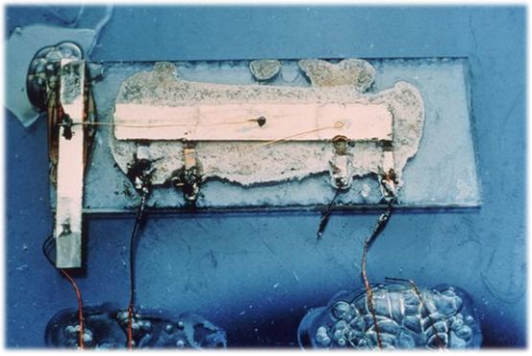
فإذا كان المطلوب نظاماً قابلاً للبرمجة لتنفيذ خوارزميات ذات عمليات حسابية معقدة، فإن الاختيار الأمثل لهذه التطبيقات هي معالجات الإشارة الرقمية (DSP)؛ أما إذا لم يكن لاعتبارات السرعة أهمية بالغة في عمل النظام وكانت التكلفة وظروف العمل لا تسمح باستخدام شرائح متطورة، فيكون استخدام المعالجات المصغرة (MCUs | MPUs) مثالياً لهذه الحالة؛ وفي حال كان النظام يتطلب مستويات أداء عالية، وسرعات معالجة عالية جداً، وبنية تنفيذ تفرعية، فإن مصفوفات البوابات القابلة للبرمجة حقلياً (FPGAs) تقدم الأداء المطلوب وتتيح للمصمم مرونة كبيرة في تصميم خوارزمية النظام، حيث أن بنية الـFPGAs تتيح إمكانية العمل المتزامن (Parallelism) لوظائف النظام، وهذا ما لا تستطيع تأمينه المعالجات أو الحلول المتكاملة الأخرى التي تعتمد في تنفيذ خوارزميتها على العمل التسلسلي.

## 8-1 تطور صناعة أنصاف النواقل (Semiconductors Industry Evolution):

في أوائل العشرينيات من القرن العشرين كانت الأجهزة تعتمد على المفاتيح الميكانيكية (Electromechanical) مثل الـRelays في عمليات التحكم بالأجهزة. في أواخر عام 1920 ظهر الجيل الأول من العناصر الإلكترونية وعرفت بالصمامات المفرغة ( Vacuum Tubes)، مبينة على الشكل6، حيث استعملت في العديد من الأجهزة الإلكترونية في ذاك الوقت: الراديو والتلفاز وغيرها.

الشكل6 الصمامات المفرغة

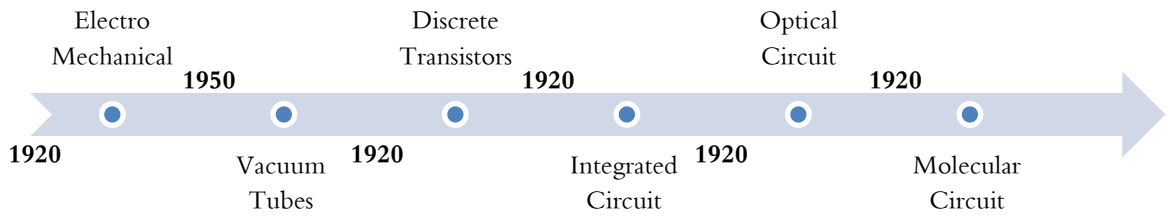
في خمسينيات القرن العشرين ظهرت الترانزستورات، وبدأت الأبحاث حول الدارات المتكاملة التي يمكن أن تحوي على عدة ترانزستورات، وكان هناك محاولات عديدة لبناء دارة متكاملة، وكانت التجربة الناجحة الأولى في عام 1958 حيث قام عالم الفيزياء Jack Kilby بتصميم أول دارة متكاملة – الشكل7.



الشكل7 أول دارة متكاملة

صناعة أنصاف النواقل ظهرت بوضوح في أوائل الستينيات، وبدأت بالتطور بشكل متسارع منذ ذاك الوقت. الجيل الأول من الدارات المتكاملة عرف بـSSI (Small-scale Integration) حيث كانت الدارات المتكاملة تتكون من عدد قليل جداً من البوابات المنطقية (OR، NOR، AND، etc.) تتشكل من عشرات الترانزستورات. تلاه عصر الجيل الثاني من الدارات المتكاملة في أواسط الستينيات وعرف بـMSI (Medium-scale Integration)، وقد تميز بزيادة كبيرة في عدد البوابات المنطقية على شريحة متكاملة وحيدة، حيث

استخدمت الشرائح في المؤقتات والعدادات والمسجلات، وتميزت بكلفة أخفض بكثير من سابقتها وإمكانية تصميم أنظمة أكثر تعقيداً. الجيل الثالث من الدارات المتكاملة ظهر في أواسط السبعينيات وعرف بـLSI (Large-scale Integration)، وفيه تم تضمين عشرات الآلاف من الترانزستورات على شريحة متكاملة وحيدة مثل الـcalculator chips،1K-bit RAMs ، وكذلك الجيل الأول من المعالجات المصغرة. الخطوة الأخيرة في عملية تطور الدارات المتكاملة كانت بظهور تقنية VLSI (Very-large-scale Integration) حيث أن عملية التطور بدأت في أوائل الثمانينيات من خلال دمج مئات الآلاف من الترانزستورات على شريحة واحدة، واستمرت لتصل في عام 2009 إلى عدة بلايين من الترانزستورات على شريحة واحدة. إن المعالجات متعددة النوى التي ظهرت مؤخراً ذات عرض ناقل 64-bit، والتي يدمج معها ذاكرة وسيطة (cache memory) ووحدات حساب ومعالجة بالفاصلة العشرية تعتبر من الأجيال المتطورة للدارات المتكاملة VLSI. الشكل8 يبين مخططاً زمنياً لمراحل لتطور تقنيات تصنيع الدارات المتكاملة. الجدول2 يبين تصنيفاً لتقنيات تصنيع الدارات المتكاملة وفقاً لعدد الترانزستورات على شريحة واحدة أو عدد البوابات المنطقية والتطبيقات لكل جيل.



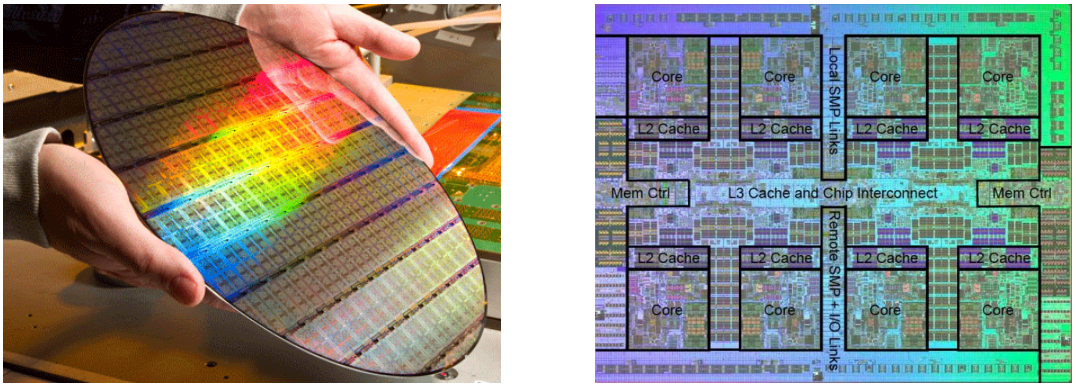

الشكل8 المراحل الزمنية لتطور تقنيات تصنيع الدارات المتكاملة

| Density | Date | Transistors | Gates | Application |
|---|---|---|---|---|
| SSI | 1961-1966 | 1~100 | 10 | AND، OR، NOT gates chips |
| MSI | 1966-1971 | 100~3000 | 100 | Decoder، encoder، multiplexer، counter |
| LSI | 1971-1979 | 3K~100K | 10،000 | Micro-controller، special-function chips |
| VLSI | 1980s | 100K~1M | 100،000 | Memory، special-function chips |
| ULSI | 1990s | > 1M | >100،000 | Memory، microprocessor chips |
| GSI | 2009s | 2M~2Bilion | >1M | Multi-core Processors |

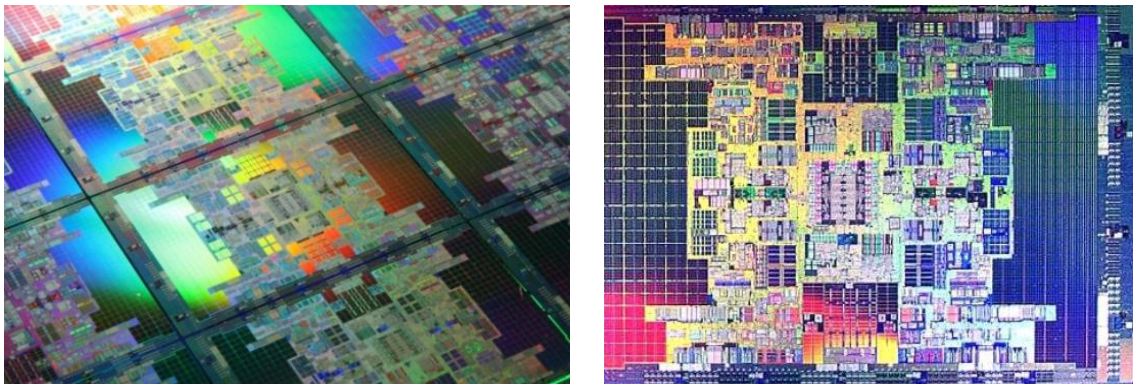
الجدول2 المراحل الزمنية لتطور تصنيع الدارات المتكاملة ومدى تعقيدها

مما يجب الإشارة إليه أن البعض (كما في اليابان) يستخدم مصطلح الـULSI اختصاراً لـUltra-large-scale Integration وكذلك المصطلح GSI اختصاراً Giga-scale Integration والذي يشير إلى الدارات المتكاملة التي تحوي على بلايين الترانزستورات، إلا أن المعظم وقف عند المصطلح VLSI وأدرج التقنيات المذكورة كفروع لتقنية الـVLSI، وإلا فإنه سوف يتوجب إيجاد مصطلحات متجددة بشكل دائم – على الأقل كل سنتين. الشكل9 يبين الشريحة السيليكونية للمعالج IBM Power-7 والذي يملك 8-core ويحوي على

1.2 بليون ترانزستور وعلى اليسار لوح الـWafer ويحوي مئات الشرائح السيليكونية قبل فصلها. الشكل10 يبين الشريحة السيليكونية ولوح الWafer للمعالج Intel Itanium والذي يملك 4-core ويحوي على 2.046 بليون ترانزستور.




الشكل9 الشريحة السيليكونية للمعالج IBM Power7 (8-core ويحوي على 1.2 بليون ترانزستور) ولوح الـ"Wafer"



الشكل10 الشريحة السيليكونية للمعالج Intel Itanium رباعي النوى ويحوي على2.046 بليون ترانزستور

## 9-1 تقنيات صناعة أنصاف النواقل (Semiconductors Industry Evolution):

تطورت تقنيات صناعة أنصاف النواقل بشكل كبير خلال العقود الثلاثة الماضية، وأصبحت تعتمد على تقنية الCMOS التي أمكنت من أن تصبح الترانزستورات أصغر حجماً، وبالتالي يمكن للدارات المتكاملة أن تحوي عدد ترانزستورات أكبر. الدارات المتكاملة الأولى كانت تستخدم الترانزستورات ثنائية القطبية (BJT)، وكانت غالبية الدارات المتكاملة تستخدم المنطق TTL أو المنطق ECL. على الرغم من أن تقنية الMOS تم اختراعها قبل الترانزستورات الثنائية، إلا أنه كان في البداية من الصعب جداً تصنيعها نظراً لمشكلة طبقة الأكسيد. في السبعينيات تم حل هذه المشكلة، وتم تطوير تقنية الـNMOS؛ في ذاك الوقت تطلب استخدام تقنية MOS عدد أقل من طبقات الـMask (أفلام تصنيع الطبقات على مستوى السليكون والأكسيد) وبالتالي كثافة أكبر واستهلاك طاقة أقل وسعر أرخص مقارنةً مع الدارات المتكاملة التي تعتمد تقنية التصنيع BJT.

مع بدايات الثمانينيات تم استبدال بوابات الترانزستورات المصنعة من الألمنيوم ببوابات الPolysilicon والتي أدت إلى تحسين كبير في تقنية CMOS حيث أنها مكنت من استخدام نوعين من الترانزستورات (PMOS،NMOS ) على نفس الشريحة السيليكونية، وبالتالي

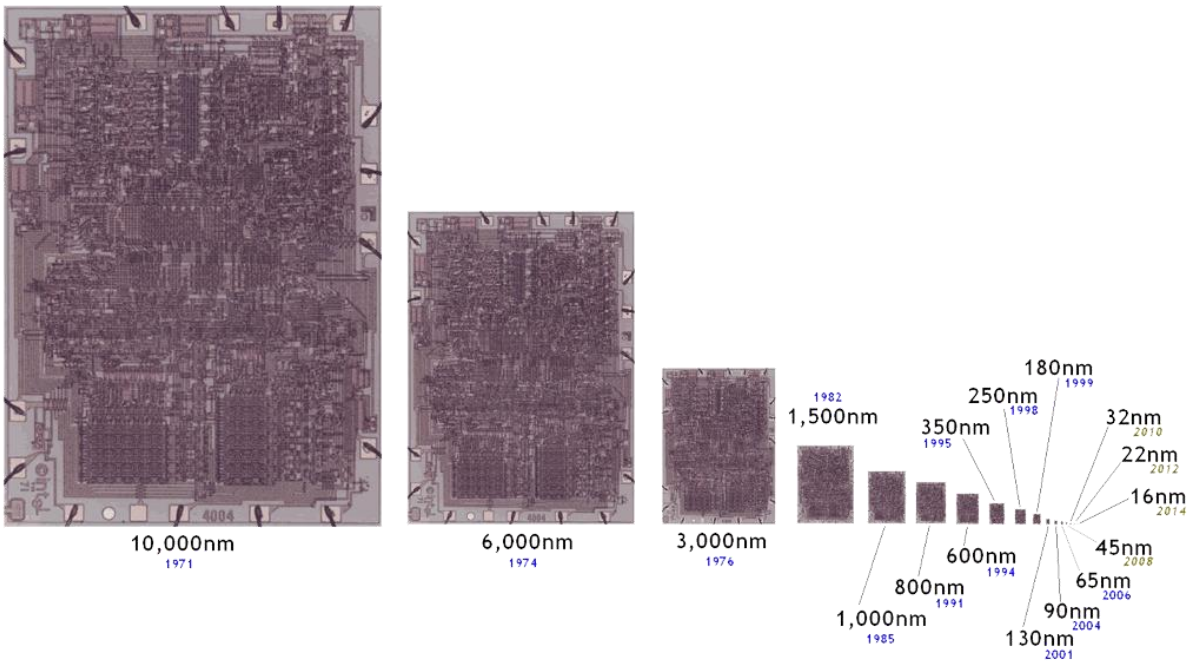
أصبحت عملية التصنيع أسهل، كما أن استهلاك الطاقة أصبح أخفض، وهذا جميعه ساعد في تصميم دارات متكاملة أصغر حجماً.

الجدول3 يبين أجيال التقنيات المستخدمة في بناء الدارات المتكاملة.

| Technology | Power Consumption | Speed | Packaging |
|---|---|---|---|
| RTL (BJT) | High | Low | Discrete |
| DTL (BJT) | High | Low | Discrete، SSI |
| TTL (BJT) | Medium | Medium | SSI، MSI |
| ECL (BJT) | High | High | SSI، MSI، LSI |
| pMOS (MOSFET) | Medium | Low | SSI، MSI |
| nMOS (MOSFET) | Medium | Medium | SSI، MSI، VLSI |
| CMOS (MOSFET) | Low | Medium | SSI، MSI، LSI، VLSI |
| GaAs (MOSFET) | High | High | SSI، MSI، LSI |

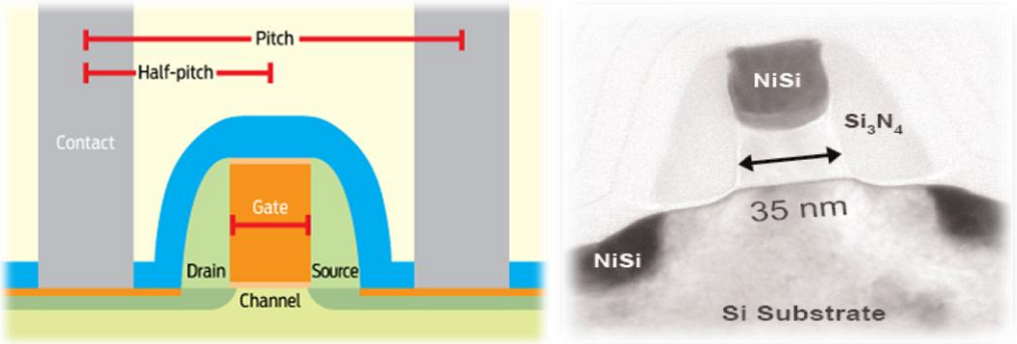
الجدول30 التقنيات المستخدمة في بناء الدارات المتكاملة

إن التطور الذي حصل خلال 40 عاماً – ابتداءً من العام 1971 وحتى عام 2010 – أدى إلى انتقال مستوى تقنية التصنيع من الحجم 10µm إلى الحجم 32nm للخلية الترانزستورية على المستوى السيليكوني، وبالتالي تضاعف عدد الترانزستورات على الشريحة الواحدة بحوالي 1000 مرة! الشكل11 يبين الشريحة السيليكونية للمعالج Intel-4004 والتي تم إنتاجها في عام 1970 على شريحة سيليكونية بمساحة 10um مقارنةً مع حجمها في عام 2010 بمساحة 32nm.




الشكل11 حجم الشريحة السيليكونية مع تطور تقنية التصنيع من العام 1970 وحتى 2014

إن متوسط نصف حجم الترانزستور على الشريحة السيليكونية يطلق عليه بـProcess Technology وهو الذي يتضاعف وفق قانون Moor كل سنتين. فمثلاً من أجل تقنية التصنيع 65nm فإن عشرات الآلاف من الترانزستورات يمكن أن تتسع في مساحة تعادل مساحة خلية دم حمراء – يمكن لعشر ملايين ترانزستور أن تتسع في مساحة تعادل $1mm^2$. الشكل12 يبين صورة ميكروية لترانزستور على الشريحة السيليكونية للمعالج Intel-Quad-core يعتمد تقنية 65nm ونجد أن متوسط نصف حجم المساحة هو 35nm.



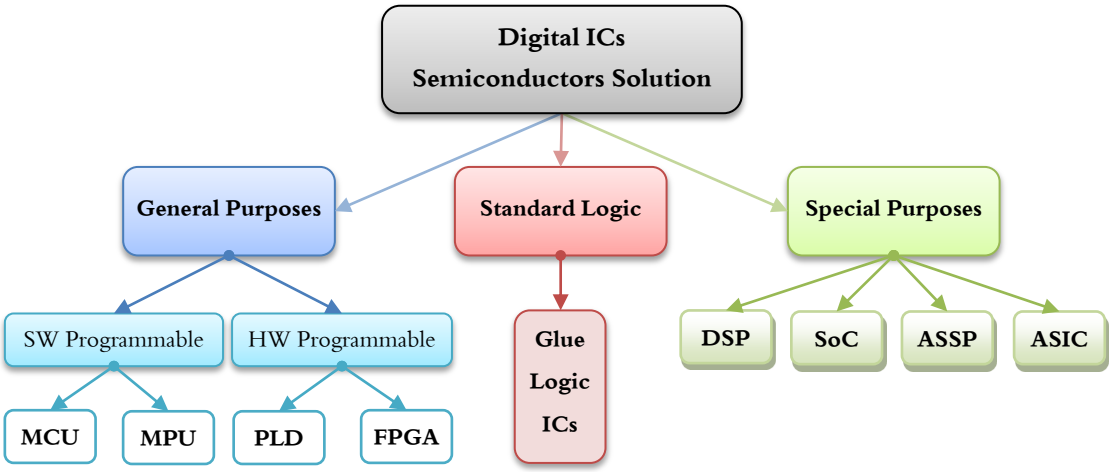

الشكل12 ترانزستور على مقطع شريحة سيليكونية يعتمد تقنية التصنيع 65nm

## 1-10 أصناف الدارات المتكاملة الرقمية (Digital Integrated Circuit Classes):

فيما يلي تفصيل مقتضب لفروع الأنظمة المدمجة وخواصها يوفر على الباحث والدارس الخوض في مئات المراجع.

بشكل عام، تصنف حلول الدارات المتكاملة الرقمية ضمن فروع رئيسية ثلاث:

1- الدارات المتكاملة القياسية (Standard Logic ICs).
2- الدارات المتكاملة ذات التطبيقات العامة (General Purposes ICs).
3- الدارات المتكاملة ذات التطبيقات الخاصة (Special Purposes ICs).




الشكل0 الفروع الرئيسة للدارات المتكاملة الرقمية

فمنها ما هو غير قابل للبرمجة، ومنها ما هو قابل للبرمجة على مستوى التعليمات البرمجية (Software)، ومنها ما هو قابل للبرمجة على مستوى الكيان الصلب (Hardware). الشكل13 يبين الفروع الرئيسة للدارات المتكاملة الرقمية. إن هذا التصنيف يمثل تصنيفاً عاماً إذ يمكن أن يوجد تصنيفات فرعية أخرى تصنف بأنها دارات متكاملة متخصصة أو عامة التطبيقات؛ فيما يلي نفصل في هذه الفروع.

### 1-10-1 الدارات المتكاملة القياسية (Standard Logic ICs):

وهي شرائح متكاملة ذات وظائف عامة تم تصميمها لوظائف محددة (Fixed Functionality) لا يمكن تغييرها وفقاً لنموذج قياسي عالمي، كما أنها لا تقوم بأي عمليات معالجة (لا تملك وحدة معالجة)؛ من خلال ربط العديد من هذه الدارات مع بعضها البعض يمكن الحصول على دارة وظيفية منطقية، كما أن أي تغيير في وظيفة الدارة يحتاج إلى إعادة ربط هذه الدارات بالكامل. مثالها: الدارات المتكاملة للبوابات المنطقية (AND،NAND، OR، NOR، NOT ) مثل شرائح العائلة 74xxxx (74HC595: Shift Register)، وشرائح العائلة 40xxxx (4018: Counter)، وشرائح التوقيت (NE555)، وغيرها من الشرائح القياسية التي تصنع من قبل العديد من الشركات.

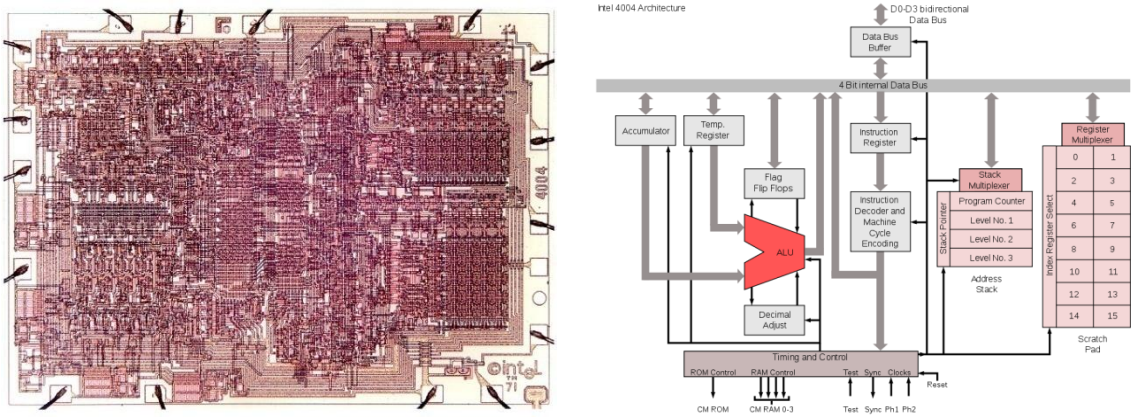
### 2-10-1 المعالجات المصغرة µP،MPU (Microprocessors):

وهي من فروع الدارات المتكاملة ذات الأغراض العامة (General Purposes ICs). تجمع المعالجات المصغرة كل وظائف وحدة المعالجة المركزية CPU في دارة متكاملة واحدة، ويتم برمجتها من أجل تطبيق خاص باستخدام لغات برمجية مخصصة لتطبيقات الأنظمة المدمجة (Embedded Systems Programming Languages). تصنف المعالجات المصغرة من حيث الاستخدام إلى نوعين رئيسيين:

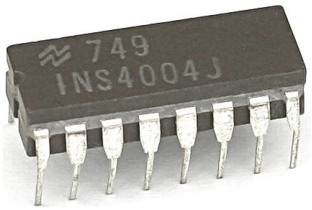
- معالجات الأغراض العامة.
- معالجات الأغراض الخاصة.

### 1-2-10-1 معالجات الأغراض العامة GPP (General Purpose Processor):

في عام 1970 ظهر المعالج Intel 4004 ذو ناقل بعرض 4-Bit واستخدم في تصميمه 2300 ترانزستور وقد تضمن ذاكرة RAM وذاكرة ROM ووحدة معالجة مركزية بتردد عمل 108KHz، استخدم هذا المعالج بشكل رئيسي في الآلات الحاسبة، ويعتبر أول معالج مصغر، وقد بلغ سعره آلاف الدولارات. في عام 1974 أعلنت شركة Intel عن أول معالج للأغراض العامة (GPP) وهو المعالج 8080 بعرض ناقل 8-Bit، واستخدم في تصميمه 4500 ترانزستور ووصلت سرعة تنفيذه إلى 290000 تعليمة في الثانية عند تردد عمل 2MHz، وتضمن أيضاً 64KB من الذاكرة المعنونة وأصبح المعالج 8080 معياراً صناعياً وبلغ سعره 395$ واستخدم في بناء أول حاسب شخصي. منذ ذاك الحين تطورت معالجات الأغراض العامة – سرعةً وأداءً – فظهرت المعالجات ذات عرض الناقل 16-Bit( Intel 8086)، ومعالجات 32-bit (Intel/AMD x86)، ومعالجات 64-Bit (Intel/AMD x64)، كما ظهرت مؤخراً المعالجات متعددة النوى (multi-core) لتسيطر على مستقبل صناعة المعالجات. الشكل14 يبين البنية الداخلية (على اليمن) والخريطة السيليكونية (على اليسار) للمعالج Intel® 4004. الشكل15 يبين شريحة المعالج Intel® 4004.

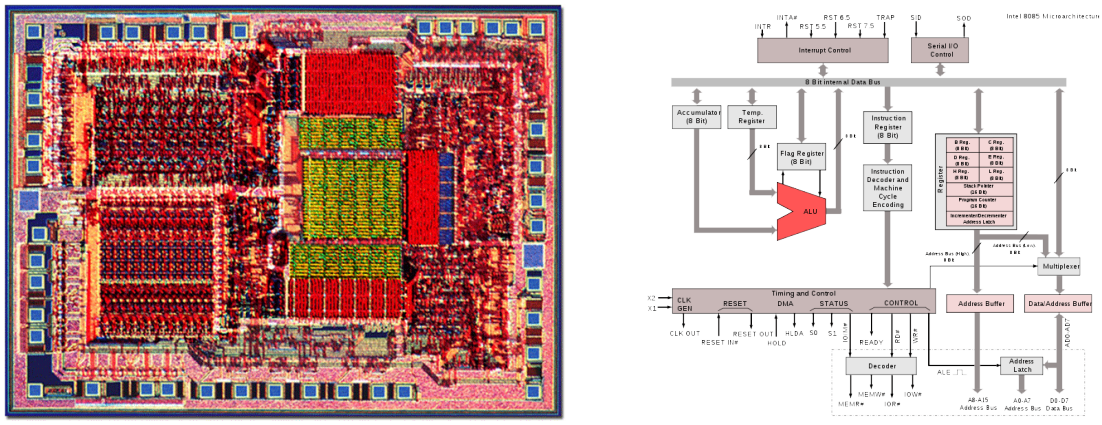



الشكل12 البنية الداخلية والخريطة السيليكونية للمعالج Intel® 4004 – 1970




الشكل15 شريحة المعالج Intel® 4004 ذو ناقل بعرض 4-Bit ويحوي على 2300 ترانزستور

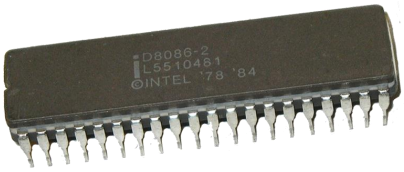
الشكل16 يبين البنية الداخلية (على اليمن) والخريطة السيليكونية (على اليسار) للمعالج Intel®8085. أيضاً الشكل17 يبين شريحة المعالج Intel®8085 الذي تم إنتاجه في عام 1976 ويحوي على 4500 ترانزستور ويعمل بتردد 3MHz. الشكل18 يبين شريحة المعالج Intel®8086 ذو ناقل بيانات بعرض 16-Bit وتم إنتاجه في 1978 ويحوي 29000 ترانزستور ويعمل بتردد 5MHz.



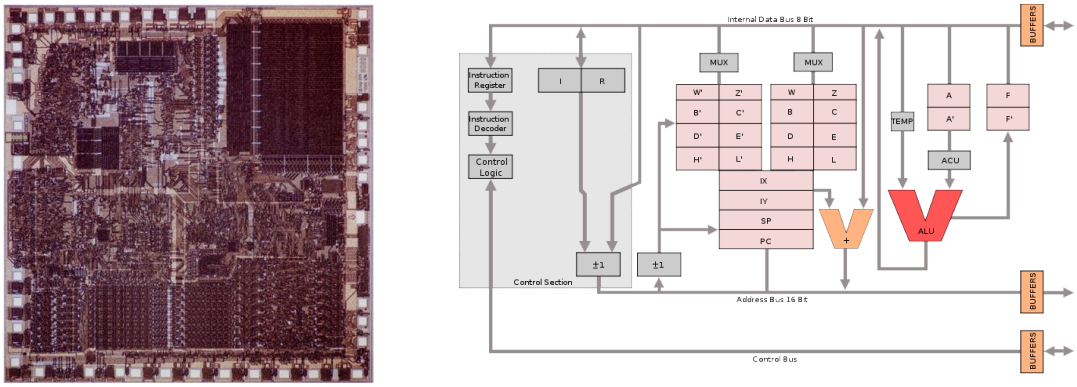

الشكل16 البنية الداخلية والخريطة السيليكونية للمعالج Intel® 8085 – 1976

الشكل17 شريحة المعالج Intel® 8085 ذو ناقل بعرض 8-Bit ويحوي على 4500 ترانزستور

الشكل19 يبين البنية الداخلية (على اليمن) والخريطة السيليكونية (على اليسار) للمعالج Intel®8086.

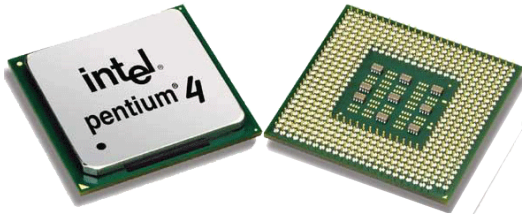



الشكل18 شريحة المعالج Intel® 8086 ذو ناقل بعرض 16-Bit ويحوي على 29000 ترانزستور

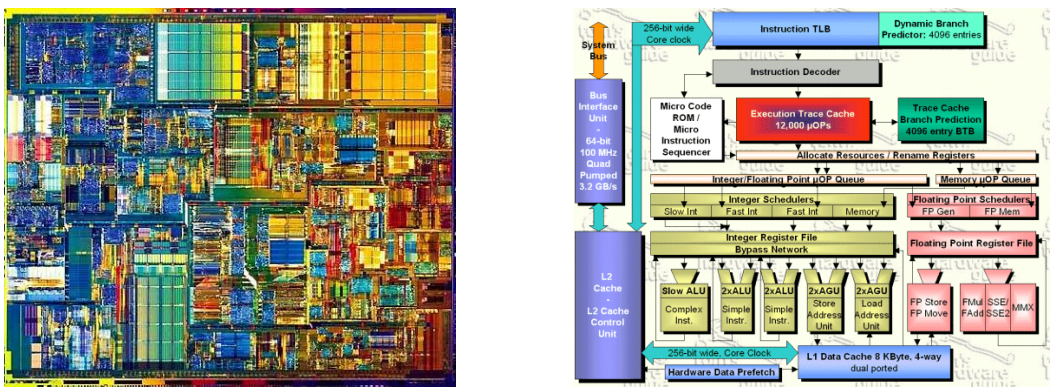



الشكل19 البنية الداخلية والخريطة السيليكونية للمعالج Intel® 8086 – 1978

الشكل20 يبين شريحة المعالج Intel®P4 ذو ناقل بيانات بعرض 32-Bit، وتم إنتاجه في عام 2000، ويحوي على 125 مليون ترانزستور، ويعمل بتردد 1،3~3،8GHz. الشكل21 يبين البنية الداخلية والخريطة السيليكونية للمعالج Intel®P4.




الشكل20 شريحة المعالج Intel®P4 ذو ناقل بعرض 32-Bit ويحوي على 125 مليون ترانزستور

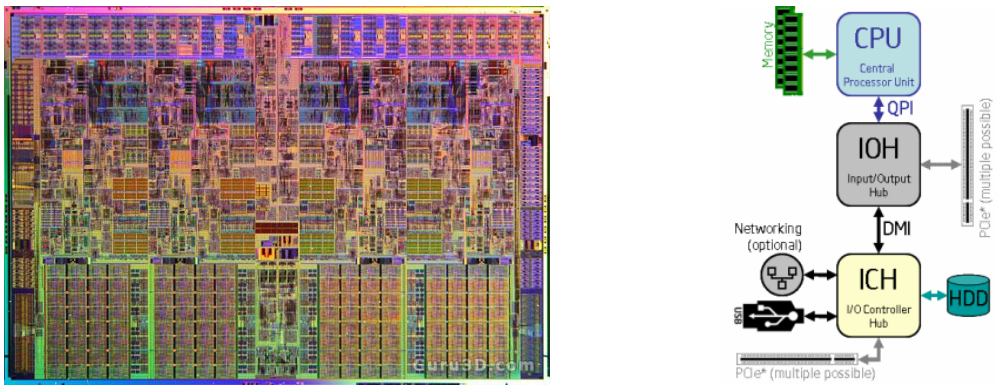



الشكل21 البنية الداخلية والخريطة السيليكونية للمعالج Intel®P4 – 2000

الشكل22 يبين شريحة المعالج Intel® i7 متعددة النوى (4-Core) ذو ناقل بيانات بعرض 64-Bit، تم إنتاجه في عام 2008، ويحوي على 731 مليون ترانزستور، ويعمل بتردد 1.6~3.47GHz. الشكل23 يبين تمثيل البنية الداخلية العامة (على اليمن) والخريطة السيليكونية (على اليسار) للمعالج Intel® i7.

الشكل22 شريحة المعالج Intel®i7 رباعي النوى وذو ناقل بعرض 64-Bit ويحوي على 731 مليون ترانزستور

تتسم التطبيقات التي تستخدم المعالجات المصغرة بالتعقيد على مستوى الكيان الصلب والبرمجي، وذلك لكون المعالج يحوي على وظائف وحدة المعالجة المركزية (CPU) وذاكرة البرنامج (ROM) فقط، وأما باقي المحيطيات كوحدات التوقيت وذاكرة المعطيات RAM ووحدات المقاطعات وغيرها، فجميعها يتم وصلها خارجياً عبر الناقل الرئيسي (BUS)، لذلك فإن معالجات الأغراض العامة تستخدم فقط في الحواسب الشخصية، ولا تستخدم في الأنظمة المدمجة.

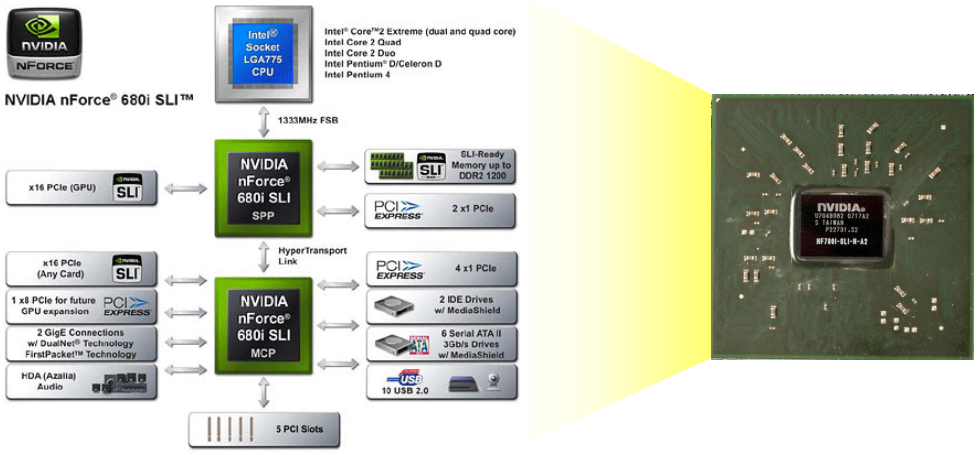



الشكل23 البنية الداخلية والخريطة السيليكونية للمعالج Intel®i7 – 2008

### 1-10-2-2 معالجات الأغراض الخاصة SPPs (Special Purpose Processors):

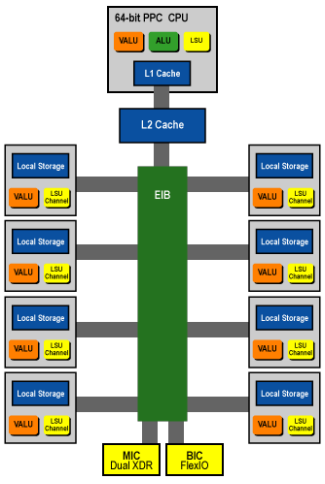
تصمم معالجات الأغراض الخاصة بحيث تؤمن سعة معالجة عالية ووظائف مخصصة متقدمة. مثالها: وحدة معالجة الرسوميات GPU (Graphics Processing Unit). الشكل24 المعالج والمخطط الصندوقي لوحدة معالجة NVIDIA nForce 680i SLI.

في عام 2001 قررت شركة Sony بالتعاون مع شركة IBM وشركة Toshiba تطوير معالج Cell-Processor عالي الأداء، واستمر تطوير هذا المعالج أربع سنوات، وتم صرف مبلغ 400 مليون دولار على أبحاث التطوير، ويشار إليه عادةً بـCBEA ( Cell Broadband Engine Architecture).



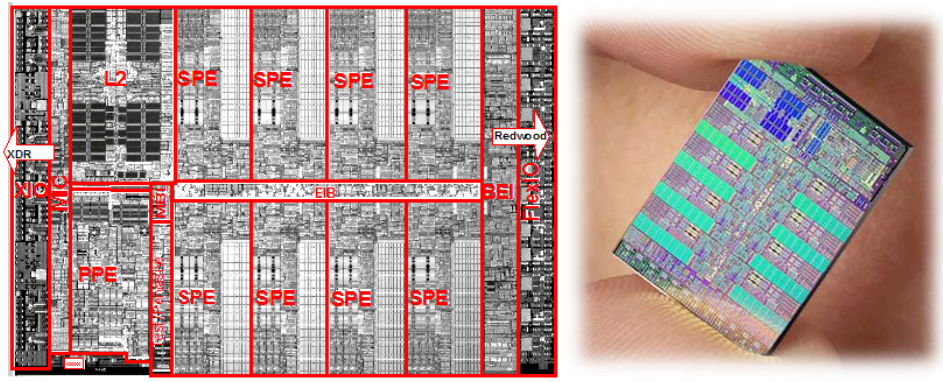

الشكل24 المخطط الصندوقي لوحدة معالجة الرسوميات nForce 680i SLI

يضم المعالج Cell-Processor معالج أغراض عامة من النوع Power-PC Core 64-bit يسمى بـ PPE إضافةً إلى مجموعة معالجات مؤازرة من المعالجات الخاصة من النوع SoCs لتسريع الرسوميات والوسائط تسمى بـ SPE، هذه المعالجات المؤازرة متصلة مع الوحدة الرئيسية PPE عبر ناقل يسمى بـEIB، وكلاهما متصل مع ذاكرة النظام عبر متحكم يدعى بـDMIC الذي يلج ذاكرة من نوع XDR بسعة 25GB/s، وكذلك يملك وحدات إدخال وإخراج من النوع FlexIO ذات سرعة تصل إلى 76.8GBs، والتشغيل الأول لهذا المعالج عند ترددz4GH.




الشكل25 المخطط الصندوقي للمعالج Cell-Processor

يتميز المعالج Cell-Processor بالإمكانيات الهائلة في معالجة العمليات الحسابية المعقدة، وخصوصاً الفاصلة العاملة (Floating-point) إضافةً إلى البرمجة الموزعة والمتعددة المهام، وحالياً يستخدم هذا المعالج في جهاز Playstation3 ويملك 9-core. الشكل25 يبين المخطط الصندوقي للمعالج Cell-Processor المستخدم في جهاز Playstation3. الشكل26 يبين الخريطة السيليكونية للمعالج Cell-Processor.




الشكل26 الخريطة السيليكونية للمعالج Cell-Processor

### 1-2-10-3 معيارية تصميم بنية المعالجات (CPUs Architecture Design Standard):

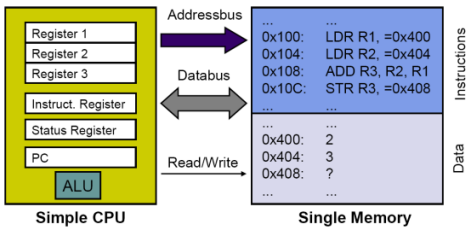
تعرف المعيارية بأنها الطريقة التي يتعامل بها المعالج مع الذاكرة في جلب وتنفيذ التعليمات وتخزين البيانات، ويوجد معياريتين أساسيتين في تصميم المعالجات:

- معيارية Harvard.
- معيارية Von-Neumann.

#### 1-3-2-10-1 معيارية Von-Neumann:

تعتمد هذه المعيارية على المعالج وناقل وحيد لنقل التعليمات والبيانات بين الذاكرة ووحدة المعالجة، وبالتالي سوف يحتاج إلى نبضات توقيت أكثر من أجل تنفيذ عملية واحدة، لذلك تتصف هذه النظم بكونها بطيئة نسبياً، مبدأ عملها يتلخص بما يلي:

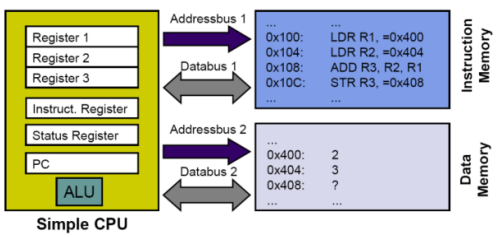
1- يقوم المعالج بجلب التعليمات من الذاكرة.
2- يقوم بقراءة البيانات من الذاكرة.
3- إجراء العمليات على البيانات.
4- إعادة كتابة تلك البيانات على الذاكرة.




الشكل27 معيارية Von-Neumann وطريقة ربط المعالج مع الذاكرة

#### 2-3-2-10-1 معيارية Harvard:

تتكون هذه المعيارية من المعالج وناقلين منفصلين أحدهما لنقل التعليمات والآخر لنقل البيانات، وتختلف ذاكرة البيانات عن ذاكرة التعليمات حيث أن لكل ذاكرة خطوط عنونة وتحكم وممر معطيات مختلفة، وبالتالي فإن عملية قراءة التعليمات والبيانات تتم في نفس الوقت، وستكون سرعة التنفيذ أكبر.




الشكل28 معيارية Harvard وطريقة ربط المعالج مع الذاكرة

### 1-10-2-4 بنى مسجلات التعليمات في المعالجات (CPUs Instruction Set Architectures):

حتى منتصف الثمانينات في القرن السابق كان التوجه السائد في عالم صناعة المعالجات هو بناء معالجات ذات تعليمات أعقد وأكثر عدداً بهدف تسهيل عملية البرمجة، ولكن في تلك الأثناء ظهر توجه آخر معاكس تماماً، وهو السعي لبناء معالجات ذات تعليمات بسيطة ومحدودة العدد يمكن تنفيذها بسرعات عالية جداً. تقسم بنى مسجلات التعليمات في المعالجات إلى ثلاث بنى أساسية:

1- CISC (300 ~ 3000 Instruction).
2- RISC (50 ~ 200 Instruction).
3- MISC (15 ~ 30 Instruction).

#### 1-10-2-4-1 البنية CISC:

وهي مجموعة أوامر الحاسب المعقدة "Complex Instruction Set Computer"؛ معظم معالجات الحواسب الشخصية تستخدم معمارية CISC، والتي تدعم مجموعة تعليمات قد يصل عددها إلى 3000 تعليمة أو أكثر.

الدافع الأساسي لهذه التقنية هو تخفيض التكلفة العامة للحواسب، وذلك عن طريق جعل البرمجة – وهي العنصر الأكثر تكلفة في أي نظام حاسوبي – أكثر سهولة وبالتالي أقل تكلفة.

يتلخص جميع ذلك بتطبيق مبدأ بسيط وهو: نقل التعقيد من البرمجيات إلى العتاد الصلب، لهذا السبب يتم تخصيص تعليمة لكل حدث يتم في المعالج، وبالتالي يمكن أن تصل مجموعة تعليمات هذه المعالجات إلى آلاف التعليمات، كما أن القاعدة الأساسية تقول: إن أداء الكيان الصلب "دائماً" أسرع بكثير من الأداء البرمجي.

على النقيض من ذلك، فإن زيادة عدد التعليمات يزيد من سهولة البرمجة، ويسرع زمن تسويق المنتج (Time to Market)، ولكن بنفس الوقت يؤدي إلى زيادة تعقيد العتاد الصلب للمعالج، حيث سيحتاج إلى وحدة ترجمة معقدة داخل نفس المعالج للتعرف على كم التعليمات الكبير، كما أن دورة تنفيذ التعليمة ستستغرق وقتاً إضافياً داخل وحدة الترجمة حتى يتم تفسيرها مما يعني تباطؤاً في الأداء، كما أنه وبسبب الحاجة إلى مسجلات داخلية إضافية لهذه التعليمات؛ فإن عدد الترانزستورات لبنية المعالج ستزداد، وبالتالي ستزداد ضياعات الطاقة في المعالج مما ينتج عنه ارتفاع في درجة حرارة المعالج، وسيحتاج إلى وحدة تبريد خاصة، وهذا بالفعل ما نلاحظه في معالجات AMD & INTEL المستخدمة في الحواسب الشخصية. إن السبب الأساسي في زيادة عدد التعليمات في المعالجات التي تتبنى البنية CISC – على الرغم من الجانب السلبي لهذا الأمر – هو أن هذه المعالجات تكون مكرسة لأغراض عامة ذات مهام معقدة، وبالتالي فإن برنامج هذه المعالجات يكون في غاية التعقيد، لهذا السبب يتم تزويد المعالج بمسجلات تعليمات لكافة العمليات الرياضية (Cos،Sin ، etc...) وغيرها وهذا لا يتوفر في المعالجات التي تتبنى البنية RISC. من أشهر عائلات المعالجات التي تتبنى البنية CISC هي: System/360، PDP-11، VAX، 68000 and x86 إضافةً إلى Intel، AMD، Cyrix and IBM.

### 1-10-2-4-2 البنية RISC:

وهي مجموعة أوامر الحاسب المختصرة "Reduced Instruction Set Computer"، وهي نوع من المعالجات التي تملك مجموعة محدودة نسبياً من التعليمات البرمجية العامة والأساسية، والتي تبلغ حوالي 200 تعليمة كحد أعظمي.

من ميزات تعليمات المعالجات ذات البنية RISC أنها قصيرة ولا تحتاج لوحدة ترجمة خاصة (Microcode)، مما يسرع في عملية التنفيذ حيث يمكن أن تصل سرعة التنفيذ في بعض المعالجات إلى دورة آلة واحدة لكل تعليمية. ميزة أخرى قد تكون أكثر أهمية، وهي أنه بسبب قلة وبساطة تعليمات هذا النوع فقد أصبح بالإمكان تقليل عدد المسجلات الداخلية، والذي يؤدي إلى تقليل عدد الترانزستورات، وبالتالي تخفيض تكلفة التصنيع واستهلاك الطاقة. على النقيض من ذلك، فإن قلة عدد التعليمات وعموميتها ينعكس سلباً على تعقيدات كتابة برنامج المعالج وطوله.

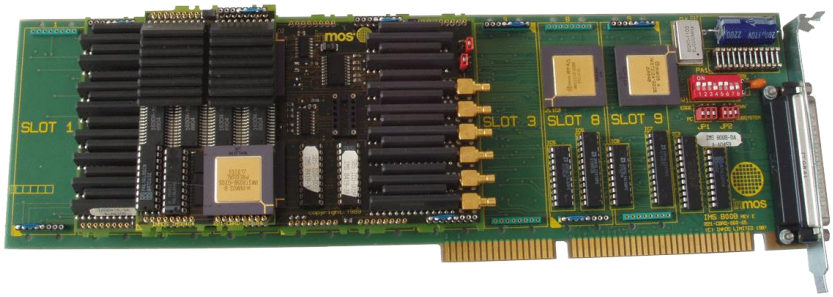
مؤخراً ومع وجود لغات برمجية عالية المستوى لم تعد هناك مشكلة في تعقيد برامج المعالجات ذات البنية RISC، وهذه البنية تعتمدها معظم أنواع المتحكمات المصغرة (Microcontrollers) ومعالجات الإشارة الرقمية (DSPs).

من أشهر عائلات المعالجات التي تتبنى البنية CISC هي: DEC Alpha، AMD 29k، ARC، ARM، Atmel AVR ، MIPS، PA-RISC، PowerPC، SuperH، SPARC and .

### 1-10-2-4-3 البنية MISC:

تقوم هذه البنية على عدد قليل جداً من التعليمات الأساسية بهدف تقليل عدد المسجلات الداخلية للمعالج، كما أن هذا النوع من التعليمات يعتمد على المكدس (Stack-based) – الذي يستخدم لتخزين عنوان العودة عند القفز إلى البرامج الفرعية في بنى التعليمات التي تستخدم المسجلات – بدلاً من كونه معتمداً على المسجلات (Register-based)، وبالتالي يتم فك تشفير التعليمات بسرعة أكبر غير أن هذا يؤدي إلى كون التنفيذ يعتمد على التسلسل التتابعي للتعليمة.

هذا النوع من بنى التعليمات شائع في Java Virtual Machine، ومن أبرز التطبيقات التجارية التي تبنت هذه البنية هو الحاسوب INMOS Transputer مبين على الشكل29.




الشكل29 اللوحة الأم للحاسوب Transputer Evaluation IMSB008

الجدول4 يلخص مقارناً بين بنية التعليمات RISC والبنية CISC.

| معالجات CISC | معالجات RISC |
|---|---|
| عدد كبير جداً من التعليمات البرمجية يصل إلى 3000 | عدد قليل من التعليمات البرمجية لا يتجاوز 200 |
| تعليمات برمجية معقدة يستغرق تنفيذها زمناً كبيراً يصل إلى 12 دورة | تعليمات برمجية أساسية بسيطة يمكن تنفيذها بدورة واحدة فقط |
| التعليمات تحتاج إلى Microcode في المعالج لترجمتها قبل التنفيذ | تنفيذ التعليمات يتم بشكل مباشر دون الحاجة لوحدة ترجمة |
| كتلة التعليمات تتفاوت في الطول والتعقيد | كتلة التعليمات بسيطة وموحدة الطول (8، 16، 32-bit ) |
| تحتاج للوصول إلى الذاكرة أثناء التنفيذ | لا حاجة للوصول للذاكرة (الأوامر في المسجلات) |
| نادراً ما تستخدم تقنية Pipelining | تستخدم تقنية Pipelining بشكل واسع |
| ذات تعقيد في الكيان الصلب (Microcode Unit) | ذات تعقيد على مستوى البرمجيات (Compiler) |
| مجموعة مسجلات وحيدة والنقل يتم من الذاكرة | العديد من مجموعات المسجلات ويتم التنفيذ منها |

الجدول4 مقارنة بين بنية التعليمات RISC البنية CISC

### 3-10-1 المتحكمات المصغرة µC،MCUs (Microcontrollers):

يمثل المتحكم المصغر منظومة حاسوبية متكاملة مصغرة متوضعة على دارة متكاملة وحيدة. بخلاف المعالجات المصغرة (MPU) المستخدمة في الحواسب الشخصية والتطبيقات الأخرى عالية الأداء، فإن المتحكمات المصغرة تستخدم في التطبيقات صغيرة الحجم حيث يكون استهلاك الطاقة محدوداً، كأجهزة التحكم عن بعد، والتجهيزات المنزلية، والألعاب، بالإضافة إلى أنظمة التحكم في السيارات وغيرها.

تعتبر المتحكمات الرقمية المصغرة القلب النابض في أنظمة التحكم وفي التجهيزات الكهربائية والإلكترونية، وبقدر ازدياد تعقيد الوظائف المطلوبة من هذه الأنظمة، يزداد تعقيد بنية هذه المتحكمات؛ لذلك تتوفر هذه المتحكمات ضمن طيف واسع جداً من العائلات التي تتنوع بتنوع وظائفها وتطبيقاتها، فمنها الخاص ومنها العام.

تعتبر صناعة السيارات القوة المحركة في ازدياد نمو تطوير المتحكمات المصغرة، وتشير الإحصاءات إلى أن 33% من المتحكمات المصنعة تستخدم في أنظمة التحكم في السيارات الحديثة، كما تشير الإحصاءات إلى أن عدد المتحكمات المصغرة التي تستخدم في السيارات ذات الكلفة المنخفضة يتراوح 40~30، في حين يستخدم 100~70 متحكم في السيارات ذات الكلفة المرتفعة. من الجدير ذكره أن متطلبات قطاع صناعة السيارات دفعت شركات تصنيع المتحكمات المصغرة إلى تطوير وتبني برتوكولات اتصال تسلسلي جديدة ذات وظائف وميزات تستوعب ربط آلاف الحساسات مثل CAN & LIN.

تصنف المتحكمات المصغرة بشكل أساسي وفقاً لعرض الناقل الرئيسي (4-bit، 8-bit، 16-bit، 32-bit، 64-bit ) الذي يصل بين وحدة المعالجة المركزية وبين ذاكرة المتحكم. إن معيار اختيار المعالج وفقاً لعرض الناقل الرئيسي يعتمد على درجة تعقيد النظام، فمثلاً: تستخدم معالجات 4-bit في أجهزة التحكم عن بعد وألعاب الأطفال، وهذه المعالجات تكون محدودة الميزات، وتعمل عند تردد لا

يتجاوز 8MHz. وأما معالجات 8-bit فتستخدم في أنظمة التحكم بالغسالات والأجهزة المنزلية، وهي تعمل عند تردد لا يتجاوز 20MHz. وتستخدم معالجات 16-bit في أنظمة التحكم الرقمي بالمحركات وهي تعمل عند تردد لا يتجاوز 40MHz، وأما معالجات 32-bit فتستخدم في الأنظمة المتقدمة التي تحتاج إلى معالجة بالفاصلة العشرية أو تحوي على نظام تشغيل مدمج (RTOS) مثل Linux، وهي تملك ميزات واسعة لا تملكها المعالجات الأدنى، مثل: إمكانية الربط مع بروتوكولات اتصال تسلسلي عالية السرعة (CAN،Wi-Fi،USB،Ethernet) وتكون بنيتها أقرب إلى بنية الحواسب وتعمل عن ترددات تتراوح من 60MHz~400MHz.

### 1-3-10-1 الميزات الوظيفية للمتحكمات المصغرة (Microcontrollers Functional Features):

تمتلك المتحكمات المصغرة العديد من الميزات الأساسية والميزات المحيطية الوظيفية؛ إن الهدف من تنوع هذه الميزات هو تقليل عدد العناصر الخارجية المحيطية على الدارة المطبوعة (PCB) وذلك بهدف:

- ✓ تخفيض استهلاك الطاقة.
- ✓ تخفيض تكلفة تطوير النظام من خلال تقليص زمن التصميم.
- ✓ الحصول على أداء أعلى.
- ✓ الحصول على وثوقية عالية.

إضافةً إلى الميزات الأساسية التي تشترك بها جميع المتحكمات المصغرة، فإن العديد من الميزات ظهرت مؤخراً، وأدت إلى دفع عجلة المتحكمات المصغرة لتكون بديلاً عن استخدام المعالجات المصغرة في العديد من مشاريع الأنظمة المدمجة.

. . . ﴿انتهت الجلسة العملية الأولى﴾ . . .

- دمتم بخير ومودة ونور -

وليد بليد

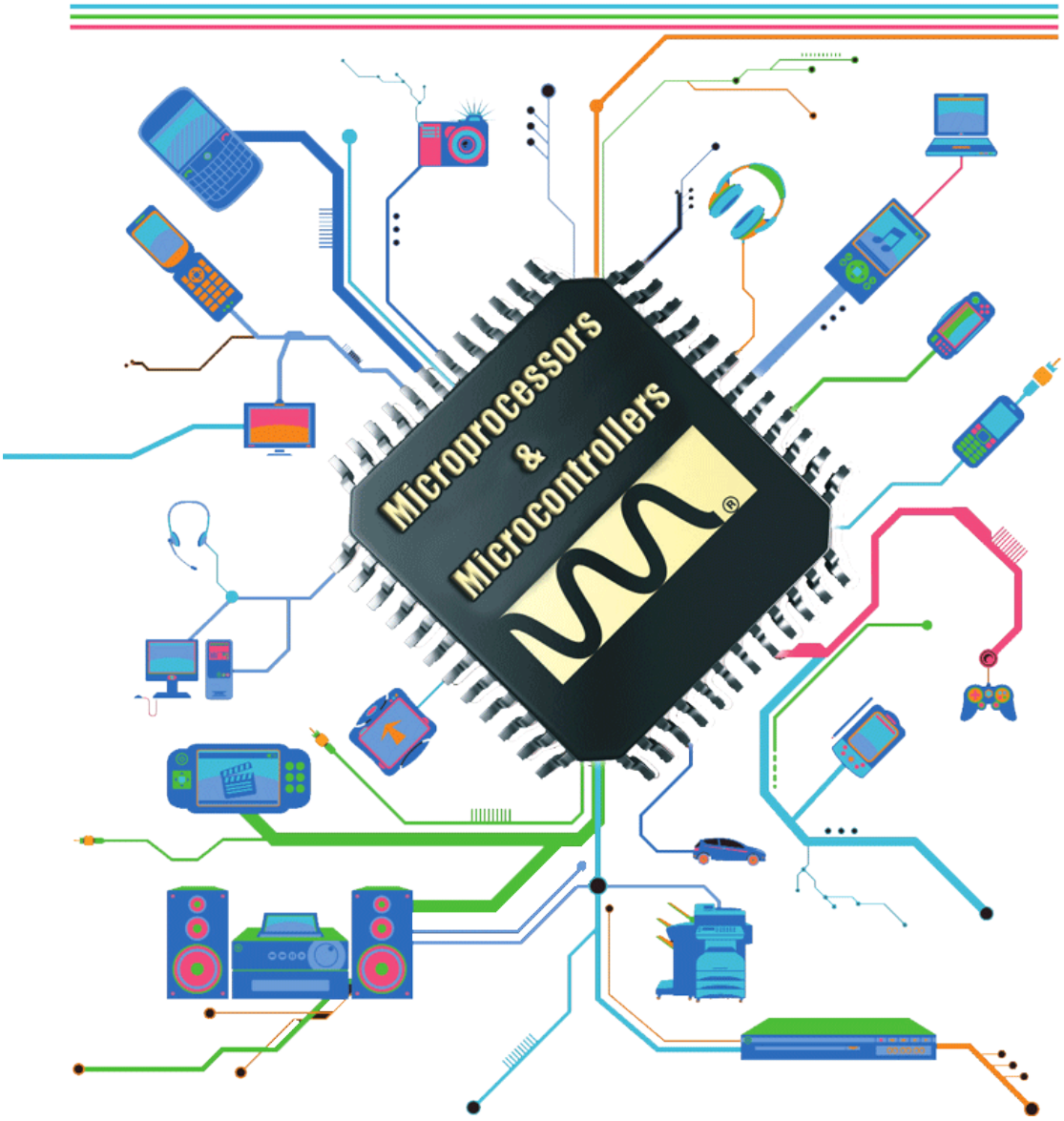
جامعـــــــــــة حلـــــــــــب
كلية الهندسة الكهربائية والإلكترونية
قسـم هندسـة التحكـم والأتمتـة

# ﴿ الجلسات العملية لمادة المعالجات والمتحكمات المصغرة ﴾

# Microprocessors & Microcontrollers Lab Sessions

## السنة الثالثة | قسم اتصالات

## ﴿ الجلسة العملية الثانية ﴾




م. وليد بليد

Copyright © 2012 Walid Balid - All rights reserved.

Wednesday, March 07, 2012

# ﴿الجلسة العملية الثانية﴾

## نظرة عامة (Overview):

هذه الجلسة تقدم تصنيفاً للغات برمجة الأنظمة المدمجة ومعايير اختيار الحلول التكنلوجية. ثم نظرة عامة عن الأدوات البرمجية ولوحة التطوير التي ستستخدم خلال الجلسات العملية ومنهجية تنفيذ التجارب. ثم مدخل إلى متحكمات AVR وعائلاتها والبنية الداخلية وتنظيم الذاكرة، ثم نظرة عامة على ميزات المتحكم ATmega32A.

## 1-2 تمهيد (Preface):

مما لا شك فيه أن لغة التجميع (Assembly) هي أكثر فعالية من غيرها من اللغات ذات المستوى الأعلى عندما يتعلق الأمر بالتعامل المباشر مع وظائف وسلوك المعالج أو المتحكم المصغر، ولكن هذا من جانب آخر يحتاج إلى مستوى عالٍ من الخبرة في بنية الكيان الصلب للمعالج لتوظيف لغات التجميع بشكل فعال، كما أنه سيحتاج إلى أضعاف مضاعفة من الوقت لبناء تطبيق محدد؛ السبب الذي يجعل مطوري البرامج الحاسوبية يعانون من نقص الخبرة حول تفاصيل وتعقيدات تطوير الكيان الصلب؛ مما يحد من مقدرتهم على تصميم الانظمة المدمجة (**ESs**).

للاستفادة من مهارات مطوري البرمجيات الحاسوبية، والحد من طابع تعقيدات برمجة الكيان الصلب بلغة التجميع، قامت شركات الـ"EDA" (Electronic Design Automation) بتطوير بيئات وأدوات برمجية تستخدم لغات عالية المستوى مثل ال:-C/C++ القياسية أو لغة الBasic أو لغة الPascal لتطوير وبرمجة الأنظمة المدمجة. هذه الأدوات البرمجية الجديدة تقوم على تحويل البرنامج من لغة عالية المستوى إلى برنامج بلغة التجميع منخفض المستوى وتدعى في أغلب الأحيان بـMappers أو Compilers. تمتلك هذه الأدوات المقدرة على تطوير وفحص وتتبع أخطاء البرامج التي هي مشابهة جدا لبيئات تطوير البرمجيات.

الجذور الأولى لنشأة لغات البرمجة عالية المستوى كانت مع إشراقة فجر عصر الحوسبة الحديثة في منتصف الخمسينيات من القرن الماضي، حيث كان فريق صغير من الباحثين في شركة IBM قد قرر إيجاد بديل آخر لاستخدام لغة التجميع منخفضة المستوى (Assembly) في برمجة الحاسب IBM-704، وكانت النتيجة ظهور لغة Fortran – شكلٌ آخر من أشكال لغات البرمجة أكثر قابلية للقراءة والفهم – والتي تهدف في الأساس إلى تسريع عمليات تطوير البرامج المختلفة.

لقد انتاب المجتمع الهندسي في البدء بعضُ الشكوك في كون هذه الطريقة الجديدة قادرة على التفوق على البرامج المكتوبة يدوياً بلغة التجميع، ولكن سرعان ما ثبت أن البرامج المكتوبة بلغة Fortran قادرةٌ على العمل تقريباً بنفس فعالية تلك المكتوبة بلغة التجميع؛ وفي نفس الوقت، استطاعت لغة Fortran تقليص عدد التعليمات البرمجية المستخدمة لبناء برنامج ما بحوالي عشرين مرة، وهذا ما جعلها

تعتبر أولى لغات البرمجة عالية المستوى، ولم يكن من المفاجئ أن لغة Fortran قد حصلت بسرعة كبيرة على رضى وقبول المجتمع العلمي في ذاك الوقت وحتى وقت متأخر. بعد نصف قرن، ما زلنا نستطيع استخلاص الكثير من العِبَر الهامة من هذه القصة وهي:

- ✓ لأكثر من خمسين عاماً، حاول المهندسون ابتكار طرق أسهل وأسرع لحل المشكلات باستخدام البرمجة الحاسوبية.
- ✓ لغات البرمجة التي اختارها المهندسون لتلبية متطلباتهم اتجهت نحو مستويات أعلى (High-level of Abstraction).

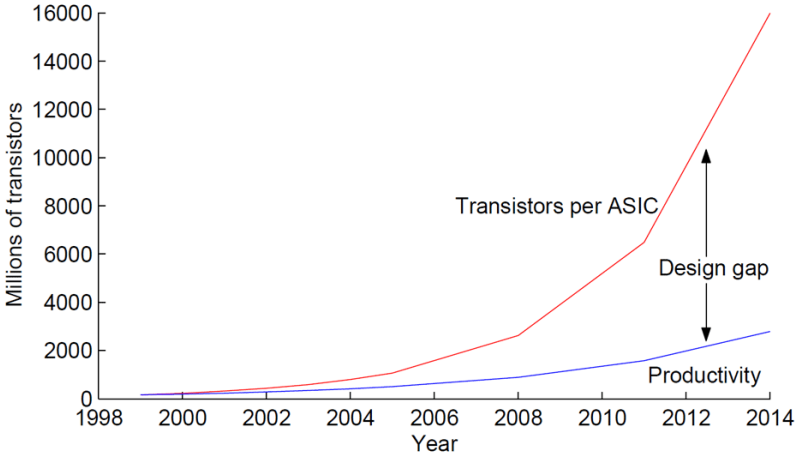
## 2-2 تصنيف لغات برمجة الأنظمة المدمجة (ESs. Programming Languages):

تصنف لغات برمجة الأنظمة المدمجة وفق أربع مجموعات رئيسية، وهي:

1. لغات البرمجة منخفضة المستوى – لغة التجميع (Low-level Programming Languages, Assembly).
2. لغات البرمجة عالية المستوى – الوظيفية (High-level F. Programming Language) مثل: Basic, Pascal.
3. لغات البرمجة عالية المستوى – الإجرائية (High-level P. Programming Languages) مثل: Embedded C/C++.
4. لغات البرمجة الرسومية (Graphical Programming Languages) مثل: LabVIEW.

## 3-2 نحو لغات برمجية للأنظمة المدمجة عالية المستوى (Toward High-level ESs. Programming Languages):

إن الازدياد المتسارع في تعقيد بنية شرائح الدارات المتكاملة وفقاً لقانون "Moore" يؤدي إلى نشوء فجوة كبيرة بين عدد الكتل المنطقية التي يمكن إنشاؤها على مساحة محددة من شريحة سيليكونية وبين إمكانية مهندسي التصميم على استثمار هذه الشريحة المتكاملة برمجياً، وهذا ما يدعى الآن بـ"Design Gap". الشكل1 يبين العلاقة بين الإنتاجية التصميمية لمهندسي تصميم الشرائح المتكاملة وبين عدد الترانزستورات التي يمكن أن توضع على شريحة سيليكونية.




الشكل1 العلاقة بين ازدياد تعقيد الدارات المتكاملة ومقدرة المصممين على تطويرها

إن نشوء هذه الفجوة يقود إلى الحاجة في البحث عن طرائق ووسائل لبرمجة الكيان الصلب بمستويات برمجية أعلى وهو ما يعرف اصطلاحاً بـ"high-level abstraction". على الرغم من أن التصميم الذي يمكن إنشاؤه من مستوى برمجي أعلى قد يكون أقل كفاءة مقارنةً مع التصميم الذي يتم إنشاؤه مباشرة باستخدام لغات التجميع، إلا أن هذا الأمر أقل أهمية بكثير من مسائل التعقيد البرمجي والجهود الكبيرة

والزمن المصروف للبرمجة بلغات التجميع. على كل حال فإنه مؤخراً تمّ بذل العديد من الجهود تهدف لإيجاد أدوات برمجية بلغات عالية المستوى تهدف إلى الوصول لتصميم أمثلي. هناك مسألة أخرى تتعلق باستخدام لغات عالية المستوى في برمجة الأنظمة المدمجة وهي أنه لكي تكون ناجحاً في الاقتصاد العالمي اليوم، فإن مسألة وصول المنتج إلى السوق يجب أن تتم بشكل أسرع من السابق، وبالتالي فإن دورة تصميم المنتج يجب أن تكون أقصر ما يمكن.

### 4-2 معايير اختيار الحلول التكنلوجية (Technology Solutions Selection Criteria):

كما أن هناك عوامل مؤثرة في تصميم الأنظمة المدمجة (الفقرة 5-1)، فإنه في مشاريع الأنظمة المدمجة عموماً هناك معايير أساسية تعتمدها الشركات في اختيار الحلول التكنلوجية والعناصر الأساسية في المنتجات الإلكترونية – من أهم العناصر الأساسية هي الشريحة المدمجة؛ متحكم مصغر (MCU)، معالج مصغر (MPU)، معالج إشارة رقمية (DSP)، مصفوفة بوابات منطقية قابلة للبرمجة حقلياً (FPGA)، دارات متكاملة ذات تطبيقات خاصة (ASIC) – في منتجاتها والتي منها: تكلفة الشريحة، مستوى المرونة في بيئة التطوير، مستوى الدعم الفني من قبل الشركة المصنعة، مستوى الشركة المصنعة تجارياً. إن جميع هذه النقاط ضرورية جداً في تحديد أفق التطوير المستقبلي للمشاريع، حيث لا يمكن الاعتماد على الشركات المنطلقة حديثاً (Startups) عند اختيار المعالجات والشرائح الرقمية القابلة للبرمجة. الفقرات التالية تعالج هذه المعايير والاعتبارات.

### 5-2 معايير اختيار الشركات المصنعة (Manufacturer Selection Criteria):

يوجد العديد من المعايير والعوامل الهامة في اختيار الشركة المصنعة للحلول التكنلوجية نوردها فيما يلي:

- ✓ توفير الأدوات والحلول البرمجية (Tools).
- ✓ الريادة في التكنولوجيا (Technology leadership).
- ✓ تزويد الوحدات البرمجية المتوافقة مع الحلول البرمجية (IP offerings).
- ✓ تزويد منتجات بمواصفات ومزايا إبداعية (Innovative product features).
- ✓ تطوير أجيال المنتجات بشكل متجدد (Solid roadmaps).
- ✓ دعم وتصنيع عائلات الشرائح لفترات طويلة جداً (Longevity of parts).
- ✓ توفر العديد من عائلات الشرائح بميزات عديدة (Multiple families).
- ✓ الدعم الفني والتقني (Support).

### 6-2 عوامل اختيار الشريحة المناسبة للتطبيق (Chip Selection Factors):

بما أن الشركات المصنعة الرائدة توفر العديد من العائلات والشرائح التي تتفاوت في ميزتها وأدائها وسعرها، فإنه من الضروري اختيار الشريحة المناسبة للتطبيق من خلال تحديد العوامل المطلوبة وهي:

- ✓ السعر (Cost).

- ✓ الحجم (Size).
- ✓ الطاقة (Power).
- ✓ السرعة (Speed).
- ✓ عدد أقطاب الدخل والخرج (I/O Count).
- ✓ المصادر المنطقية الأساسية على الشريحة (Logic fabric resources).
- ✓ مصادر إدارة تردد عمل الشريحة (Clock management resources).
- ✓ مصادر الذاكرة (Memory resources).
- ✓ الشكل الفيزيائي للشريحة (Packaging).
- ✓ إمكانية وجود بدائل للشريحة (Common footprint component migration options).
- ✓ متطلبات الربط مع المحيطيات (Interface requirements).
- ✓ ميزات أدوات التصميم وانتشارها (Design tool features and familiarity).

خلال دراستنا هذه وقع الاختيار على شركة ATMEL الرائدة في مجال تصنيع المتحكمات المصغرة حيث تعتبر من أقوى الشركات عالمياً في مجال تطوير وتصنيع متحكمات 8-bit وفقاً لتقرير Gartner، كما أن تنتج طيفاً واسعاً من متحكمات 8-bit بأداء عالٍ وميزات كبيرة جداً وكل ذلك بسعر منخفض. أضف إلى ذلك انتشارها الواسع جداً (حتى في أسواقنا المحلية المتواضعة) وكثرة المصادر التعليمية المتوفرة على الشبكة..

## 7-2 لغات برمجة المتحكمات المصغرة (Microcontrollers Programming Languages):

يعتمد عمل المتحكم المصغر على مبدأ أساسي وهو تنفيذ مجموعة التعليمات الموجودة بداخله في ذاكرة البرنامج, إذ يقوم بتنفيذ التعليمات بشكل تتابعي (تعليمة تلو الأخرى) إلى أن يصل إلى نهاية البرنامج، وعندها يعود إلى التعليمة الأولى في بداية البرنامج ليبدأ بتنفيذ دورة أخرى من البرنامج وهكذا...

تختلف اللغات المستخدمة في كتابة برنامج المتحكم وتتفاوت في مقدار صعوبتها وتعقيدها, وسابقاً كانت تتم برمجة المتحكمات بشكل أساسي باستخدام لغة التجميع (Assembly)، حيث يمتلك كل نوع من المتحكمات لغة تجميع خاصة به تتعلق بنواة المعالج, حتى أن متحكمات نفس الشركة تختلف في لغة التجميع الخاصة بها من نوعٍ إلى آخر؛ فمتحكمات شركة ATMEL – على سبيل المثال – ذات نواة AT89xxxx تمتلك لغة تجميع تختلف عن لغة التجميع الخاصة بنواة متحكمات AT90xxxx. ومن المعروف أن لغة التجميع لغة اختصاصية وغير مرنة، كما أنها صعبة التدقيق والمراجعة والتطوير... فكتابة برنامج ما باستخدام لغة التجميع سيحتاج إلى وقت طويل وخبرة كبيرة من قبل المبرمج.

مع التطور التكنولوجي الكبير في صناعة الأنظمة المدمجة عموماً والمتحكمات المصغرة خصوصاً تنبهت الشركات إلى أن استمرار استخدام لغة التجميع يحد من سهولة استخدام هذه المتحكمات، وبالتالي تبقى محصورة ضمن فئة معينة من المهندسين المختصين، لذلك تم البحث عن طرق أسهل لبرمجة المتحكمات، مما أدى إلى ظهور لغات البرمجة عالية المستوى (High Level Languages) مثل: لغة C/C++ ولغة BASIC ولغة PASCAL وغيرها... في هذه اللغات يقوم المبرمج بكتابة البرنامج بإحدى لغات البرمجة عالية المستوى ويقوم مترجم خاص (Compiler) بتحويل هذا البرنامج إلى البرنامج المقابل له في لغة التجميع الخاصة بالمتحكم المصغر, وبالتالي لا حاجة إلى تعلم لغة التجميع كلما احتاج المبرمج استخدام عائلة جديدة من المتحكمات. من أشهر مترجمات متحكمات AVR نذكر:

| اسم المترجم | اللغة | موقع الشركة |
|---|---|---|
| BASCOM-AVR | Basic | http://www.mcselec.com/ |
| AVR-Studio | C/C++ | http://www.atmel.com/ |
| CodeVisionAVR | C | http://www.hpinfotech.ro/ |
| Win-AVR | C++ | http://winavr.sourceforge.net/ |
| ImageCraft ICC-AVR | C | http://www.imagecraft.com/ |
| MikroPascal For AVR | Pascal | http://www.mikroe.com/ |
| MikroBasic For AVR | Basic | http://www.mikroe.com/ |

وبالتالي وباستخدام لغات برمجة عالية المستوى انكسر حاجز الاختصاص التي تفرضه لغة التجميع وأصبحت المتحكمات في متناول الجميع ممن لديهم خبرة متواضعة في لغات البرمجة ومعرفة كافية في بنية المتحكمات, كما أن وجود المكتبات المختلفة في هذه المترجمات جعلت عملية برمجة المتحكمات عملية ممتعة وسلسة ولا تحتاج إلى الكثير من الوقت في دراسة متطلبات الكيان الصلب. فيما يلي مقارنة سريعة لبعض الميزات الأساسية للّغات عالية المستوى مع ما يقابلها من اللغات المنخفضة المستوى.

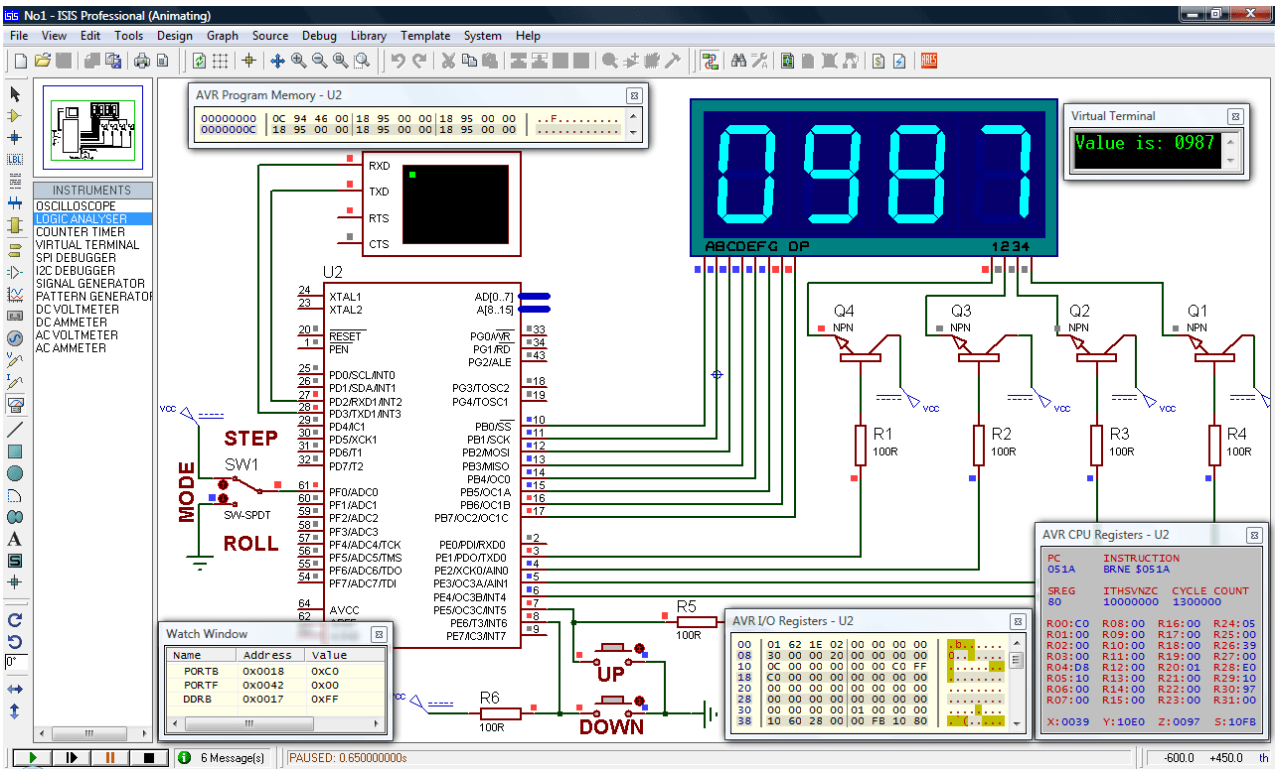
| برامج مكتوبة بلغات برمجة منخفضة المستوى (Assembly) | برامج مكتوبة بلغات برمجة عالية المستوى (Basic, Pascal, C) |
|---|---|
| – تحتاج إلى وقت كبير لتعلمها | – يمكن تعلمها بوقت قصير جداً |
| – كتابتها تحتاج إلى خبرة كبيرة وأشخاص مختصين | – كتابتها لا تحتاج إلى تلك الخبرة وهي متاحة للجميع |
| – صعوبة في كتابة البرامج وتنقيحها ومراجعتها وتطويرها | – سلاسة في كتابة البرامج وتنقيحها وتطويرها |
| – برامج طويلة تتألف من عدد كبير جداً من التعليمات | – برامج أقصر وذات تعليمات أقل وأبسط |
| – برامج صعبة الكتابة القراءة والفهم | – برامج سهلة الكتابة وواضحة القراءة وتفهم بسهولة |
| – تستهلك مساحة أصغر من ذاكرة البرنامج | – تستهلك مساحة أكبر من ذاكرة البرنامج |
| – عدد محدود جداً من التعليمات الأساسية | – عدد كبير جداً من التعليمات الوظيفية تسهل البرمجة |

## 8-2 حول منهجية تنفيذ التجارب العملية (About the Laboratory Sessions):

في هذا المنهاج العملي التطبيقي سوف نعالج بشكل خاص برمجة المتحكمات المصغرة من العائلة AVR باستخدام لغة عالية المستوى وهي لغة BASCI والتي تعتمدها بيئة التطوير Bascom-AVR (IDE). كذلك سنقوم بمحاكاة جميع الأمثلة والتطبيقات في بيئة المحاكاة LabCenter Proteus الذي يعد من أقوى البرامج التي تحاكي عمل المعالجات. سنقوم بعدها بتنفيذ التجارب عملياً على لوحة التطوير mini-Phoenix المعدة خصيصاً لهذا المختبر والتي تم تصميمها بحيث ترتقي بالمتعلم من مستوى مبتدئ إلى مستوى متقدم متضمنةً أكثر من 25 تجربة تشمل جميع الوظائف الأساسية للمتحكمات بالإضافة إلى وظائف متقدمة أخرى.

## 9-2 بيئة المحاكاة (The Simulation Environment):

في مرحلة التصميم وقبل التطبيق العملي للأمثلة والتمارين على لوحة التطوير، فسيتم محاكاة برامج الوحدات المحيطية للوحة التطوير في البرنامج PROTEUS والذي هو عبارة عن بيئة مخصصة لأغراض محاكاة الأنظمة الرقمية والمعالجات المصغرة.



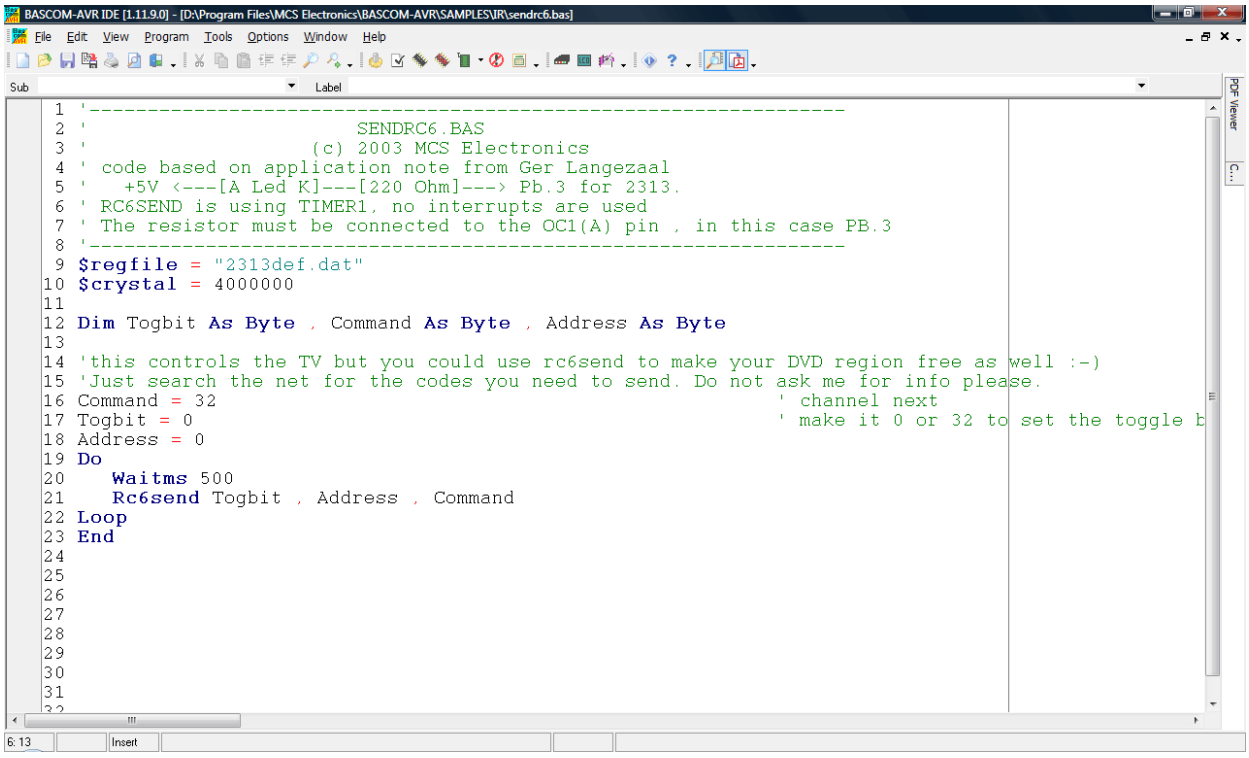

الشكل2 استخدام بيئة المحاكاة PROTEUS لتحليل دارة عداد تصاعدي تنازلي قابل للضبط

يعتبر برنامج PROTEUS من أقوى برامج المحاكاة للمتحكمات المصغرة، وهو يملك العديد من المكتبات التي تغطي جميع أنواع المحيطيات التي يمكن وصلها مع المتحكم المصغر بالإضافة إلى أدوات القياس العديدة، وسوف نستخدم هذا البرنامج لمحاكاة جميع التجاب التي سوف نتطرق إليها لاحقاً، كما سيتم استخدامه بشكل أساسي في المختبر الافتراضي.

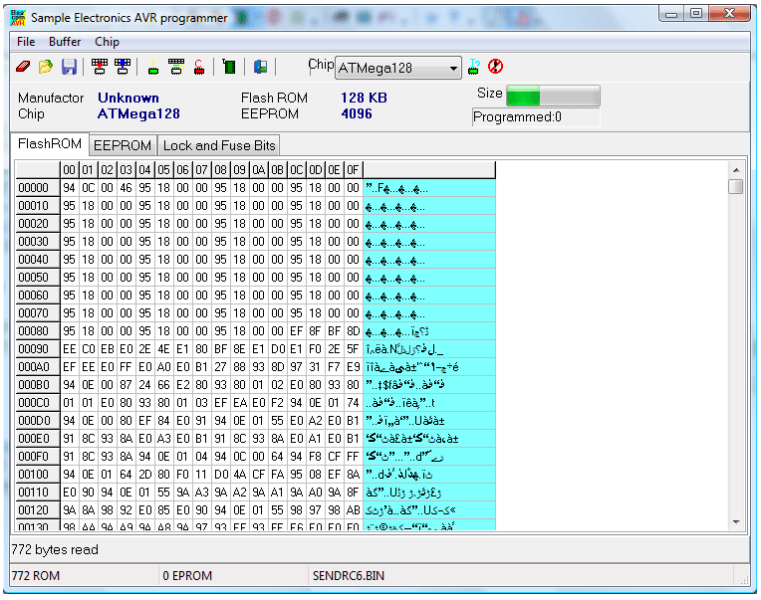
## 10-2 بيئة التطوير Bascom-AVR (The Integrated Development Environment):

تعتبر بيئة التطوير Bascom-AVR (IDE) من أشهر وأقوى بيئات التطوير البرمجية التي تستخدم لغة عالية المستوى وهي لغة BASCI لبرمجة المتحكمات المصغرة من العائلة AVR. تمتلك هذه البيئة واجهات تطبيقات متعددة وهي:

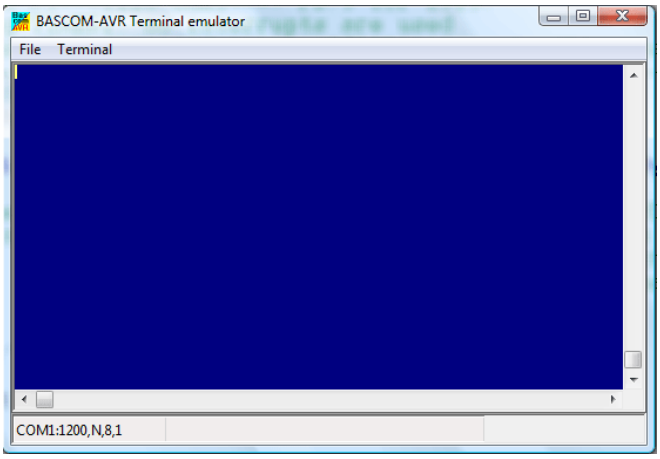
1. الواجهة البرمجية الرئيسية: وهي محرر التعليمات والأوامر البرمجية.



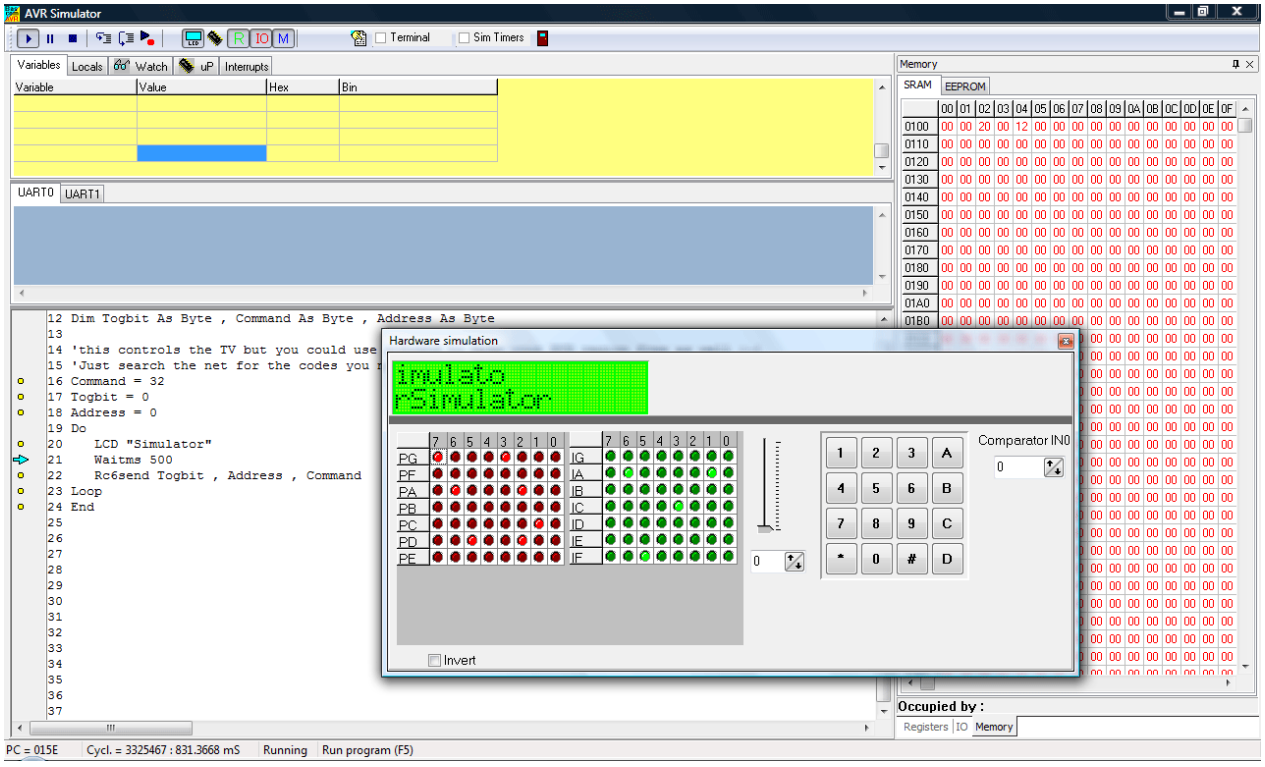

2. واجهة المبرمجة: وفيها يتم برمجة المعالج بعد إجراء عملية توليد الملف البرمجي بالأمر Compile.



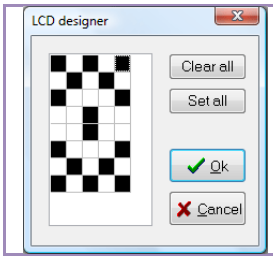
3. واجهة الربط البيني: وفيها يتم عرض المعلومات المرسلة والمتلقاة بين المعالج والحاسب بهدف مراقبة بارامترات النظام بشكل آني.



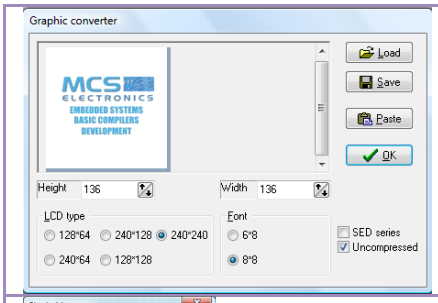

4. واجهة المحاكاة: وفيها يتم تشغيل البرنامج خطوة_خطوة ومراقبة حالة المسجلات الداخلية الذواكر.



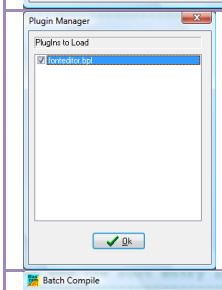

5. بالإضافة إلى الواجهات الأربعة يملك برنامج Bascom-AVR أدوات مساعدة وهي:



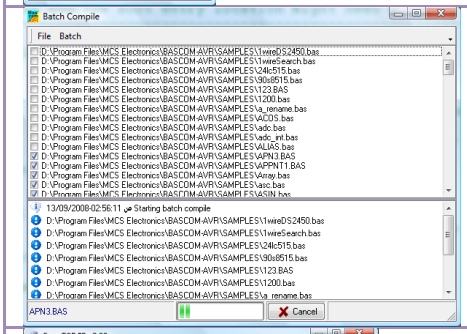

أداة تصميم المحارف (LCD Designer): وتستخدم لتصميم المحارف التي لا توجد على لوحة مفاتيح الحاسب من أجل إظهارها على شاشة الإظهار الكريستالية.



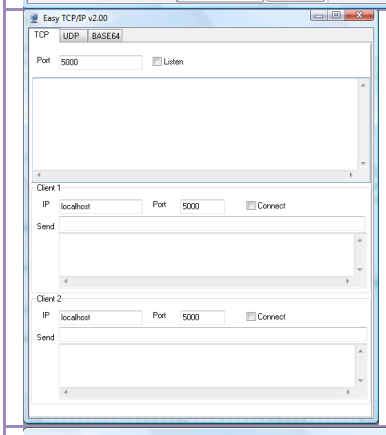

أداة تحويل الصور (Graphic Converter): وتستخدم لتحويل امتداد الصور المراد إظهارها على شاشة الإظهار الرسومية GLCD إلى الصيغة *.bgf



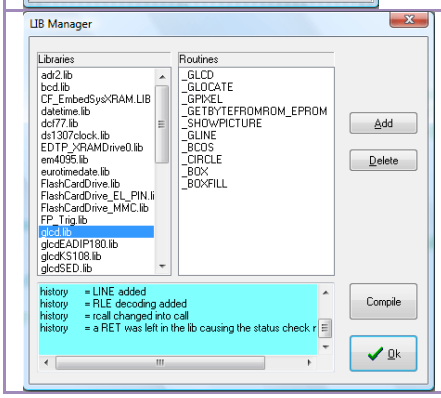

أداة مدير الإضافات (Plugin Manager): وتستخدم لإضافة/حذف الأدوات والموديولات الخارجية.



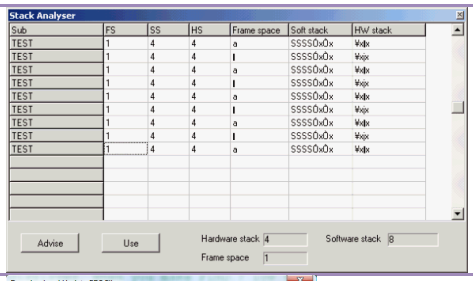

أداة مترجم الملفات المتعددة (Patch Compiler): وتستخدم لتوليد الملف البرمجي لعدة ملفات في آن واحد.



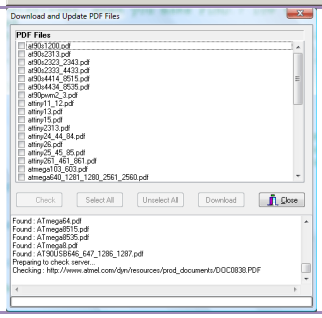

أداة التحكم بالبروتوكول TCP/IP: تستخدم للتحكم ومراقبة المعلومات الموجودة على خط المعطيات.




أدارة إدارة المكتبات (LIB Manager): وتستخدم لإدارة مكتبات البرنامج (حذف \ إضافة).




أداة محلل حالة المكدس (Stack Analyzer): وتستخدم لتحديد حجم المكدس المناسب للتطبيق.



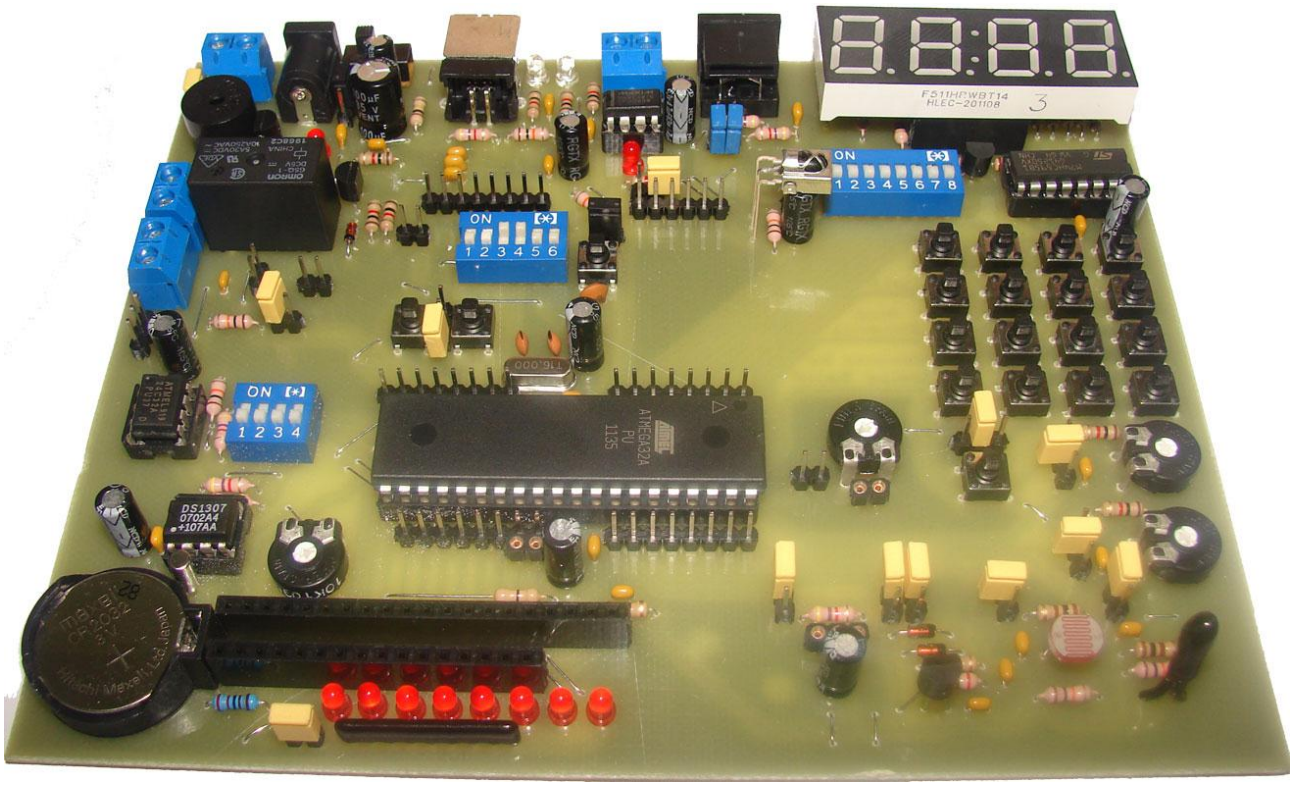

أداة ملفات الوثائق الفنية (PDF Update): تقوم هذه الأداة بالاتصال مع موقع شركة ATMEL وإحضار آخر تحديث للوثائق الفنية للمعالجات المستخدمة من العائلة AVR.

تتميز بيئة التطوير Bascom-AVR بمكتباتها الوظيفية القوية التي تمكن المطور أو المصمم من بناء نظام معقد بزمن قصير نسبياً مقارنة بباقي بيئات التطوير التي تستخدم اللغة C/C++.. الشكل3 يبين تصنيف أنواع التعليمات في البيئة Bascom-AVR حسب وظيفتها.

الشكل3 تصنيف أنواع التعليمات في البيئة Bascom-AVR حسب وظيفتها

## 11-2 لوحة التطوير المخبرية (The Laboratory Development Board):

لقد تم تصميم هذه اللوحة خصيصاً بحيث تخدم المبتدئ والمتقدم في تعلم برمجة المتحكمات المصغرة من العائلة AVR، حيث تضم أكثر من 25 وحدة محيطية على نفس اللوحة لتغطي ما يقارب 50 تجربة أساسية، وقد تصل إلى أكثر من 75 تجربة بالدمج بين الوظائف المحيطية على اللوحة، بالإضافة إلى إمكانية ربط وحدات خارجية عن طريق وحدات التوسعة المحيطية الموزعة على أطراف اللوحة. يمكن الرجوع إلى الملف "mini-Phoenix-AVR Manual.pdf" من أجل معرفة تفاصيل عن لوحة التطوير (طريقة تجميع اللوحة وميزاتها ومخططاتها التصميمية ...).



الشكل4 لوحة التطوير mini-Phoenix-AVR

## 12-2 مقدمة إلى متحكمات AVR (Introduction to AVR Microcontrollers):

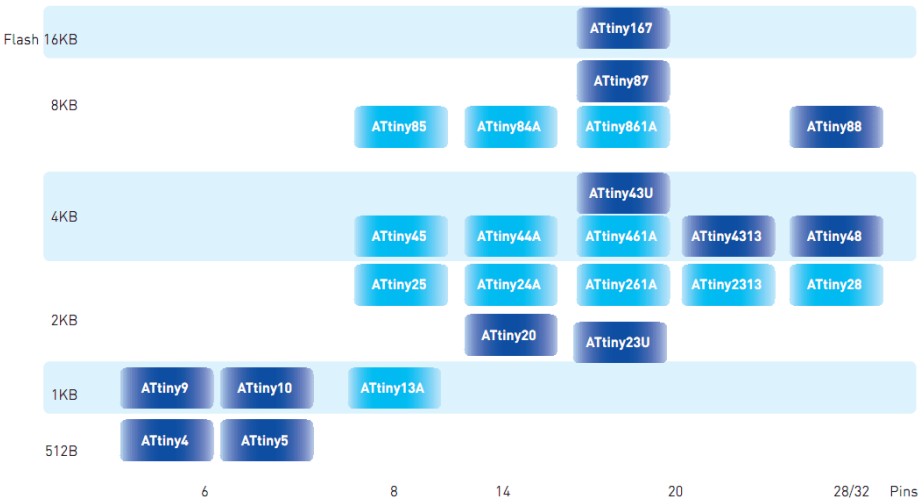
تعتبر متحكمات AVR إحدى منتجات شركة ATMEL الأمريكية، وقد تم تطويرها في مختبرات الشركة الموجودة في النرويج في أواخر التسعينيات، وتعتبر من أكثر المتحكمات المصغرة انتشاراً لما تتميز به من العديد من الميزات التي جعلتها مناسبة لكثير من التطبيقات.

لقد أحدثت شركة ATMEL ثورة في عالم المتحكمات المصغرة بإنتاجها لمتحكمات AVR التي تفوّقت بشكلٍ كبير على العديد من نظيراتها من متحكمات 8-bit، حيث تم استخدام البنية RISC التي تتميز بالأداء العالي وبالطاقة المنخفضة، واحتوت قائمة التعليمات في متحكمات AVR على 132 تعليمة – يُنفذ معظمها خلال دورة آلة واحدة (1-cycle) – وبالتالي عند وصل هزاز 16MHz إلى المتحكم فإنه سينفذ حوالي 16MIPS (مليون تعليمة في الثانية الواحدة)، كما زُوّدت هذه المتحكمات بذاكرة برنامج قابلة للمسح والكتابة لأكثر من 100000 مرة، وضمنت شركة ATMEL أن يبقى البرنامج داخل المتحكم يعمل بشكل صحيح حتى 25 سنة، كما تملك متحكمات AVR وحدات محيطية مدمجة متعددة الوظائف الأمر الذي يوفر استخدام دارات متكاملة خارجية، كذلك زُوّدت معظم متحكمات AVR بمبدل تشابهي رقمي متعدد الأقنية مدمج داخل المتحكم، إضافةً إلى إمكانية برمجة المتحكم دون فصلة عن النظام (In-system Programming)، وكذلك تتوفر في الأسواق بكميات كبيرة وسعرها منخفض مقارنة مع ميزاتها.

## 13-2 عائلات متحكمات AVR (AVR MCUs Families):

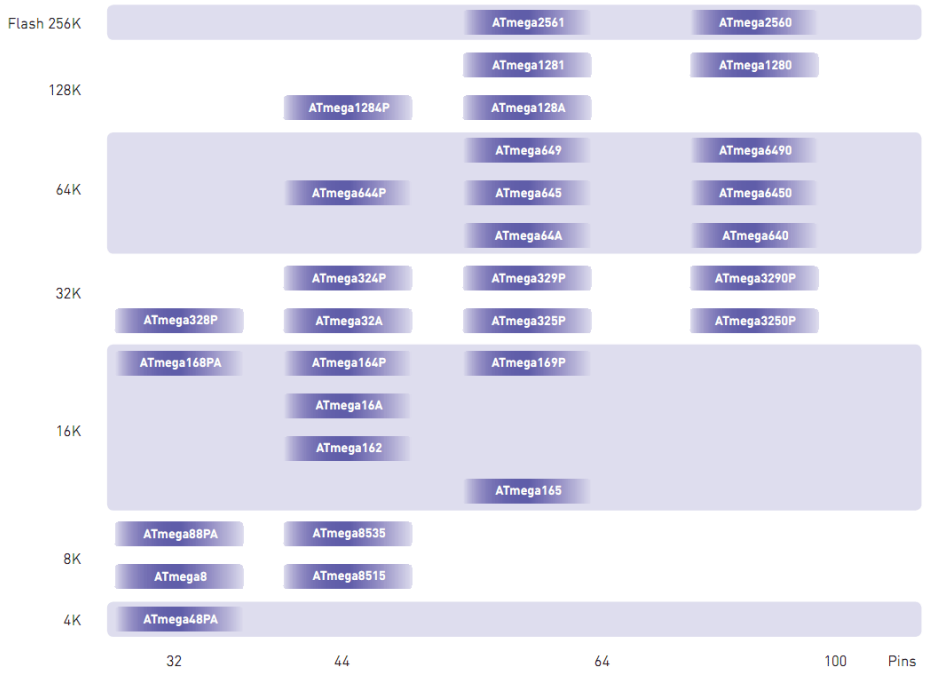
تقسم عائلات متحكمات AVR ذات عرض ناقل 8-bit إلى أربع مجموعات أساسية، إضافة إلى مجموعات أخرى ذات وظائف خاصة، تمتلك جميعها نفس البنية وتختلف عن بعضها البعض بالميزات والخصائص الموجودة في كل نوع:

- ✓ العائلة AT90Sxxxx: العائلة الكلاسيكية التي كان منها الانطلاقة الأولى لمتحكمات AVR في عام 1997 وقد توقف تصنيعها.
- ✓ العائلة ATtinyxx: وهي العائلة الصغرى لمتحكمات AVR المطورة والتي ظهرت في أوائل عام 2000، وهي تملك عدد أقطاب قليل (6~32pin) وحجم ذاكرة برنامج صغير نسبياً (0.5~16KB) وموارد محدودة على الشريحة الأمر الذي يجعل سعرها منخفض مقارنة مع متحكمات ATmega.



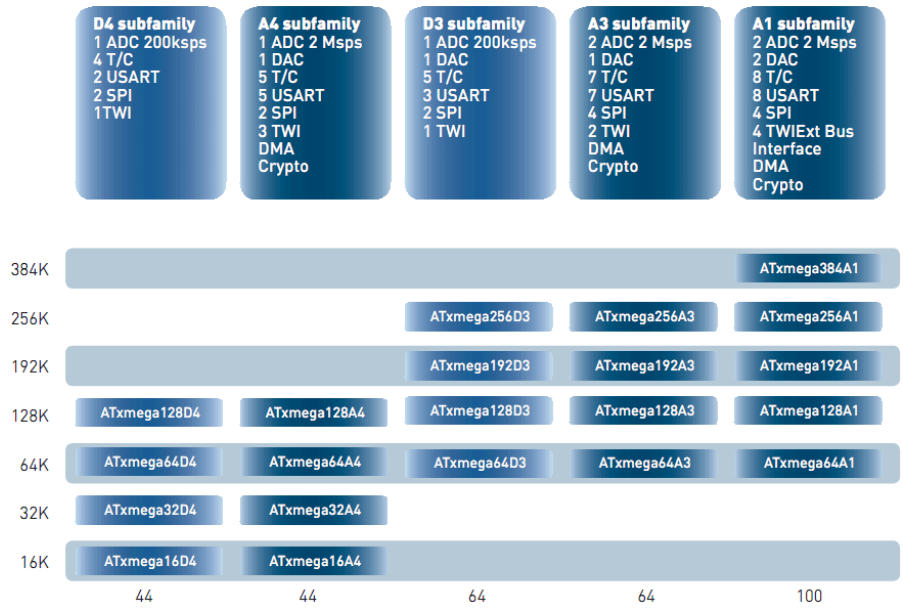

الشكل5 توزع متحكمات العائلة ATinyxxx وفقاً لعدد الأقطاب وحجم ذاكرة البرنامج

✓ العائلة ATmegaxxxx: وهي العائلة الكبرى لمتحكمات AVR المطورة والتي ظهرت في أوائل عام 2000 وهي تملك عدد أقطاب كبير (28~100pin) وحجم ذاكرة برنامج كبير نسبياً (8~256KB) وموارد متنوعة على الشريحة.




الشكل6 توزع متحكمات العائلة ATmega وفقاً لعدد الأقطاب وحجم ذاكرة البرنامج

✓ العائلة ATxmegaxxxx: وهي العائلة المتطورة والأحدث لمتحكمات AVR وقد ظهرت في عام 2008 وهي تملك ميزات متنوعة وسعة معالجة كبيرة نسبياً وتعمل بترددات أعلى من سابقاتها (32MHz)... كما أنها تملك عدد أقطاب كبير (44~100pin) وحجم ذاكرة برنامج كبيرة (16~384KB) إضافة إلى ميزات جديدة لا تتوفر في سابقاتها.




الشكل7 توزع متحكمات العائلة ATxmega وفقاً لعدد الأقطاب وحجم ذاكرة البرنامج

✓ إضافةً إلى العائلات الأربعة الأساسية يوجد عائلات أخرى ذات وظائف وتطبيقات خاصة موضحة في الجدول أدناه.

| الصنف | الاستخدام |
|---|---|
| **Automotive AVR** | تستخدم في أنظمة التحكم بالمحركات وأنظمة التحكم بالسيارات |
| **Z-Link AVR** | تستخدم في بروتوكولات الإرسال الراديوي اللاسلكي في IEEE 802.15.4 / ZigBee |
| **Battery Management AVR** | تستخدم للتحكم في شحن المدخرات ومراقبتها وهي تعمل في جهود مرتفعة 1.8~25 فولت |
| **CAN AVR** | تستخدم للتحكم بالبروتوكول CAN وتدعم: OSEK، DeviceNet،CANopen |
| **LCD AVR** | تستخدم كمعالجات تشغيل أساسية لشاشات الإظهار الكريستالية LCD |
| **Lighting AVR** | تستخدم في تطبيقات التحكم الاستطاعية بسرعة المحركات وشدة الإضاءة |
| **USB AVR** | تستخدم كموزع أو مخدم للبروتوكول USB |

الشكل8 بين ملخصاً للعائلات والخصائص الأساسية لكل منها...

| BatteryM AVR | Lighting AVR | USB AVR | megaAVR | tinyAVR |
|---|---|---|---|---|
| 18 ~ 48 Pin | 24 ~ 32 Pin | 32 ~ 64 Pin | 28 ~ 100 Pin | 8 ~ 32 Pin |
| MAX I/O 4~18 | MAX I/O 19~27 | MAX I/O 22~48 | MAX I/O 23~86 | MAX I/O 6~28 |
| 4KB~40KB Flash | 8KB~16KB Flash | 8KB~128KB Flash | 4KB~256KB Flash | 1KB~8KB Flash |
| 256B~1KB EPROM | 512B EPROM | 512B~4KB EPROM | 512B~4KB EPROM | 64B~512B EPROM |
| 512B~2KB SRAM | 512B~1KB SRAM | 512B~8KB SRAM | 512B~16KB SRAM | 32B~512B SRAM |
| Up To 8MIPS | Up To 16MIPS | Up To 16MIPS | Up To 20MIPS | Up To 20MIPS |
| 1.8V – 25V | 2.7V – 5.5V | 2.7V – 5.5V | 1.8V – 5.5V | 1.8V – 5.5V |

| Automotive AVR | CAN AVR | LCD AVR | AVR Z-Link | xmegaAVR |
|---|---|---|---|---|
| 14 ~ 64 Pin | 64 Pin | 64 ~ 100 Pin | MCU Wireless | 44 ~ 100 Pin |
| MAX I/O 6~54 | 32KB~128KB Flash | MAX I/O 54~69 | chipset for: | MAX I/O 36~78 |
| 2KB~128KB Flash | 1KB~4KB EPROM | 16KB~64KB Flash | IEEE 802.15.4 | 16KB~384KB Flash |
| 128B~4KB EPROM | 1K~4KB SRAM | 512B~2KB EPROM | and | 1KB~4KB EPROM |
| 128B~4KB SRAM | Up To 16MIPS | 1KB~4KB SRAM | ZigBee | 2KB~32KB SRAM |
| Up To 16MIPS | 2.7V – 5.5V | Up To 20MIPS | applications. | Up To 32MIPS |
| 2.7V – 5.5V | | 1.8V – 5.5V | | 1.8V – 3.6V |

الشكل8 الخصائص العامة لعائلات متحكمات AVR

## 14-2 مقارنة بين أشهر عائلات المتحكمات المصغرة (Comparison between most famous µC families):

الجدول التالي يبين مقارنة بين أشهر عائلات متحكمات 8-bit.

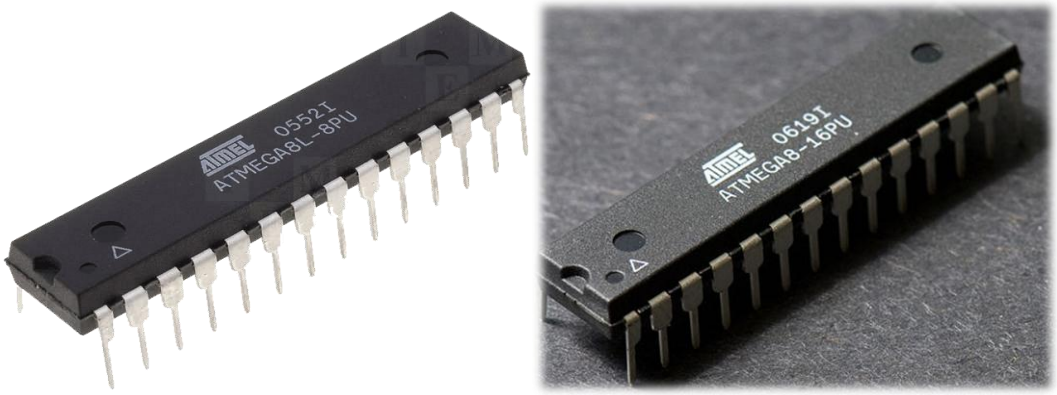
| **الميزة / العائلة** | AVR (Atmel) | 8051 (Intel) | PIC (Microchip) | HC11 (Motorola) |
|---|---|---|---|---|
| البنية الأساسية | Harvard | Von-Neumann | Harvard | Von-Neumann |
| تقنية النواة | RISC | CISC | RISC | CISC |
| تردد التشغيل الأعظمي | 20MHz | 24MHz | 20MHz | 8MHz |
| نبضة لكل تعليمة | 1 | 12 | 4 | 8 |
| تعليمة في الثانية | 16MIPS | 2MIPS | 5MIPS | 1MIPS |
| عدد التعليمات | 132 | 215 | 32 | 200 |
| حجم ذاكرة البرنامج | 256KB | 32KB | 64KB | 32KB |
| عرض ناقل التعليمات | 16-bit | 8-bit | 12-bit | 8-bit |

من خلال قراءة الجدول نستنتج أفضلية متحكمات AVR للأسباب التالية:

- ✓ متحكمات AVR أسرع من متحكمات PIC بأربع مرات، وأسرع من متحكمات 8051 بثمانية مرات!
- ✓ متحكمات AVR تملك ذاكرة برنامج ذات حجم أكبر من باقي العائلات مما يمكن من كتابة برامج ضخمة.
- ✓ متحكمات AVR مبنية بالاعتماد على تقنية Harvard التي تقوم على الفصل بين ذاكرة البيانات وذاكرة التعليمات بحيث يكون لكل من الذاكرتين خطوط عنونة منفصلة (عناوين فيزيائية مستقلة) وكذلك الأمر بالنسبة لخطوط التحكم وممر المعطيات، الأمر الذي يمكن من أن تحدث عملية قراءة التعليمات مع قراءة أو كتابة البيانات في نفس اللحظة، وكذلك يتيح لطول كلمة البيانات أن يكون مختلفاً عن طول كلمة التعليمات بسبب عدم اشتراك البيانات والتعليمات في نفس الذاكرة. بالمقارنة مع البنية von-Neumann فإن هذه الأخيرة منظمة بحيث لا يوجد فصل بين ذاكرة التعليمات وذاكرة البيانات ولهما نفس خطوط العنونة ونفس ممر المعطيات، بالتالي فإن الفائدتين اللتان تم ذكرهما سابقاً لا توفرهما البنية von Neumann مما يجعل بنية Harvard ذات أداء أعلى من حيث سرعة المعالجة و تنفيذ البرنامج.
- ✓ متحكمات AVR تملك نواة من التقنية RISC التي تمكن من إنجاز تعليمة خلال دورة هزاز واحدة بخلاف التقنية CISC التي تحتاج عدة دورات هزاز لتنفيذ تعليمة واحدة. كذلك فإن البنية RISC أقل تكلفة من البنية CISC.

## 15-2 قراءة تكويد معالجات العائلة AVR (Reading AVR Package information):

بشكل عام تزود الشرائح المتكاملة والعناصر الإلكترونية برقم أود تكويد خاص بكل صنف. في هذه الفقرة سنشرح تكويد شركة Atmel لمعالجات العائلة AVR بحيث يمكننا معرفة الكثير عن الشريحة من خلال التكويد الموجود على غلافها الخارجي. لنأخذ على سبيل المثال المتحكم المصغر ATmega8 الموضح على الشكل9 نلاحظ أن الغلاف يحوي على رمز الشركة المصنعة (Atmel) ورقم تصنيع تسلسلي (0619I) خاص برقم الدفعة وبيئة عمل المعالج ومن ثم تكويد المعالج وهو: ATmegaa8-16PU.



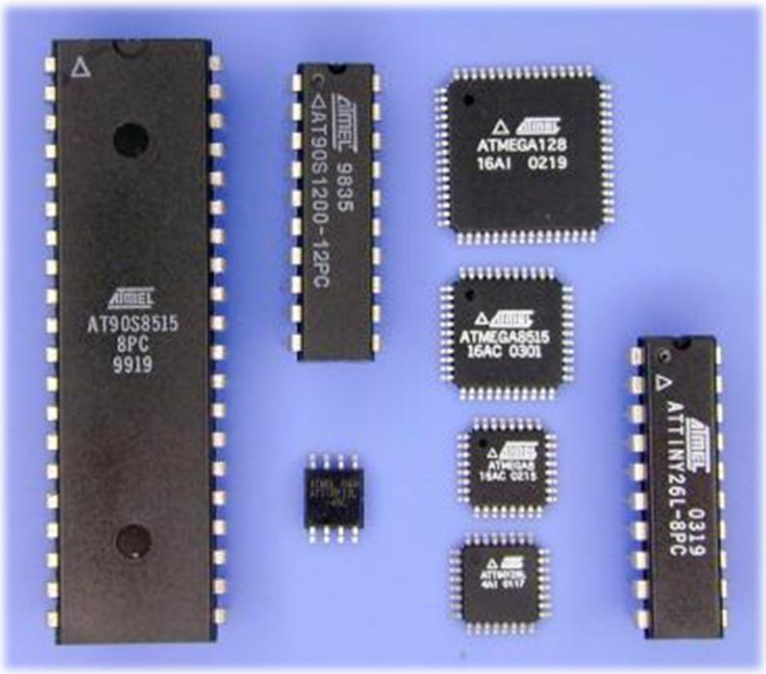

الشكل9 المتحكم المصغر ATmegaa8-16PU

بالنسبة لرمز بيئة عمل المعالج فهو يستخدم للدلالة على نوع التطبيق الذي يمكن أن ستخدم المعالج لأجله، فإما أن يكون تجارياً (**C**) أو صناعياً (**I**) أو عسكرياً (**M**) والاختلاف في ذلك هو من حيث قدرة المعالج على تحمل درجات الحرارة والضجيج العالي:

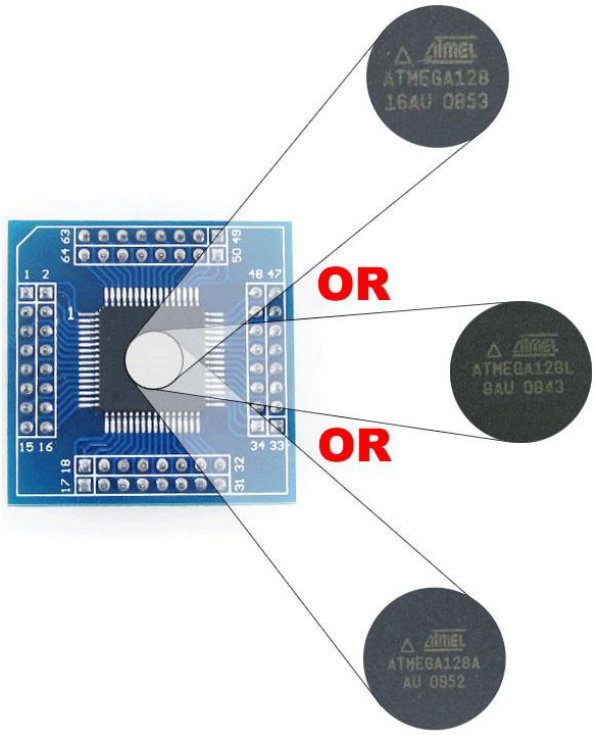
| **C** (Commercial) | **I** (Industrial) | **M** (Martial) |
|---|---|---|
| تطبيقات تجارية | تطبيقات صناعية | تطبيقات عسكرية |

بالنسبة لكود المعالج فهو على الشكل التالي:

| الرمز | الدلالة |
|---|---|
| **AT** | اختصار لاسم الشركة المصنعة ATMEL. |
| **Mega** | العائلة التي ينتمي إليها هذا المعالج (Xmega، Mega، Tiny،90S ). |
| **8** | هذا الرقم يعبر عن حجم ذاكرة البرنامج كيلو بايت (8، 16، 32، 64، 128، 256 KB ). |
| **L** | في حال أن الكود يملك حرف L فهذا يعني أنه قابل على العمل بجهود منخفضة (2.7V~5.5V) وبدون هذا الحرف فهذا يعني أن المعالج يعمل عند جهود (4.5V~5.5V). |
| **8** | تردد العمل الأعظمي للمعالج (8، 16، 20، 24، 36 ). |
| **PU** | شكل غلاف الشريحة، فإما أن تكون من نوع يتم لحامه على الطبقة السفلية للدارة المطبوعة ورمزه PU (PDIP)، أو من النوع السطحي الذي يتم لحامه على الطبقة العلوية للدارة المطبوعة ورمزه AU (TQFP). |




الشكل10 نموذج من متحكمات AVR بغلاف خارجي من النوع PDIP والنوع TQFP




من الجدير ذكره أن شركة Atmel قامت بدمج المعالج ذو اللاحقة ATmegaxxxL-8xx (يعمل عند جهد منخفض 2.7V وتردد منخفض 8MHz) مع المعالج ATmegaxxx-16xx (يعمل عند جهد 4.5V وتردد حتى 16MHz) في معالج جديد متطور ATmegaxxxA-xx يجمع خصائص كلا المعالجين؛ هذا الأخير يعمل عند جهود من 2V5 – 5V5 وتردد 1MHz – 16MHz وقد صدر هذا الإصدار من المتحكمات في عام 2010 ليحل محل الإصدارات القديمة ذات اللاحقة L-8 واللاحقة 16-.

الشكل جانباً يبين مقارنة بين الحالات الثلاث لتكويد المتحكم ATmega128 حيث:

ATmega128L-8AU (قديم لم يعد في طور التصنيع).

ATmega128-16AU (قديم لم يعد في طور التصنيع).

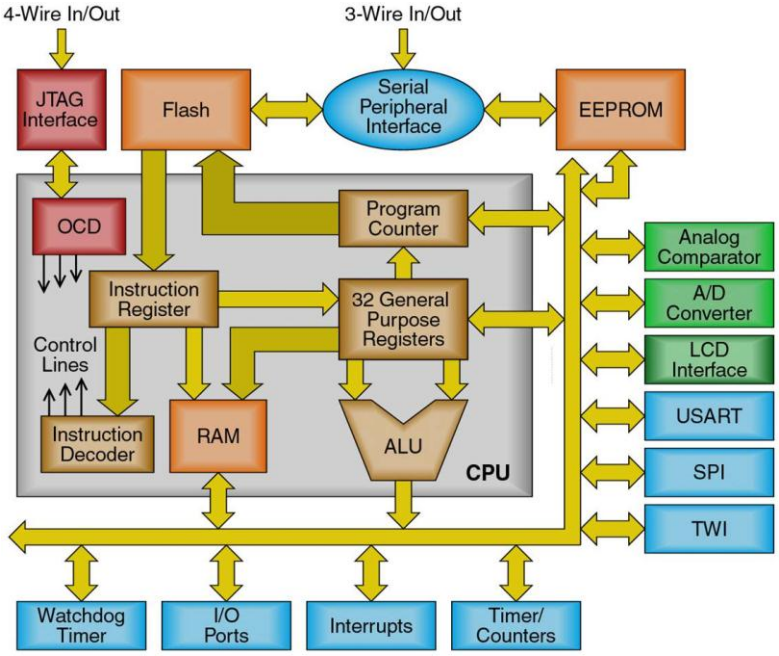
ATmega128A-AU (الجيل الجديد وهو موافق تماماً لسابقيه).

## 16-2 نواة معالجات العائلة AVR (AVR CPU Core):

تتمثل المهمة الرئيسية لنواة المعالج في ضمان تنفيذ البرنامج بشكل صحيح والذي بدوره يتطلب توفير إمكانية الولوج إلى الذواكر والقدرة على تنفيذ العمليات الحسابية والتحكم بالطرفيات... إضافة إلى التعامل مع المقاطعات المختلفة.

## 2-17 نظرة عامة على البنية الداخلية للنواة (CPU Core Architectural Overview):

تعتمد العائلة AVR البنية Harvard بهدف زيادة مستوى الأداء والعمل المتوازي – بحيث تكون هناك ذاكرة مخصصة للبيانات مع ممر معطيات خاص وذاكرة أخرى منفصلة لتعليمات البرنامج مع ممر خاص أيضاً. يتم تنفيذ التعليمات في ذاكرة البرنامج بمستوى واحد من المعالجة التفرعية بحيث أنه في الوقت الذي تقوم فيه وحدة المعالجة المركزية (CPU) بتنفيذ إحدى التعليمات يتم إحضار شفرة التعليمة التالية لها من ذاكرة البرنامج؛ إن هذا المبدأ يسمح بتنفيذ تعليمة عند كل نبضة ساعة.




الشكل 11 مخطط البنية الداخلية لنواة متحكمات AVR

يرتبط مع وحدة الحساب والمنطق ALU (Arithmetic Logic Unit) ملف المسجل ذو الولوج السريع والذي يحتوي على 32 مسجلاً للأغراض العامة كل منها بطول 8-bit وبزمن ولوج قدره نبضة واحدة من نبضات الهزاز – ويقصد بالولوج السريع للملف المسجلات على أنه الولوج الذي يستغرق زمن قدره دورة ساعة واحدة؛ هذا يعني أنه خلال دورة ساعة واحدة تقوم وحدة الحساب والمنطق بتنفيذ عملية واحدة، فهي تقوم أولاً بإخراج المعاملين من ملف السجلات، ومن ثم تنفذ العملية، ومن ثم تعيد تخزين النتيجة في ملف المسجلات، وكل ذلك يتم خلال دورة ساعة واحدة.

## 2-18 وحدة الحساب والمنطق (Arithmetic Logic Unit):

تدعم وحدة الحساب والمنطق العمليات الحسابية والمنطقية بين محتوى المسجلات أو بين محتوى المسجل وقيمة ثابتة، كما يمكن إجراء عملية معينة على محتوى مسجل وحيد، وبعد تنفيذ العملية يتم تحديث مسجل الحالة ليعطي المعلومات عن ناتج تلك العملية. تتمتع وحدة الحساب والمنطق ALU عالية الأداء في العائلة AVR بالارتباط المباشر مع مسجلات العمل الاثنين والثلاثين ذات الأغراض العامة، فخلال دورة ساعة واحدة تنفذ وحدة الحساب والمنطق عملية ما بين مسجلين في ملف المسجلات أو بين مسجل وقيمة ثابتة.

ويمكن تقسيم عمليات ALU إلى ثلاث فئات رئيسية: فئة حسابية – فئة منطقية – فئة العمليات على مستوى البت (Bit). كما تملك معالجات العائلة AVR في بنيتها ضارب متطور يدعم عمليات ضرب للأعداد المؤشرة وغير المؤشرة والاعداد الكسرية.

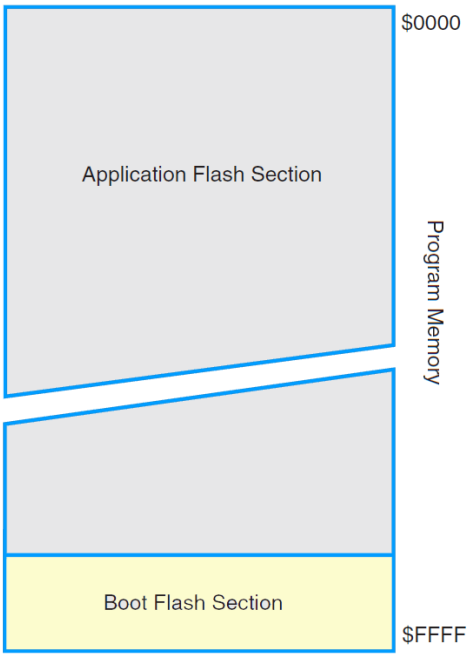
## 19-2 الذواكر في متحكمات AVR (Memories in AVR MCUs):

تمتلك معظم متحكمات AVR ثلاث أنواع من الذواكر المدمجة على الشريحة وهي: ذاكرة البيانات (Data Memory) – ذاكرة البرنامج (Flash Memory) – ذاكرة المعطيات (EEPROM).

### 19-2-1 ذاكرة البرنامج (Flash Memory):

يملك المتحكم ATmega32A ذاكرة وميضية من النوع Flash مدمجة ضمن الشريحة بحجم 32KB ($7FFF - $0000) مخصصة لتخزين برنامج عمل المتحكم المصغر، وبما أن جميع تعليمات متحكمات AVR هي ذات شفرة وحيدة بطول كلمة واحدة (16-bit) أو كلمتين (32-bit)، فقد تم تنظيم ذاكرة البرنامج لتشكل 16K x 16، كما تم تقسيمها إلى قسمين منفصلين: قسم برنامج الإقلاع (Boot Program) وقسم برنامج التطبيق (Application Program) وذلك بهدف الحماية من سرقة البرنامج.

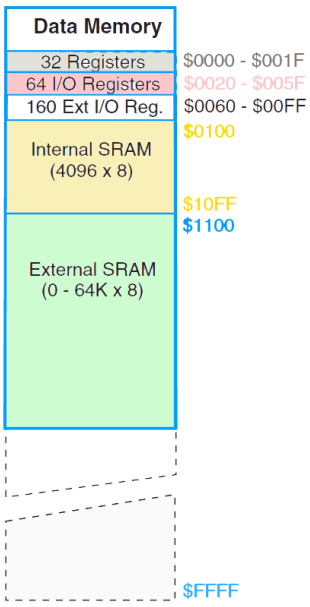
تتميز ذاكرة البرنامج في متحكمات AVR بديمومة 10,000 دورة مسح/كتابة على الأقل، كما أنها قابلة لإعادة البرمجة في النظام دون الحاجة إلى نقل المتحكم إلى مبرمجة خاصة وهو ما يدعى بـ"In System Programing". كذلك يملك المتحكم ATmega32A عداد برنامج (PC) بطول 16-bit (وهو متغير الطول بالنسبة بكل معالج) وبذلك يستطيع عنونة كامل مجال ذاكرة البرنامج الذي هو بطول 16KB x 16bit.




الشكل12 خريطة ذاكرة البرنامج

### 19-2-2 ذاكرة البيانات SRAM (SRAM Data Memory):

وهي عبارة عن ذاكرة وصول عشوائي (Static Random Access Memory) مؤقتة يتم فيها إجراء العمليات على المتحولات؛ يملك المتحكم ATmega32A ذاكرة بيانات من النوع SRAM بطول 2KB.




الشكل13 تنظيم الذاكرة الدخلية SRAM للمتحكم ATmega128A

### 19-2-3 ذاكرة المعطيات EEPROM (EEPROM Data Memory):

يملك المتحكم ATmega32A ذاكرة معطيات من النوع EEPROM بطول 1KB وهي منظمة مستقل لتخزين البيانات، ويمكن فيه القراءة من أو الكتابة على أي بايت من بايتات هذه الذاكرة بشكل مستقل، كما تسمح ديمومة الذاكرة EEPROM بأكثر من 100,000 عملية كتابة ومسح.

## 20-2 مصادر التوقيت في متحكمات AVR (Clock Sources in AVR MCUs):

تملك متحكمات AVR العديد من مصادر إشارة التوقيت والتي تقسم بشكل رئيسي إلى خمسة مصادر:

- هزاز كريستالي/سيراميكي خارجي (External Crystal/Ceramic Resonator).
- هزاز كريستالي خارجي ذو تردد منخفض (External Low-frequency Crystal).
- هزاز RC خارجي (External RC Oscillator).
- هزاز RC داخلي معاير (Calibrated Internal RC Oscillator).
- إشارة توقيت خارجية (External Clock).

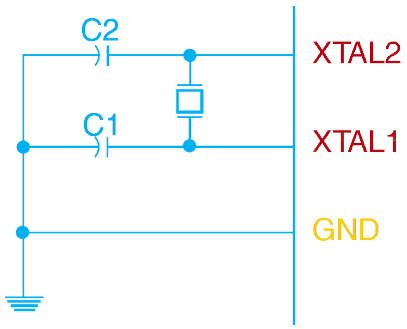
يتم ضبط مصدر إشارة التوقيت من خلال ضبط الفيوزات الداخلية للمتحكم والمسؤولة عن مصدر التوقيت (CKSEL3:0). لمزيد من التفاصيل يمكن الرجوع إلى الوثيقة الفنية للمتحكم/الفقرة "8.2 - Clock Sources".

### 20-2-1 الهزاز RC الداخلي للمتحكم (Calibrated Internal RC Oscillator):

تملك معظم متحكمات AVR وخصوصاً العائلة ATmega هزاز RC داخلي معاير مخبرياً بتردد ثابت يمكن ضبطه من أجل القيم 1MHz, 2MHz, 4MHz, 8MHz، وإن الحالة الافتراضية لمصدر إشارة التوقيت لمعظم متحكمات AVR-Mega هو "هزاز RC داخلي معاير" بتردد 1MHz ويمكن تغييره من خلال الفيوزات الداخلية للمتحكم الخاصة بمصدر إشارة التوقيت (CKSEL3:0).

### 20-2-2 وصل هزاز كريستالي خارجي (External Crystal Oscillator):

من أجل دقة أعلى لإشارة التوقيت، أو من أجل تشغيل المعالج عند تردد عمل أكبر من 8MHz، فإنه يمكن استخدام هزاز كريستالي خارجي يتم وصله مع المتحكم عبر القطبين XTAL1 (الدخل) والقطب XTAL2 (الخرج) مع مراعات إضافة مكثفات تحميل خارجية بقيمة 12-22pF كما هو موضح على الشكل14. من الضروري إعادة ضبط الفيوزات الداخلية للمتحكم والخاصة بمصدر إشارة التوقيت (CKSEL3:0) على "External Crystal Resonator".




الشكل14 وصل هزاز كريستالي خارجي مع المتحكم المصغر

### 20-2-3 وصل هزاز كريستالي خارجي ذو تردد منخفض (External Low-frequency Crystal):

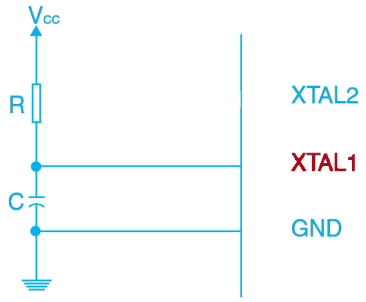
يمكن وصل هزاز كريستالي خارجي بتردد منخفض 32.768KHz كمصدر توقيت مخصص لتطبيقات الساعات الرقمية بعد إعادة تعيين الفيوزات الداخلية للمتحكم الخاصة بمصدر إشارة التوقيت (CKSEL3:0) على "External Low-frequency Crystal" حيث CKSEL = "1001".

### 20-2-4 وصل هزاز RC خارجي (External RC Oscillator):

من أجل التطبيقات التي لا تحتاج إلى دقة في إشارة التوقيت، فإنه يمكن استخدام دارة RC لتوليد إشارة التوقيت (100Hz~12MHz) كما هو مبين على الشكل15. القيمة التقريبية للتردد المولد يمكن حسابها من المعادلة:

$$f = \frac{1}{3 \times R \times C}$$

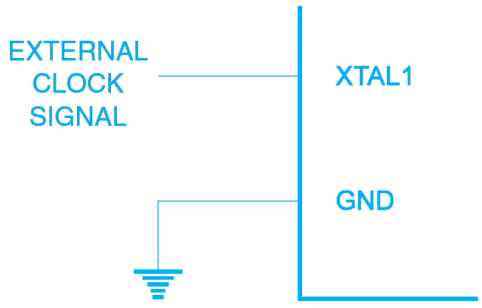
حيث أن قيمة C يجب أن تكون أكبر من 22pF. كما أن القيمة الأعظمية للتردد f يجب أن لا تتجاوز 12MHz. يجب ضبط الفيوزات الداخلية للمتحكم الخاصة بمصدر إشارة التوقيت (CKSEL3:0) على "External RC Oscillator".




الشكل15 وصل هزاز RC خارجي مع المتحكم المصغر

### 20-2-5 إشارة توقيت خارجية (External Clock):

يمكن كتطبيق إشارة توقيت خارجية مباشرة على القطب XTAL1 بعد إعادة تعيين الفيوزات الداخلية للمتحكم الخاصة بمصدر إشارة التوقيت (CKSEL3:0) على"External Clock" حيث "0000" = CKSEL. الشكل16 يبين طريقة تطبيق الإشارة.




الشكل16 تطبيق إشارة توقيت خارجية للمتحكم

## 2-21 الدراسة التطبيقية – المتحكم المصغر ATmega32A (Case Study – Atmega32A):

السبب في اختيار المعالج ATmega32 هو أنه يعتبر من المعالجات المتقدمة في العائلة ATmegaxxx ويضم معظم الميزات والوحدات المحيطية المتوفرة في العائلة Mega وبالتالي فإننا ندرس الآن الحالة الأعم والأشمل، كما أنه متوفر بغلاف من النوع PDIP.

بشكل عام وعند شراء دارة متكاملة ما، فإن أول ما يحتاج إليه هو معرفة خصائصها وميزاتها وكيفية عملها وتوزع أقطابها، وهذا يتم من خلال قراءة المعلومات المهمة من الوثيقة الفنية (Datasheet) الخاصة بالدارة المتكاملة. وهنا أود التنويه إلى أنه عند استخدام لغات البرمجة عالية المستوى فإن المبرمج لن يحتاج إلى قراءة الوثيقة الفنية للمتحكم المصغر كاملةً من أجل برمجته، وهذا بدوره يختصر وقتاً كبيراً في تعلم برمجة المتحكمات المصغرة بدون اللجوء إلى دراسة البنية الداخلية للمعالج مفصلةً كما هو الحال عند البرمجة بلغة التجميع (Assembly). لذلك، سوف أشرح المعلومات التي تفيدنا في الوثيقة الفنية كالميزات الأساسية للمعالج وتوزع الأقطاب، مع العلم أنه لابد – لاحقاً – من العودة إلى بعض التفاصيل في البنية الداخلية للمعالج عن مرحلة متقدمة.

## 2-22 الميزات الأساسية للمتحكم ATmega32A (ATmega128A Features):

الميزات الأساسية تأتي دائماً في الصفحة الأولى من الوثيقة الفنية لأي دارة متكاملة... في ما يلي ميزات المعالج ATmega32A.

- ✓ متحكم 8-bit بأداء عالٍ واستهلاك منخفض للطاقة.
- ✓ بنية متطورة من النوع RISC (أقل عدد ممكن من التعليمات):
  - 133 تعليمة معظمها تنفذ بدورة آلة واحدة
  - 32 x 8 مسجلات أغراض عامة + ومسجلات تحكم محيطية
  - عمل مستقر ومناعة ضد الضجيج
  - قادر على تنفيذ 16 مليون تعليمة في الثانية عند تردد 16MHz
  - يحوي على مضاعف دورة العمل
- ✓ ذاكرة معطيات دائمة:
  - 32KB ذاكرة برنامج يمكن برمجتها بدون فصل المعالج عن الدارة، قابلة للمسح والكتابة 10000 مرة.
  - أقفال برمجية مستقلة مع قطاع مخصص لكود إقلاع.
  - 1KB ذاكرة معطيات دائمة EEPROM قابلة للمسح والكتابة 100000 مرة.
  - 2KB ذاكرة وصول عشوائي مؤقتة SRAM
  - إمكانية عنونة 64KB (وصل) ذاكرة برنامج خارجية
  - أقفال برمجية من أجل حماية البرنامج على الشريحة
  - واجهة ربط تسلسلية (SPI) من أجل برمجة المعالج دون فصله
- ✓ واجهة اختبار (JTAG):

- قابلية مسح المسجلات الداخلية للمعالج وقراءة حالاتها
- دعم متقصي أخطاء (Debug) شامل للشريحة
- إمكانية برمجة ذاكرة البرنامج وذاكرة المعطيات.

✓ الميزات المحيطية:

- مؤقت/عداد 8-bit عدد 2 مزود بأنماط مقسم ترددي وحادثة مقارنة
- مؤقت/عداد 16-bit موسع مزود بأنماط مقسم ترددي وحادثة مقارنة وحادثة مسك
- عداد الزمن الحقيقي مع هزاز مستقل
- أربعة قنوات خرج (PWM) تعديل عرض النبضة 16-bit مع إمكانية التحكم بالدقة من 2 وحتى 16 بت
- ثمان قنوات تبديل تشابهي/رقمي بدقة 10-bit
- قناتي مداخل تفاضلية للتبديل تشابهي/رقمي بدقة 10-bit مع دارة ربح 1x, 10x, or 200x
- نافذة اتصال تسلسلية ثنائية (I2C)
- نافذتي اتصال تسلسلي (USARTs) قابلة للبرمجة
- نافذة اتصال تسلسلية (SPI) بنمطي عمل قائد/تابع
- مؤقت مراقبة قابل للبرمجة مع هزاز مستقل
- نافذة مقارن تشابهي

✓ الميزات الخاصة للمعالج:

- تصفير عند وصل التغذية وكاشف انخفاض جهد التغذية للشريحة
- هزاز داخلي معاير
- مصادر مقاطعة خارجية وداخلية
- ستة أنماط لتخفيض الطاقة ولتخفيض ضجيج المبدل
- إمكانية تحديد تردد الهزاز الداخلي برمجياً
- إلغاء شامل لمقاومات الرفع الداخلية للبوابات

✓ عدد أقطاب الدخل/الخرج وشكل الغلاف الخارجي المعالج:

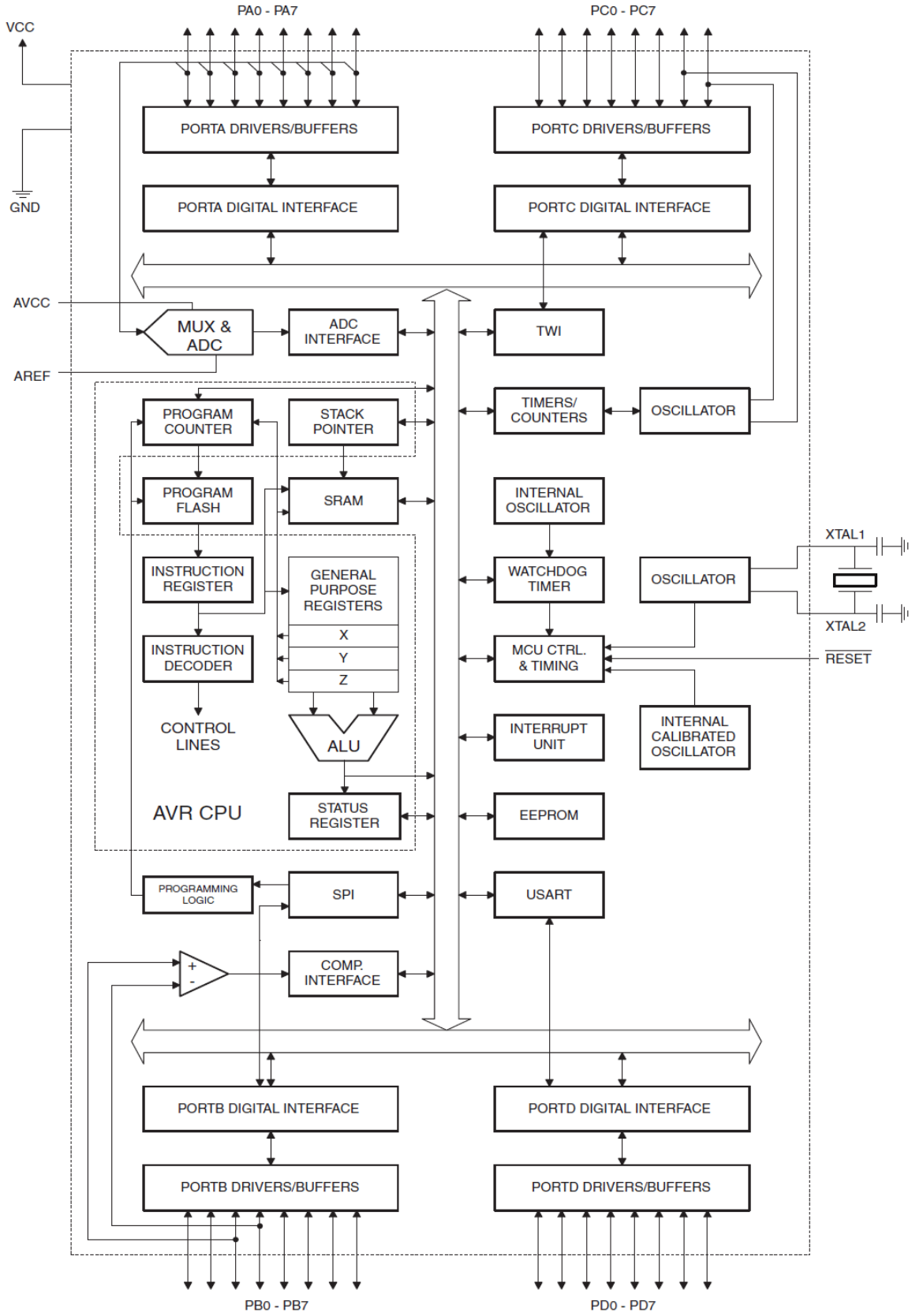
- 32 قطب دخل/خرج قابل للبرمجة متوفر من أجل شريحة 40 قطب بغلاف PDIP.

✓ جهود العمل للشريحة في المجال 2.7 - 5.5V

✓ تردد عمل أعظمي حتى 0 - 16MHz

✓ استهلاك الطاقة في النمط الفعال (Active) 0.6mA وفي نمط البطالة (Idle) 0.2mA وفي نمط الطاقة التحتية 1uA.

## 23-2 مخطط البنية الداخلية للمتحكم ATmega32A (ATmega32A Block Diagram):




الشكل17 المخطط الصندوقي للبنية الداخلية للمتحكم ATmega32A

لقد تم صناعة الشريحة ATmega32A باستخدام تقنية ذواكر ATMEL الغير قابلة للزوال ذات الكثافة العالية، مع إمكانية برمجة ذاكرة البرنامج الوميضية (Flash) المبنية على شريحة المتحكم إما من خلال الوصلة التسلسلية SPI أو باستخدام مبرمجة تفرعية أو باستخدام برنامج إقلاع موجود على الشريحة (Boot program) حيث تستطيع البرمجية المخزنة في جزء الإقلاع (Bootloader) في الذاكرة الوميضية متابعة عملها أثناء تحديث القسم الرئيسي في ذاكرة البرنامج. لقد أدى الجمع ما بين معالجات RISC ذات 8-bit مع ذاكرة البرنامج القابلة لإعادة البرمجة إلى إنتاج المتحكم ATmega132A الذي يتمتع بالقوة و المرونة العالية وبالكلفة المنخفضة للعديد من تطبيقات التحكم المتطورة. الشكل17 يبين المخطط الصندوقي للبنية الداخلية للمتحكم ATmega32 وهو يبين طريقة ربط الوحدات المحيطية والمسجلات مع وحدة المعالجة المركزية.

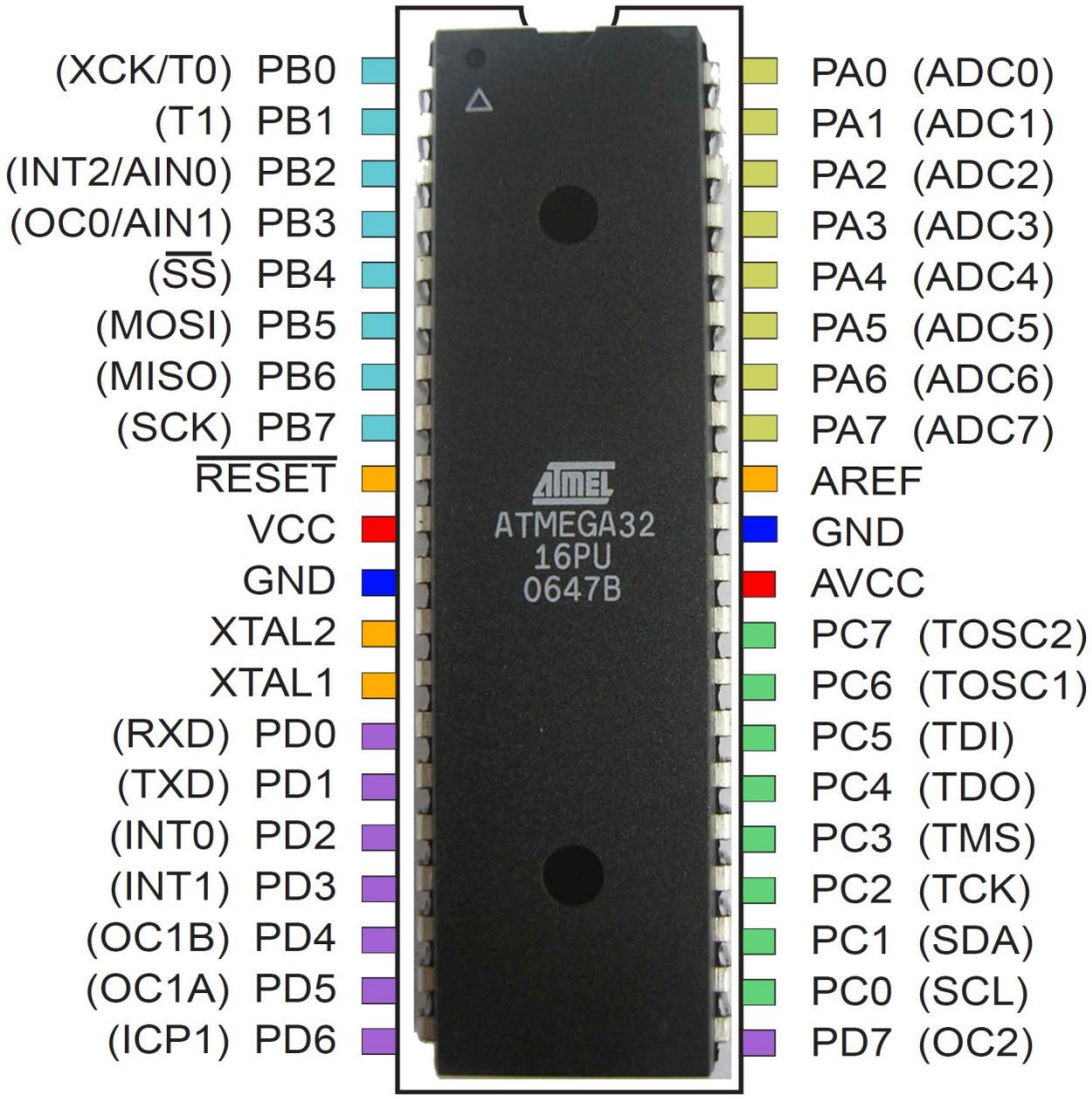
## 24-2 تصنيفات الأقطاب وظيفياً في متحكمات AVR (AVR MCUs Pins Functional Classification):

يمكن تصنيف أقطاب متحكمات العائلة AVR من حيث وظائفها على الشكل التالي:

1. أقطاب التغذية الرقمية (VCC, GND).
2. أقطاب التغذية التشابهية (AVCC, AGND).
3. أقطاب الدخل والخرج (Input / Output Pins = Ports).
4. أقطاب البرمجة:
   - الواجهة البرمجية ISP (MISO, MOSI, SCK, RESET).
   - الواجهة البرمجية PDI (PDO, PDI).
5. أقطاب المقاطعات الخارجية (INT0 – INT7).
6. أقطاب مصادر التوقيت (XTAL1, XTAL2, TOSC1, TOSC2).
7. أقطاب المؤقتات/العدادات (T0, T1, T2, ICP).
8. أقطاب المبدلات التشابهية الرقمية (ADC0 – ADC7).
9. أقطاب النوافذ التسلسلية:
   - النافذة التسلسلية UART / USART (XCK, TXD, RXD).
   - النافذة التسلسلية SPI (MISO, MOSI, SCK, SS).
   - النافذة التسلسلية I2C (SDA, SCL).
10. أقطاب المقارن التشابهي (AIN0, AIN1).
11. أقطاب إشارات PWM (OCA1, OCB1, OCC1, …).
12. أقطاب نافذة المراقبة JTAG (TDI, TDO, TMS, TCK).
13. أقطاب الوصل مع ذاكرة خارجية XRAM (AD0-7, A8-15, WR/RD).

## 25-2 وصف أقطاب المتحكم ATmega32A (ATmega32A Pin Description):

يملك المتحكم ATmega32A مجموعة من الأقطاب عددها 40 قطب موزعة على الأطراف الفيزيائية لشريحة المتحكم وهي:




الشكل18 توزع الأقطاب على الشريحة ATmega32A

| الاسم | الوظيفة |
|---|---|
| **VCC** | قطب جهد التغذية الموجب VCC = 2.5 – 5.5V |
| **GND** | قطب جهد التغذية الصفري (الأرضي) GND = 0V |
| **Port A** (PA7:PA0) | البوابة A: وهي عبارة عن بوابة دخل/خرج ذات ثمانية أقطاب ثنائية الاتجاه، وقد زودت الأقطاب بمقاومات رفع داخلية (Pull-up) مع إمكانية اختيار مقاومة الرفع لكل قطب على حدى. تمتلك البوابة A وظيفة ثانوية أخرى وهي قنوات المبدلات التشابهية الرقمية (ADC0-ADC7). |

| | |
|---|---|
| **Port B** (PB7:PB0) | البوابة B: وهي عبارة عن بوابة دخل/خرج ذات ثمانية أقطاب ثنائية الاتجاه، وقد زودت الأقطاب بمقاومات رفع داخلية (Pull-up) مع إمكانية اختيار مقاومة الرفع لكل قطب على حدى، كذلك تمتلك البوابة B وظائف ثانوية أخرى وهي: واجهة اتصال تسلسلية SPI (SS, SCK, MOSI, MISO)، وقطب توليد إشارة PWM (OC0)، وأقطاب المقارن التشابهي (AIN0, AIN1)، المقاطعة الخارجية (INT2)، العدادات (T0/T1). |
| **Port C** (PC7:PC0) | البوابة C: وهي عبارة عن بوابة دخل/خرج ذات ثمانية أقطاب ثنائية الاتجاه، وقد زودت الأقطاب بمقاومات رفع داخلية (Pull-up) مع إمكانية اختيار مقاومة الرفع لكل قطب على حدى. تمتلك البوابة C وظائف ثانوية أخرى حيث تعمل كنافذة تتبع أخطاء JTAG (TCK, TMS, TDO, TDI)، نافذة تسلسلية I2C (SCK, SDA). |
| **Port D** (PD7:PD0) | البوابة D: وهي عبارة عن بوابة دخل/خرج ذات ثمانية أقطاب ثنائية الاتجاه، وقد زودت الأقطاب بمقاومات رفع داخلية (Pull-up) مع إمكانية اختيار مقاومة الرفع لكل قطب على حدى. تمتلك البوابة D وظائف ثانوية أخرى وهي: المقاطعات الخارجية (INT0-INT1)، النافذة التسلسلية USART (TXD, RXD) ، مدخل حادثة المسك للموقت1 (ICP1). وأقطاب توليد إشارات PWM (OC2, OC1B, OC1A). |
| **RESET** | مدخل تصفير الشريحة؛ عند تطبيق إشارة كهربائية ذات منطق منخفض على القطب RESET لمدة دورتي آلة، فإن دارة التصفير الداخلية تعمل على تصفير المتحكم – عداد البرنامج PC = 0. |
| **XTAL1** | مدخل الهزاز الخارجي: وهو عبارة عن مدخل دارة مضخم الهزاز العاكس. |
| **XTAL2** | مدخل الهزاز الخارجي: وهو عبارة عن خرج دارةمضخم الهزاز العاكس. |
| **AVCC** | قطب التغذية للمبدل التشابهي الرقمي (ADC)، إذا كان المبدل ADC غير مُستخدم فإن هذا القطب يجب وصله إلى القطب Vcc، أما إذا كان المبدل ADC مُستخدماً فيوصل هذا القطب مع Vcc عن طريق مرشح تمرير مُنخفض. |
| **AGND** | إذا كانت الدارة التي نقوم بتصميمها لها أرضي تشابهي مستقل، فيجب ربط هذا القطب مع هذا الأخير، وإلا يُربط هذا القطب مع القطب الأرضي العام GND. |
| **AREF** | مدخل الجهد المرجعي التشابهي للمبدل ADC ويجب أن تتراوح قيمته عند عمل المبدل ما بين $A_{Vcc}$ – 2V. |

. . . ﴿انتهت الجلسة العملية الثانية﴾ . . .

وليد بليد

– دمتم بخير ومودة ونور –

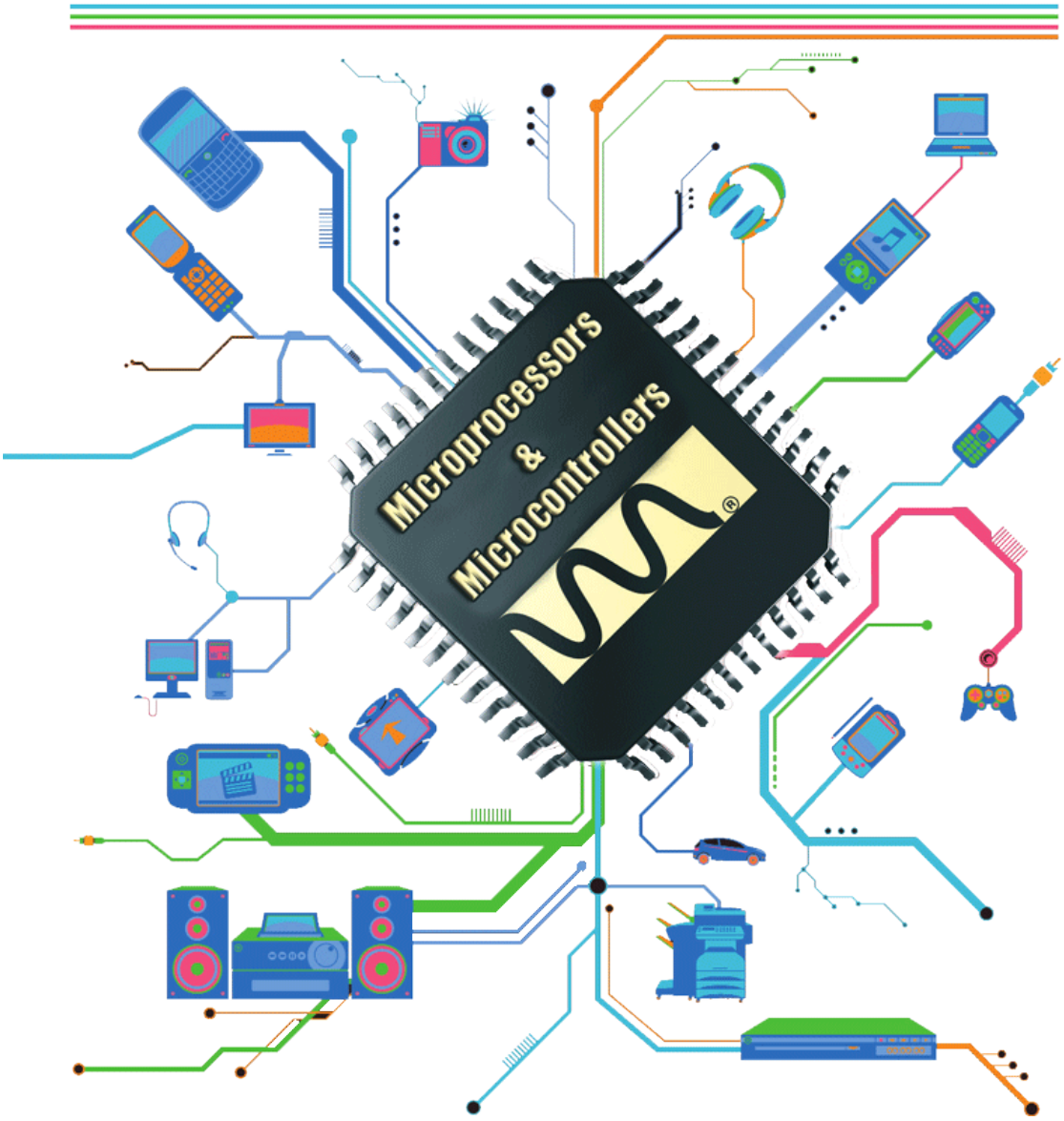
جامعـــــــــــة حلـــــــــــب
كلية الهندسة الكهربائية والإلكترونية
قسـم هندسـة التحكـم والأتمتـة

# ﴿ الجلسات العملية لمادة المعالجات والمتحكمات المصغرة ﴾

# Microprocessors & Microcontrollers Lab Sessions

السنة الثالثة | قسم اتصالات

## ﴿ الجلسة العملية الثالثة ﴾




م. وليد بليد

Copyright © 2012 Walid Balid - All rights reserved.

Wednesday, March 14, 2012

# الجلسة العملية الثالثة

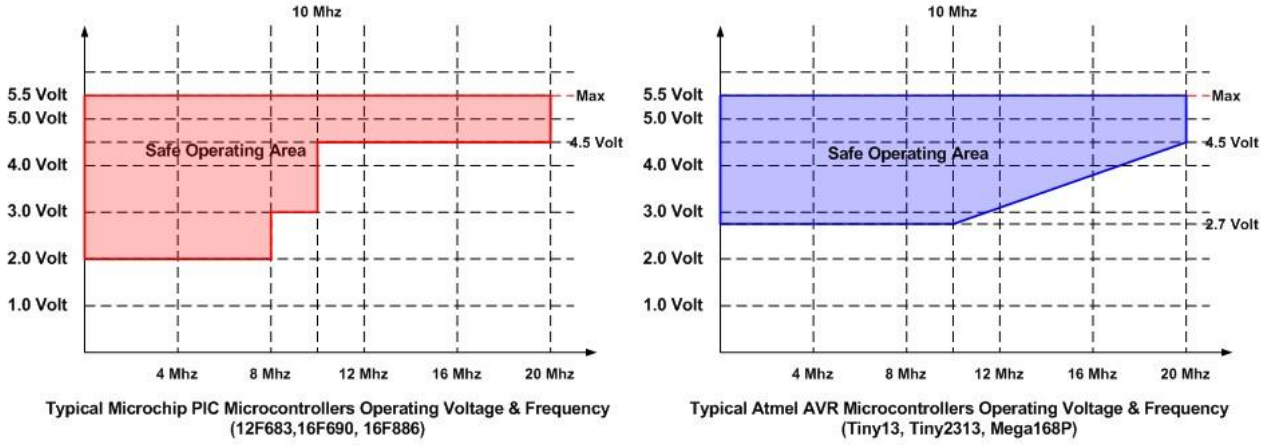
## نظرة عامة (Overview):

هذه المحاضرة تشرح بنية بوابات الدخل والخرج لمتحكمات AVR وتشرح المسجلات الداخلية لبوابات الدخل والخرج. ثم تقدم تطبيقاً عملياً لاستثمار أقطاب الدخل والخرج لمتحكمات AVR. وبرمجتها في البيئة BASCOM-AVR ومحاكاتها في البيئة Proteus.

## 1-3 العلاقة بين تغذية المتحكم وتردد التشغيل الأعظمي (Power vs. Frequency):

بقدر ما تكون التغذية الرئيسية – لأي دارة إلكترونية – مصممة بشكل جيد وفق اعتبارات تصميمية قياسية، بقدر ما يكون عمل العناصر الإلكترونية في الدارة مستقراً وقريباً من منحني العمل الأمثلي. إن التغذية الكهربائية التي توصل للمتحكم المصغر هي بمثابة الروح التي تبث الحياة والحركة في المتحكم المصغر، كما أن استهلاك التغذية في المتحكم يتعلق مباشرة بسرعة عمل المتحكم المصغر، حيث أنه كلما ازداد تردد عمل المعالج، ازداد استهلاك التغذية في المعالج.

الشكل3-1 بين منحني العمل الآمن للمعالج نسبة إلى التغذية المطبقة من أجل كل تردد عمل. من أجل متحكم مصغر من العائلة "AVR" فإن التغذية 4.5V ستؤمن عمل آمن للمعالج عند كامل مجال تردد الهزاز الكريستالي، أما من أجل جهد تغذية "3V" فإن أقصى سرعة عمل للمتحكم يجب أن لا تزيد عن "8MHZ" لكي يبقى المعالج ضمن منطقة العمل الآمنة.




الشكل3-1 منحني العمل الآمن للمعالج نسبة إلى التغذية المطبقة من أجل كل كامل مجال تردد التشغيل

## 2-3 اعتبارات قيم التشغيل الأعظمية (Powering MCU & the Maximum Ratings):

أحد أهم الاعتبارات التي يجب أن تؤخذ بعين الاعتبار عن ربط أقطاب المتحكم إلى الأحمال هو التيار الأعظمي المستهلك من قطب المتحكم (Vcc-to-Gnd). إن قيمة التيار التي يمكن سحبها أو تصريفها لقطب دخل/خرج من أقطاب المتحكم تتراوح عادة من

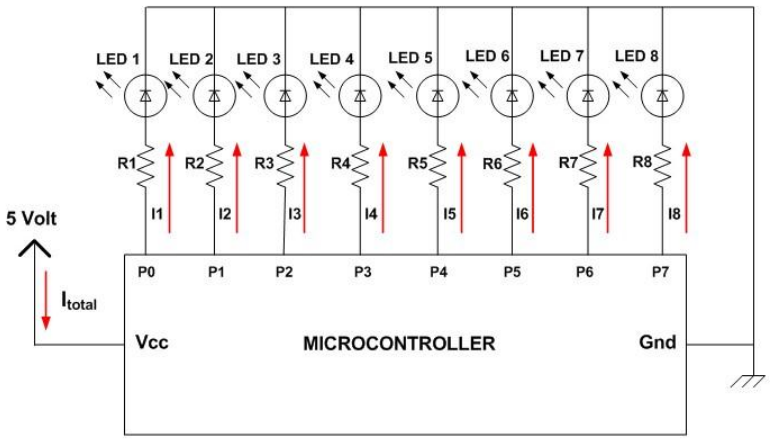
20~40mA حسب المواصفات الكهربائية للمتحكم المصغر. كما أن التيار الأعظمي الذي يمكن سحبه أو تصريفه عن طريق المتحكم بشكل كلي بالنسبة لمتحكمات AVR هو 200mA. الشكل3-2 يبين معدلات القيم الكهربائية الأعظمية لمتحكمات العائلة AVR.

**Absolute Maximum Ratings**

| | |
|---|---|
| Operating Temperature | -55·C to +125·C |
| Storage Temperature | -65°C to +150°C |
| Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground | -0.5V to $V_{CC}$+0.5V |
| Voltage on $\overline{\text{RESET}}$ with respect to Ground | -0.5V to +13.0V |
| Maximum Operating Voltage | 6.0V |
| DC Current per I/O Pin | 40.0 mA |
| DC Current $V_{CC}$ and GND Pins | 200.0 - 400.0mA |

الشكل3-2 معدلات قيم التشغيل الأعظمية لمتحكمات العائلة AVR

إن التيار الأعظمي الذي يمكن استجراره من المتحكم هو مجموع تيارات الأقطاب إضافةً إلى تيار التشغيل للمتحكم، وإن زيادة التيار فوق الحدود العظمى سوف يؤدي إلى عطل دائم في المتحكم ويتوجب بعدها تغييره. في الشكل3-3 تم استخدام ثمانية أقطاب من متحكم مصغر كأقطاب خرج لتشغيل ثنائيات ثمانية ضوئية.




Typical LEDs Display on the Microcontroller I/O Ports

الشكل3-3 توصيل ثنائيات ضوئية إلى أقطاب متحكم مصغر

إن التيار الأعظمي المسحوب من المتحكم هو مجموع تيارات الثنائيات الثمانية بالإضافة لتيار عمل المتحكم ويمكن حسابه بالشكل:

$$I_{total} = I_{operating\_current} + (8 \times I_{LED})$$

بافتراض أن جهد عمل الثنائي الضوئي هو "2V" وقيمة المقاومة التسلسلية (مقاومة تحديد تيار عمل الثنائي الضوئي) هي "150Ω"، فيمكن حساب قيمة التيار المستجر من كل قطب من العلاقة التالية:

$$I_{LED} = V / R = (5 - 2) / 150 = \mathbf{20mA}$$

كما أن تيار عمل المتحكم من العائلة AVR في النمط الفعال هو 2.4mA، وبالتالي يمكن حساب التيار الكلي من العلاقة:

$$I_{total} = 2.5mA + 8 \times I_{LED} = 8 \times 20mA = \mathbf{162.5mA}$$

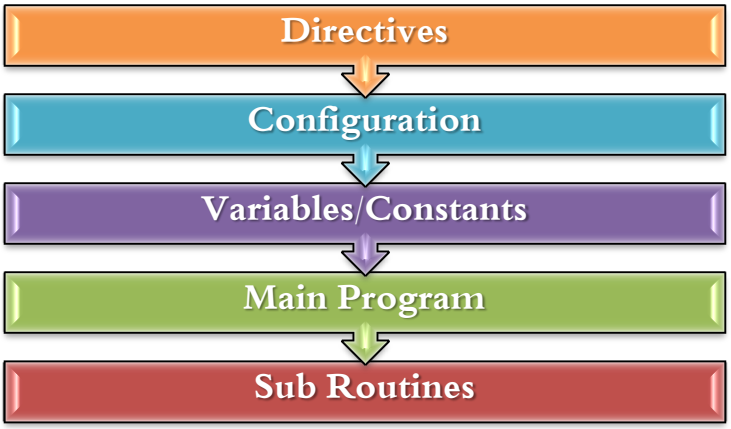
كما هو واضح فإن هذه القيمة تقترب من القيمة العظمة للتيار المسموح استجراره من متحكمات العائلة AVR والذي هو 200mA، بينما تفوق القيمة العظمة للتيار المسموح استجراره من متحكمات العائلة PIC والذي هو 90mA. وبالتالي فإن حساب التيارات المسحوبة من أقطاب المتحكم يعتبر من أهم الأمور التي يجب دراستها في بداية أي مشروع وهو ما سوف نناقشه فيما يأتي.

ملاحظة: عملياً ينصح بأن لا يتجاوز التيار المسحوب من المتحكم نصف قيمة التيار الأعظمي المسموح به لتخفيض ضجيج العمل وللتأكد من أن المتحكم قادر على تيار لعمل الأحمال الموصولة معه بشكل جيد.

## 3-3 تسلسل كتابة برنامج في Bascom-AVR (Writing a Scalable Code in Bascom-AVR):

من أجل كتابة كود برمجي متماسك ومفهوم مع إمكانية تطويره بسهولة مستقبلاً، فإنه يجب الالتزام بالهيكلية التالية في مراحل كتابته:

1) كتابة التوجيهات (Directives) المخصصة للمترجم (Compiler).
2) كتابة الإعدادات وأوامر التهيئة للمحيطيات (Configurations).
3) تعريف المتحولات (Variables) والثوابت (Constants).
4) كتابة حلقة البرنامج الرئيسي الدورية (Main Program).
5) كتابة البرنامج الفرعية (Sub-Routines).

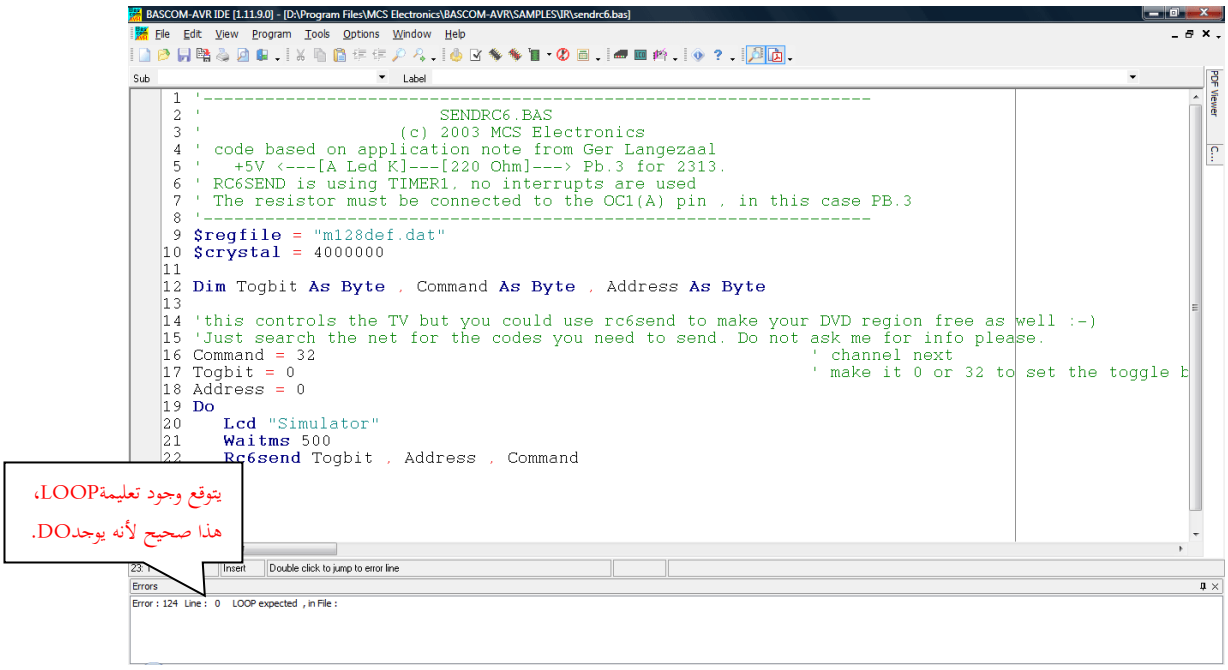



الشكل3-4 تسلسل كتابة برنامج لمتحكم مصغر

## 4-3 خطوات كتابة كود برمجي في البيئة Bascom-AVR (Writing a Code in Bascom-AVR):

1) من أجل كتابة كود برمجي قم باختيار New من القائمة File وابدأ بكتابة البرنامج وفق التسلسل المبين على الشكل3-4.

2) بعد الانتهاء من كتابة الكود البرمجي قم بحفظه في مجلد ثم قم باختيار أمر تفحص الأخطاء "Syntax Check" من القائمة Program. في حال وجود خطأ برمجي سوف تشير نافذة الأخطاء (أسفل الواجهة الرئيسية) إلى موقع الخطأ وسببه.




الشكل3-5 محرر التعليمات ونافذة تتبع الأخطاء في البيئة Bascom-AVR

3) بعد الانتهاء من تفحص الأخطاء، قم باختيار أمر الترجمة "Compile" من القائمة Program ليقوم البرنامج بتوليد الملفات البرمجية اللازمة للمبرمجة والتي سيتم توليدها في نفس المجلد، والملف الذي تحتاجه المبرمجة هو ذو امتداد "*.hex".

4) يمكن تشغيل نافذة المحاكاة الخاصة بالبيئة Bascom-AVR ومحاكاة التطبيق.

5) قم باختيار أمر الإرسال إلى المبرمجة (Send to programmer) من القائمة Program.

## 5-3 التعليمات الأساسية في Bascom-AVR (Essential Instructions in Bascom-AVR):

تعليمات التوجيهات الأساسية:

| وظيفة التعليمة | شكل التعليمة |
|---|---|
| تحديد اسم المعالج المستخدم (ATmega128) | `$regfile = "m128def.dat"` |
| تحديد تردد المزاز الكريستالي الذي يعمل عليه المعالج | `$crystal = 1000000` |
| تحديد معدل بود النقل لنافذة الاتصال التسلسلي | `$baud = 9600` |

تعليمات التأخير الزمني:

| وظيفة التعليمة | شكل التعليمة |
|---|---|
| تأخير زمني (قيمة التأخير Value تعطى بالثانية) | **Wait** value |
| تأخير زمني (قيمة التأخير Value تعطى بالميلي ثانية) | **Waitms** value |
| تأخير زمني (قيمة التأخير Value تعطى بالميكرو ثانية) | **Waitus** value |

تعليمات تعريف الأقطاب (دخل/خرج) ومقاومات الرفع الداخلية:

| وظيفة التعليمة | شكل التعليمة |
|---|---|
| تعريف البوابة C كبوابة خرج | **Config** PORTC = **Output** |
| تعريف القطب رقم 5 من البوابة C كقطب خرج | **Config** PINC.5 = **Output** |
| تعريف البوابة C كبوابة دخل | **Config** PORTC = **Input** |
| تعريف القطب رقم 5 من البوابة C كقطب دخل | **Config** PINC.5 = **Input** |
| تفعيل مقاومات الرفع الداخلية للبوابة C | PORTC = 255 |
| تفعيل مقاومة الرفع الداخلية للقطب رقم 5 من البوابة C | PINC.5 = 1 |
| إلغاء تفعيل مقاومة الرفع الداخلية للقطب رقم 5 من البوابة C | PINC.5 = 0 |
| تفعيل بعض مقاومات الرفع الداخلية للبوابة C | PORTC = &B11110000 |
| يمكن استخدام هذا الشكل لتعريف الأقطاب من البوابة كدخل/خرج حيث أن (0) تعني قطب دخل، والـ (1) تعني قطب خرج. | **Config** PORTC = &B11110000 |
| يصرح إلى أن البوابة (C) سوف يشار إليها أثناء البرنامج بالاسم (Leds) | Leds **Alias** PORTC |
| يصرح إلى أن القطب (5) سوف يشار إليه أثناء البرنامج بالاسم (Led) | Leds **Alias** PORTC.5 |

تعليمات التعامل على مستوى البت (Set/Reset):

| وظيفة التعليمة | شكل التعليمة |
|---|---|
| جعل قيمة (البت/بت من متحول المتحول) واحد منطقي | **Set** bit |
| جعل قيمة (البت/بت من متحول المتحول) صفر منطقي | **Reset** bit |
| تغيير قيمة (البت/بت من متحول المتحول) إلى الحالة المعاكسة | **Toggle** bit |

تعليمات الحلقات:

| وظيفة التعليمة | شكل التعليمة |
|---|---|
| يستمر بالدوران في الحلقة وتنفيذ التعليمات الموجودة في جسم الحلقة حتى تحقق الشرط أو الخروج القسري من الحلقة. | **Do**<br>Statements<br>**Loop** [**until** Expression] |

| | |
|---|---|
| **While** Condition<br>Statements<br>**Wend** | تنفيذ جملة من التعليمات طالما أن الشرط محقق. |
| **For** Var = **Start To End** [**step** Value]<br>Statements<br>**Next** Var | تنفيذ جملة من التعليمات عدداً من المرات يبدأ من القيمة Start وينتهي عند القيمة End. يمكن تحديد خطوة العد بالمتحول step. |
| **Exit For**<br>**Exit Do**<br>**Exit While** | خروج قسري من الحلقة For<br>خروج قسري من الحلقة Do<br>خروج قسري من الحلقة While |

التعليمات الشرطية:

| شكل التعليمة | وظيفة التعليمة |
|---|---|
| **If** Expression1 **Then**<br>Statements1<br>...<br>**Elseif** Expression2 **Then**<br>Statements2<br>...<br>**Else**<br>Statements3<br>...<br>**End If** | اختبار حالة أو قيمة متحول وتنفيذ تعليمات معينة تبعاً لنتيجة شروط الاختبار.<br>إذا تحقق الشرط1 فنفذ التعليمات1<br>وإلا إذا تحقق الشرط2 فنفذ التعليمات2<br>وغير ذلك نفذ التعليمات3 |
| **SELECT CASE** var<br>**Case** Test1 : Statements1<br>**Case** Test2 : Statements2<br>**Case Else** : Statements3<br>**END SELECT** | اختبار حالة أو قيمة متحول وتنفيذ تعليمات معينة تبعاً لنتيجة شرط الاختبار المتحقق.<br>إذا كان var = Test1 فنفذ التعليمات1<br>إذا كان var = Test2 فنفذ التعليمات2<br>وغير ذلك نفذ التعليمات3 |

تعليمات تعريف المتحولات في الذاكرة SRAM:

| شكل التعليمة | وظيفة التعليمة |
|---|---|
| **Dim** Var1 **As Bit** | تعريف متحول عددي نوع بت (0 or 1). |
| **Dim** Var2 **As Byte** | تعريف متحول عددي نوع بايت (0 to 255). |
| **Dim** Var3 **As Integer** | تعريف متحول عددي صحيح (-32,768 to +32,767). |
| **Dim** Var4 **As Word** | تعريف متحول عددي نوع وورد (0 to 65535). |
| **Dim** Var5 **As Long** | تعريف متحول عددي طويل (-2147483648 to 2147483647). |
| **Dim** Var6 **As Single** | تعريف متحول عددي مؤشر (1.5 x 10^-45 to 3.4 x 10^38). |
| **Dim** Var7 **As Double** | تعريف متحول عددي مؤشر مضاعف. |
| **Dim** Var8 **As String** * 1 | تعريف متحول نوع محرفي محدد المحارف بـ (chr_num *). |

| | |
|---|---|
| تعريف مصفوفة بثمان بايتات. | `Dim Array(8) As Byte` |
| تعريف متحول رقمي ثابت. | `Const Symbol = Numconst`<br>`Ex. Const Pi = 3.14159265358979` |
| تعريف متحول محرفي ثابت. | `Const Symbol = Stringconst`<br>`Ex. Const S = "TEST"` |
| تعريف تعبير رياضي ثابت. | `Const Symbol = Expression`<br>`Ex. Const E =(b1 * 3) + 2` |
| تعريف متحول محلي في برنامج فرعي أو برنامج فرعي وظيفي. | `Local Var As Type` |

تعليمات قراءة حالة مفاتيح موصولة مع أقطاب دخل:

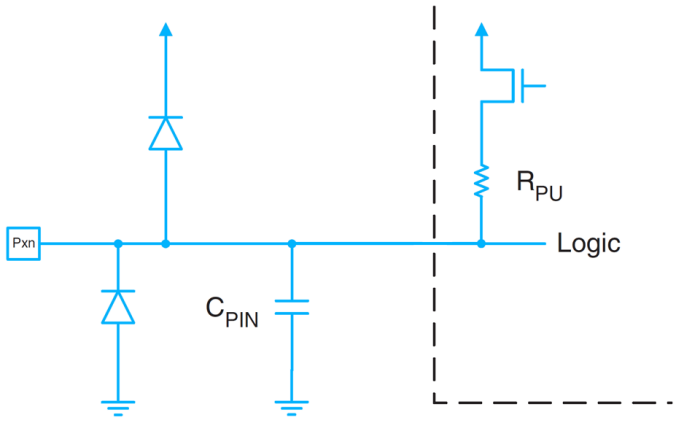
| وظيفة التعليمة | شكل التعليمة |
|---|---|
| يراقب حالة القطب المحدد في Px.y كلما مر عليه وعندما تصبح حالته موافقة للحالة المحددة في state، سوف يقفز إلى البرنامج الفرعي عند اللافتة label وينفذ البرنامج ويعود. | `Debounce` $P_{x.y}$ `, state , label , Sub`<br>`Ex. Debounce Key1 , 0 , Sw1 , Sub` |
| تهيئة زمن تأخير (ميلي ثانية) عن استعمال تعليمة Debounce للتخلص من العطالة الميكانيكية للمفتاح. | `Config Debounce = time` |
| سوف يقف البرنامج عند هذه التعليمة وينتظر أن تصبح حالة البت (القطب) صفر أو واحد منطقي عندها يكمل البرنامج. | `Bitwait x , Set/reset`<br>`Ex. Bitwait Pinb.7 , reset` |

يمكن الاطلاع على مبادى وأساسية البرمجة في البيئة Bascom من خلال ملف المساعدة (BASCOM-AVR IDE Help) وضمن "Language Fundamentals".

## 6-3 بوابات الدخل والخرج في متحكمات AVR (GPIOs in AVR MCUs):

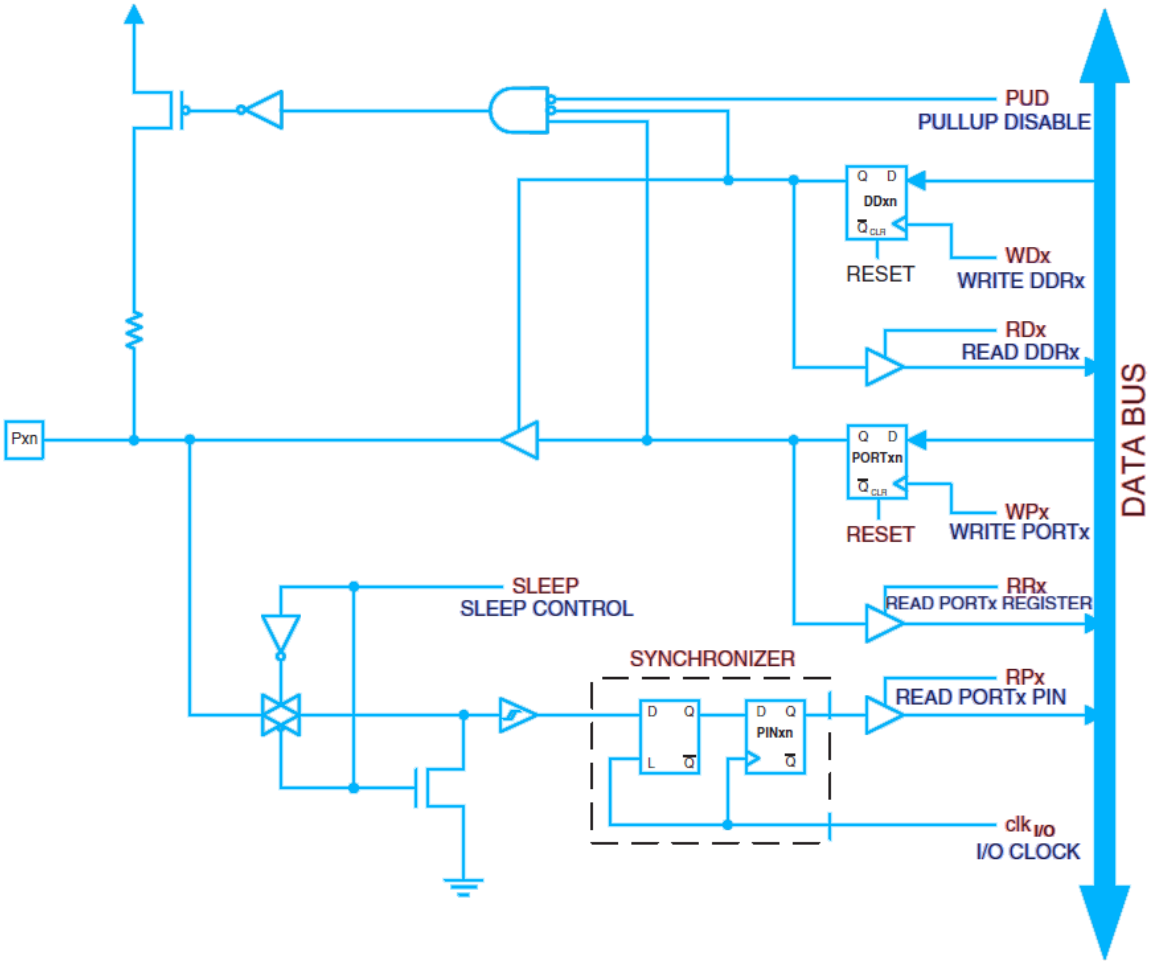
تتمتع جميع أقطاب بوابات متحكمات العائلة AVR بأنها أقطاب ثنائية الاتجاه وظائف قراءة وكتابة وتعديل عند استخدامها كأقطاب دخل/خرج للأغراض العامة (GPIOs)، كما يمكن تغيير اتجاه أحد أقطاب بوابة بشكل منفصل – خلافاً لمتحكمات 8051 – فيمكن تعريف كل قطب من الأقطاب على حدى كقطب دخل أو خرج. كذلك تمتلك الأقطاب عند تعريفها كأقطاب دخل مقاومات رفع داخلية – إلى التغذية – يمكن تفعيلها أو إلغاء تفعيلها لكل قطب بشكل منفصل.

إن بنية الأقطاب هي من النوع "Push-pull" أي أنها قادرة على قيادة الخرج على المستوى المنطقي "0" والمستوى "1" حيث أن التيار الذي يمكن أن يزوده القطب قادر على قيادة ثنائي ضوئي (LED) بشكل مباشر دون الحاجة إلى دارة مفتاح ترانزستوري. كما أن جميع الأقطاب مزودة بدارة حماية من تفريغ الشحنات الستاتيكية (ESD) مؤلفة من ثنائيين شوتكي أحدها موصل إلى التغذية (للحماية من شحنات التفريغ الموجبة) والآخر موصل إلى النقطة الأرضية (للحماية من شحنات التفريغ السالبة) كما هو مبين في الشكل3-6.




الشكل3-6 دارة الحماية من شحنات التفريغ الستاتيكية لقطب متحكم AVR

يتم تصنيف أقطاب الدخل/الخرج العامة (GPIOs) في مجموعات تسمى بوابات (PORTs) كل بوابة تتألف من ثمانية أقطاب (PINs)، ويختلف عدد البوابات باختلاف عدد أقطاب المتحكم حيث يمكن أن يصل عدد البوابات في متحكمات AVR المتقدمة إلى أحد عشر بوابة ويشار إليها بالأحرف **PORT**A, B, C, D, E, F, G, H, J, K, L.




الشكل3-7 البنية الداخلية الكاملة والدارات المنطقية لقطب متحكم AVR

## 3-7 مسجلات بوابات الدخل والخرج في متحكمات AVR (AVR MCUs GPIO Registers):

تمتلك كل بوابة من بوابات المتحكم ثلاث مسجلات تحكم، حيث يمثل الرمز **x** رمز البوابة (A, B, C, D, E, F, G, H, J, K, L) والرمز y قطب البوابة (7 : 0).

1) **مسجل التحكم باتجاه المعطيات للبوابة DDRx** (Data Direction Register): يتم من خلاله تحديد وظيفة كل قطب – دخل أو خرج. تمثل كل خانة من خانات مسجل اتجاه المعطيات الثمانية قطباً من أقطاب البوابة الموافقة (المسجلات أدناه هي للبوابة A)، حيث أنه عند وضع القيمة "**1**" في خانة المسجل DDRx.y فإن القطب الموافق لهذه الخانة يصبح قطب خرج، أما عند وضع القيمة "**0**" في خانة المسجل DDRx.y فإن القطب الموافق لهذه الخانة يصبح قطب دخل.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **DDRA** | **DDA7** | **DDA6** | **DDA5** | **DDA4** | **DDA3** | **DDA2** | **DDA1** | **DDA0** |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

الشكل3-8 مسجل اتجاه المعطيات للبوابة PORTA

2) **مسجل الخرج للمعطيات PORTx** (Data Output Register): يوجد له وظيفتان:

- في حال كان القطب معرفاً في مسجل DDRx كقطب خرج ("1" = DDRx.y)، فإن مسجل خرج المعطيات سيحدد الحالة المنطقية المطبقة على القطب بحيث إما أن يكون منبع للتيار ("1") أو مصرف للتيار ("0").
- في حال كان القطب معرفاً في مسجل DDRx كقطب دخل ("0" = DDRx.y)، فإن مسجل خرج المعطيات سيتحكم بوصل ("1") أو فصل ("0") مقاومة الرفع الداخلية للقطب المعني.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **PORTA** | **PORTA7** | **PORTA6** | **PORTA5** | **PORTA4** | **PORTA3** | **PORTA2** | **PORTA1** | **PORTA0** |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

الشكل3-9 مسجل خرج المعطيات للبوابة PORTA

3) **مسجل الدخل للمعطيات PINx** (Data Input Register): يستخدم لقراءة الحالة الخارجية المطبقة على القطب المعني ("0" | "1") عند تعريف القطب في مسجل اتجاه المعطيات كقطب دخل ("0" = DDRx.y).

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **PINA** | **PINA7** | **PINA6** | **PINA5** | **PINA4** | **PINA3** | **PINA2** | **PINA1** | **PINA0** |
| Read/Write | R | R | R | R | R | R | R | R |
| Initial Value | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |

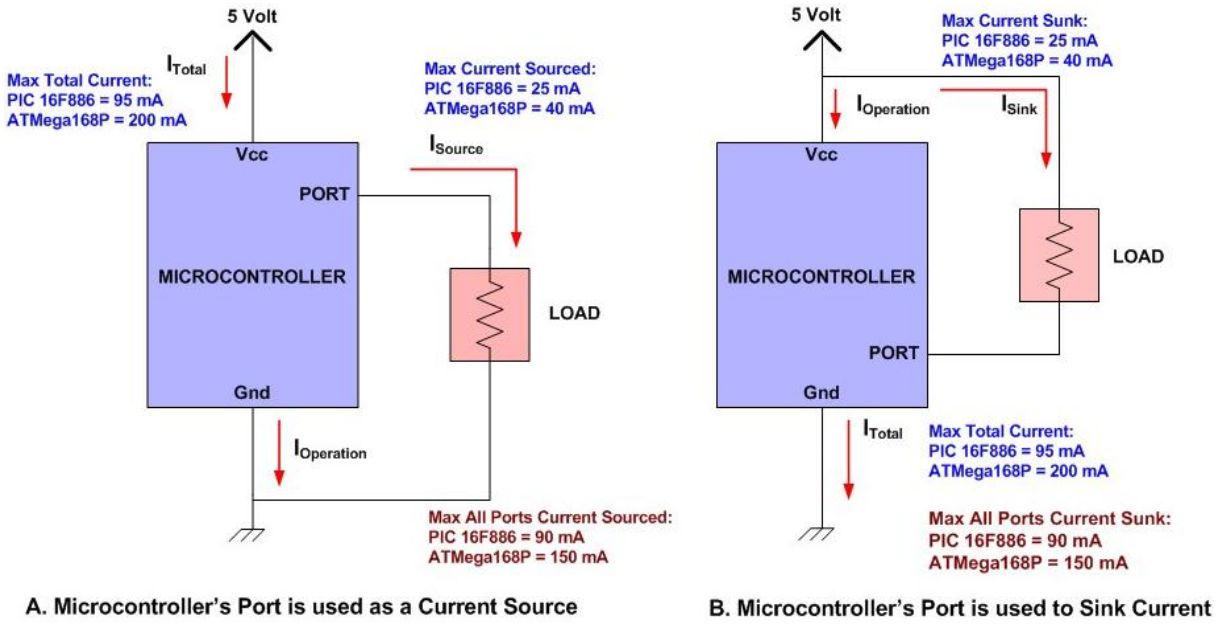
الشكل3-10 مسجل دخل المعطيات للبوابة PORTA

## 3-8 طرق توصيل الأحمال مع أقطاب المتحكم (Interfacing Loads with MCU):

إن بنية الأقطاب في متحكمات AVR ومتحكمات PIC هي من النوع "Push-pull"، أي أنها قادرة على قيادة الخرج على المستوى المنطقي "0" والمستوى "1" حيث أن التيار الذي يمكن أن يزوده القطب قادر على قيادة ثنائي ضوئي (LED) بشكل مباشر دون الحاجة إلى دارة مفتاح ترانزستوري. وبالتالي يمكن وصل الأحمال مع أقطاب المتحكم بطريقتين:

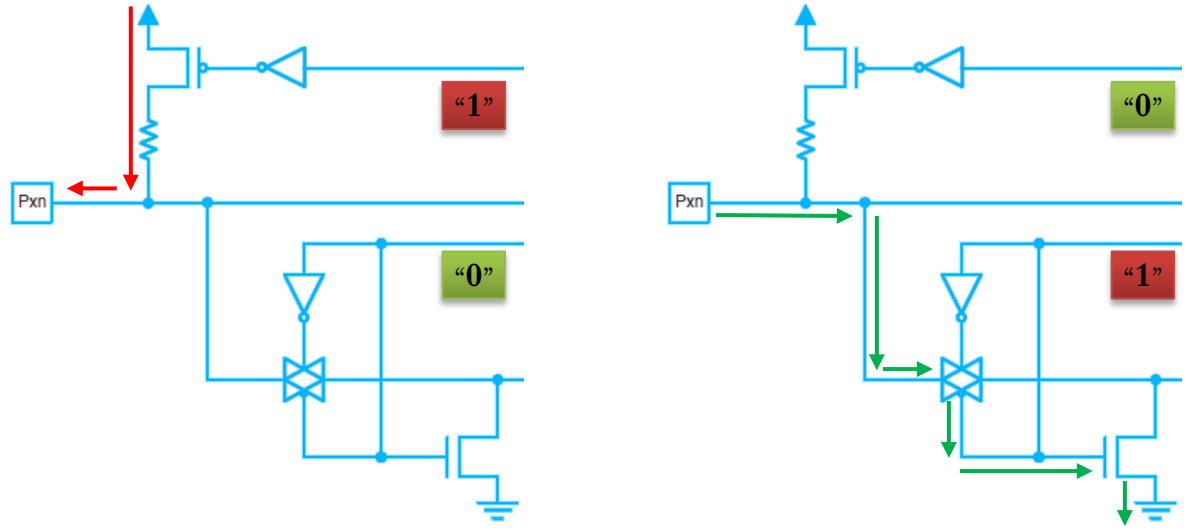
- القطب يعمل كمنبع لتيار تشغيل الحمل (Source) – الشكل3-11 (A).
- القطب يعمل كمصرف لتيار تشغيل الحمل (Sink) – الشكل3-11 (B).

في الحالة الأولى – القطب يعمل كمنبع (Source) لتيار تشغيل الحمل – يتم تزويد التغذية للحمل عن طريق التغذية الداخلية للمتحكم والتصريف يكون من خلال النقطة الأرضية مباشرة؛ أما في الحالة الثانية – القطب يعمل كمصرف (Sink) لتيار تشغيل الحمل – فإنه يتم تزويد التغذية للحمل مباشرة من التغذية الرئيسية ويتم التصريف من خلال المتحكم والنقطة الأرضية له. في كلا الحالتين سيكون الأداء للمتحكم واحداً إلا أنه يوصى عادة بالطريقة الثانية وذلك لتخفيض ضجيج التغذية VCC داخل المتحكم، كما أن توزع مسارات النقطة الأرضية GND داخل المتحكم أكبر وبالتالي التصريف سيكون موثوقاً ومناعته للضجيج أكبر.




الشكل3-11 طرق توصيل الأحمال مع أقطاب المتحكم – كمنبع أو مصرف للتيار

إن مبدأ سير التيار في الحالة الأولى (المسار باللون الأحمر على اليسار) والثانية (المسار باللون الأخضر على اليمين) داخل البنية الداخلية لأقطاب المتحكم مبين على الشكل3-12.

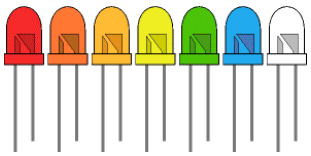


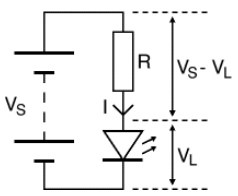

الشكل3-12 سير التيار داخل البنية الداخلية لقطب متحكم AVR كمصرف ومنبع للتيار

## 3-9 حساب قيمة واستطاعة مقاومة تحديد التيار (Calculating Current Resistor Value):

إن قيمة مقاومة تحديد التيار للحمل تتعلق مباشرة بجهد تشغيل الحمل وتياره ومقاومته الأمامية. من أجل حساب قيمة مقاومة تحديد التيار لثنائي ضوئي (LED) على سبيل المثال فإنه يجب معرفة تيار وجهد التشغيل للثنائي. إن تيار وجهد العمل للثنائيات الضوئية يختلف حسب لون الثنائي الضوئي، الجدول التالي يوضح المواصفات الكهربائية للثنائيات الضوئية.

| Type | Colour | $I_F$ max. | $V_F$ typ. | $V_F$ max. | $V_R$ max. | Luminous intensity | Viewing angle | Wavelength |
|---|---|---|---|---|---|---|---|---|
| Standard | Red | 20mA | 2.0V | 2.3V | 5V | 5mcd @ 10mA | 60° | 660nm |
| Super bright | Bright red | 25mA | 3.0V | 3.4V | 5V | 80mcd @ 10mA | 60° | 625nm |
| Standard | Yellow | 20mA | 2.1V | 2.3V | 5V | 32mcd @ 10mA | 60° | 590nm |
| Standard | Green | 20mA | 3.2V | 3.5V | 5V | 32mcd @ 10mA | 60° | 565nm |
| High intensity | Blue | 20mA | 3.4V | 3.6V | 5V | 60mcd @ 20mA | 50° | 430nm |
| Super bright | White | 20mA | 3.4V | 3.6V | 5V | 500mcd @ 20mA | 60° | 660nm |






IF max: التيار الأعظمي الأمامي المار في الثنائي.

VF typ: الجهد الأمامي النموذجي من اجل تشغيل الثنائي.

VF max: الجهد الأمامي الأعظمي الذي يمكن للثنائي أن يتحمله.

VR max: الجهد العكسي الأعظمي الذي يمكن للثنائي أن يتحمله.

Luminous intensity: شدة السطوع للثنائي.

Viewing angle: زاوية انعكاس الرؤية للإضاءة.

Wavelength: طول موجة الضوء الصادر.

وبالتالي من أجل ثنائي ضوئي ذو لون أحمر فإن جهد وتيار العمل هو 2V/20mA،

وبالتالي يمكن حساب مقاومة تحديد التيار من العلاقة:

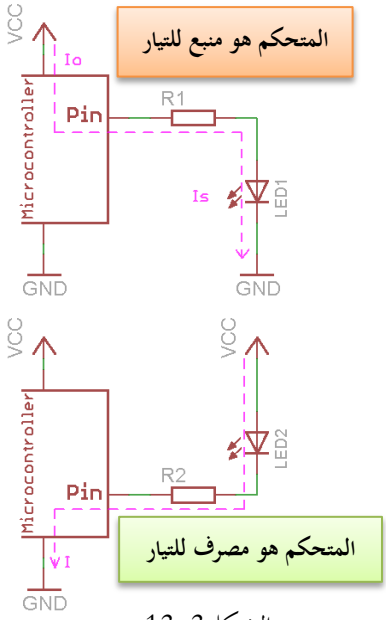
$$R_{LED} = \frac{V_{cc} - V_{LED}}{I_{LED}}$$

$$R_{LED} = \frac{5-2}{20} = \frac{3}{20} = 150\Omega$$

$$PR_{LED} = V_R \times I_R = (V_{cc} - V_{LED}) \times I_{LED}$$

$$PR_{LED} = (5-2) \times 20 = 60mW$$

وبالتالي فإن الذي نحتاجه هو مقاومة 150Ω ذات استطاعة $1/4_{Watt}$.

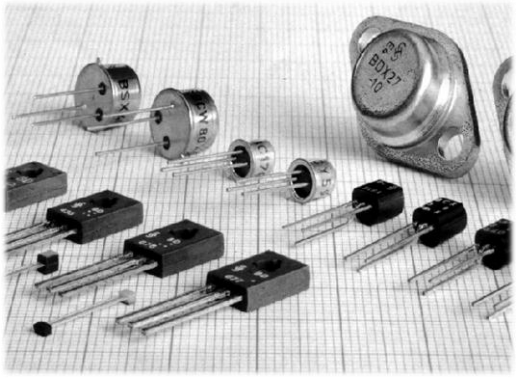



الشكل3-13

## 3-10 مفاتيح التحكم الترانزستورية (Transistors as Control Switches):

من أجل التحكم بأحمال ذات تيارات كبيرة (محركات، ريله، سخانات) فإن تيار الخرج لقطب المتحكم (20mA) لا يمكنه قيادة هذه الأحمال، لذا يتم استخدام الترانزستورات كمفاتيح إلكترونية (On/Off) للتحكم بهذه الأحمال. بشكل عام يوجد نوعين من الترانزستورات:

- الترانزستورات ثنائية القطبية (BJT).
- الترانزستورات أحادية القطبية (FET).



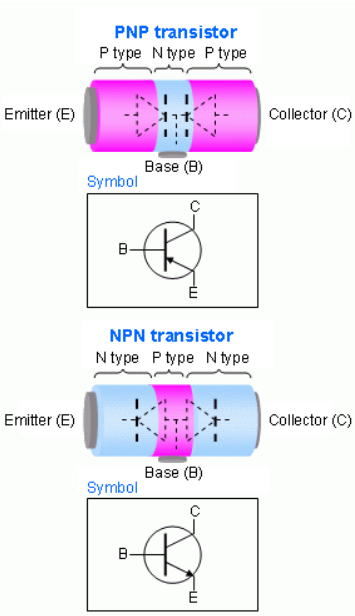
عملياً، إن الاستخدام لكل منها يختلف بحسب طبيعة الحمل المقاد، الجدول التالي يبين الفرق بين كلا النوعين:

| | BJT | FET/MOSFET |
|---|---|---|
| طريقة التحكم | يتم التحكم به عن طريق تيار القاعدة ويحتاج تيار 1 – 10mA بالإضافة إلى VBE=0.6V | يتم التحكم به عن طريق جهد البوابة ويختلف الجهد حسب استطاعة الترانزستور. |
| سرعة الفتح والإغلاق | أبطئ لا يتجاوز 200MHZ (uS) | 10 مرات أسرع (nS) |
| العمل | تأثر كبير بالحرارة | أقل تأثراً بالحرارة |
| المقاومة الأمامية | المقاومة الأمامية (هبوط جهد أمامي) صغيرة جداً | مقاومة أمامية كبيرة نسبياً |
| التأثر | لا يتأثر بالشحنات الساكنة | يمكن أن يتأثر ويدمر بالشحنات الساكنة |
| ممانعة الدخل | متوسطة | كبيرة جداً ($10^{12}$ Ω) |
| مجالات جهود العمل | صغيرة لا تتجاوز 100V | كبيرة جداً |
| تيار الحمل | يعمل من أجل تيارات أحمال صغيرة | يمكنه أن يقود أحمال بتيارات عالية (محرك) |
| ضجيج العمل | ضجيج عالي | ضجيج منخفض |

يتم استخدام الترانزستورات ثنائية القطبية من أجل التحكم بأحمال ذات تيارات صغيرة. بينما تستخدم الترانزستورات الحقلية من أجل التحكم بأحمال ذات تيارات وجهود متوسطة وكبيرة.

إن مجال استخدام الترانزستورات في أنظمة التحكم الرقمي يقتصر على استخدام هذه الترانزستورات كمفاتيح إلكترونية تحكمية (On/Off) – يعمل في منطقتي القطع والإشباع، وبالتالي فإن اختيار الترانزستور نسبة إلى الحمل سيعتمد على ثلاث عوامل أساسية:

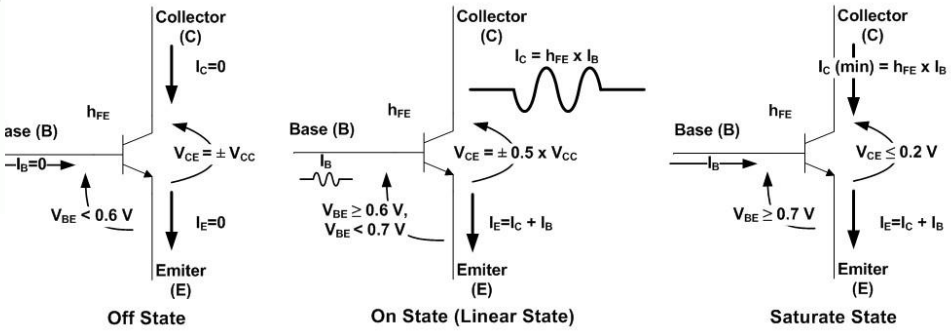
- التيار المار في الترانزستور.
- الاستطاعة المبددة في الترانزستور.
- سرعة الفتح والإغلاق للترانزستور.




في حالة القطع (Off state): يكون تيار القاعدة IB=0.

في الحالة الفعالة (On active state): يكون فيها تيار المجمع $\mathbf{I_C = I_B \times h_{FE}}$ وهي الحالة التي يستخدم فيها الترانزستور كمضخم فعال – أي زيادة في تيار القاعدة ينتج عنه زيادة في تيار المجمع.

في حالة الإشباع (On saturate state): في هذه الحالة يمرر الترانزستور كامل التيار.

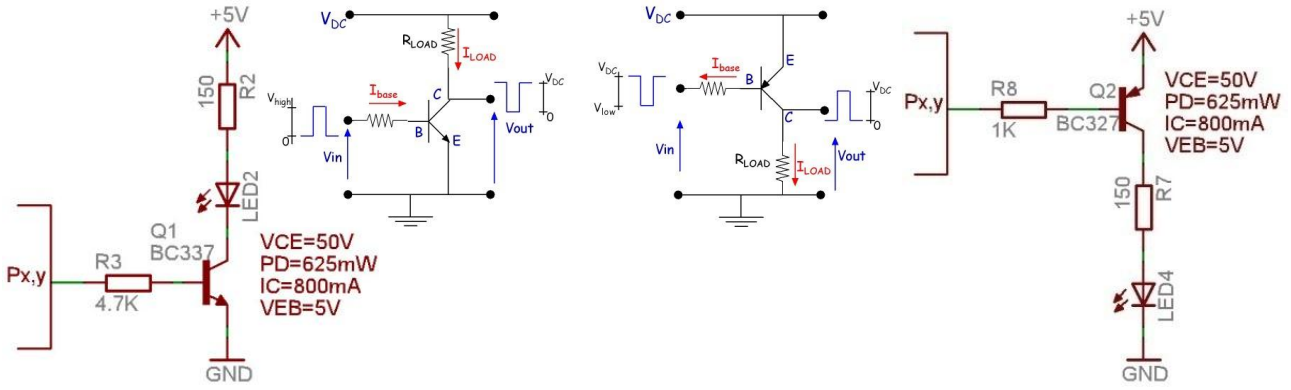



الشكل3-14 عمل الترانزستور وما يقابل كل حالة من شروط للجهد والتيار

## 3-11 استخدام مفاتيح التحكم الترانزستورية ثنائية القطبية (Using BJT Transistors as Control Switches):

يمكن توصيل المفاتيح الترانزستورية بطريقتين:

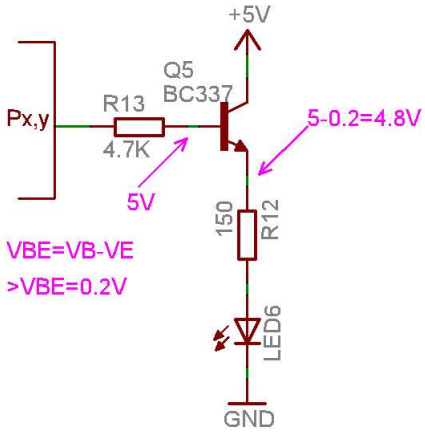
1) متحكم بها لتكون فعالة عند المنطق العالي "**1**": وبالتالي فإن الترانزستور سوف يعمل كمفتاح لوصل/فصل النقطة الأرضية (GND) للحمل، وفي هذه الحالة سوف يستخدم ترانزستور من نوع NPN – الشكل3-15.
2) متحكم بها لتكون فعالة عند المنطق المنخفض "**0**": وبالتالي فإن الترانزستور سوف يعمل كمفتاح لوصل/فصل نقطة التغذية (VCC) للحمل، وفي هذه الحالة سوف يستخدم ترانزستور من نوع PNP – الشكل3-16.



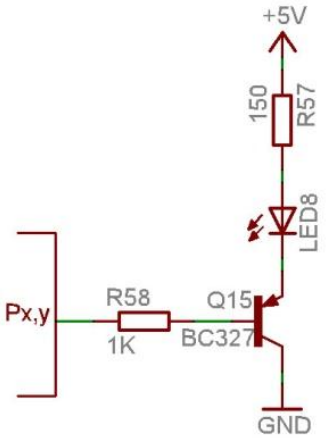

الشكل3-15 مفتاح ترانزستوري فعال عند المنطق "0"

الشكل3-16 مفتاح ترانزستوري فعال عند المنطق "0"

في بعض الأحيان يحصل خطأ في تصميم دارة المفتاح الإلكتروني باستخدام الترانزستور ثنائي القطبية، وهو من خلال استخدام الترانزستورات من نوع NPN كمفتاح لوصل/فصل نقطة التغذية (VCC) للحمل – الشكل3-17، أو استخدام الترانزستور من نوع PNP كمفتاح لوصل/فصل النقطة الأرضية (GND) للحمل – الشكل3-18.



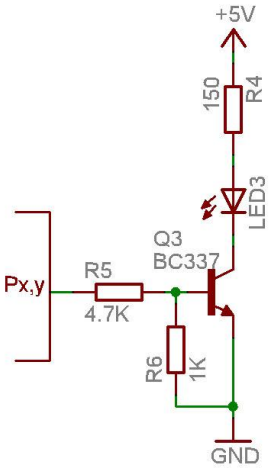

الشكل3-17 توصيل خاطئ لمفتاح ترانزستوري نوع NPN



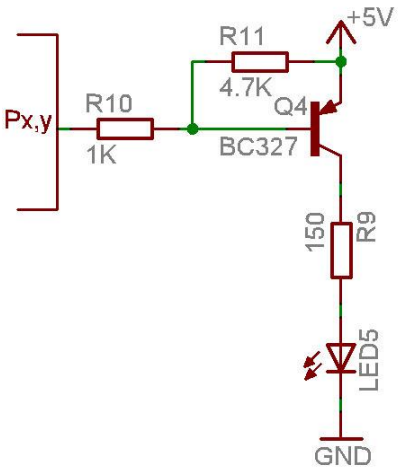

الشكل3-18 توصيل خاطئ لمفتاح ترانزستوري نوع PNP

لنوضح الخطأ من خلال الحسابات التالية: حتى يفتح الترانزستور بشكل كامل (الإشباع)، فيجب أن يكون الجهد $V_{BE}=0.7V$. بالنظر إلى الدارة على الشكل6-10 نجد أن الجهد الموجود على المشع (E) هو: $V_E= V_{CC}-V_{CE} = 5 - 0.2 = 4.8V$. كما أن الجهد على قاعدة الترانزستور هو: $V_B = V_{PIN} = 5V$. وبالتالي فإن: $V_{BE}= V_B - V_E = 5 - 4.8 = 0.2V$!! هذا يعني أن الترانزستور يعمل في المنطقة الفعالة ولن يكفي تيار مجمع الترانزستور ($I_C$) لتشغيل الحمل وسيعمل الثنائي الضوئي بشكل خافت.

إن الجهد المطبق على قاعدة الترانزستور في الدارة المبينة في الشكل3-15 والشكل3-16 يساوي 5V وهو نفسة جهد منطق بوابة المتحكم المصغر، بنفس الوقت من أجل الفتح الكامل للترانزستور فإنه يكفي تطبيق 0.7V، وإن هذا الجهد الزائد على القاعدة يؤدي إلى سحب تيار زائد وضياع في الاستطاعة، وبالتالي يمكن إضافة مقاومة مع مقاومة القاعدة ليتشكل لدينا مقسم كمون خرجه يتراوح بين 0.7~1V.
الشكل3-19 والشكل3-20 يوضحان طريقة إضافة المقاومة وحساب الجهد VBE.




الشكل3-20 دارة مفتاح ترانزستوري فعال عند المنطق "1"

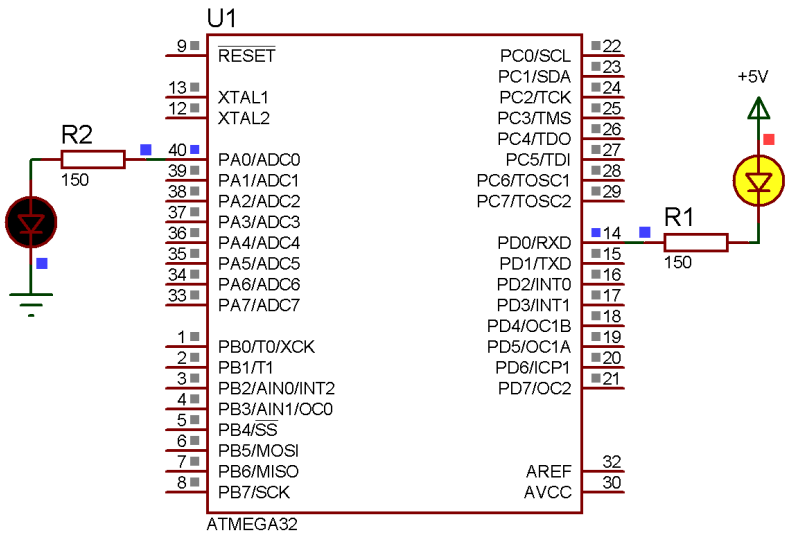



الشكل3-19 دارة مفتاح ترانزستوري فعال عند المنطق "0"

$$VBE = Vcc \times \frac{R6}{R6 + R5} = 5 \times \frac{1}{4.7 + 1} = 0.87V$$

## 12-3 برمجة بوابات الدخل والخرج في متحكمات AVR (Programming AVR MCUs GPIOs):

التجربة الأولى: تم وصل ثنائيين ضوئيين إلى متحكم ATmega32A، الثنائي الأول (LED1) موصول إلى القطب PINA.0 بحيث أن قطب المتحكم هو منبـع للتيار (فعال عند المستوى المنطقي "1") ، الثنائي الثاني (LED2) موصول إلى القطب PIND.0 بحيث أن قطب المتحكم هو مصرف للتيار (فعال عند المستوى المنطقي "0")، والمطلوب: كتابة برنامج خفقان بالتناوب لكلا الثنائيين كل 0.5S.

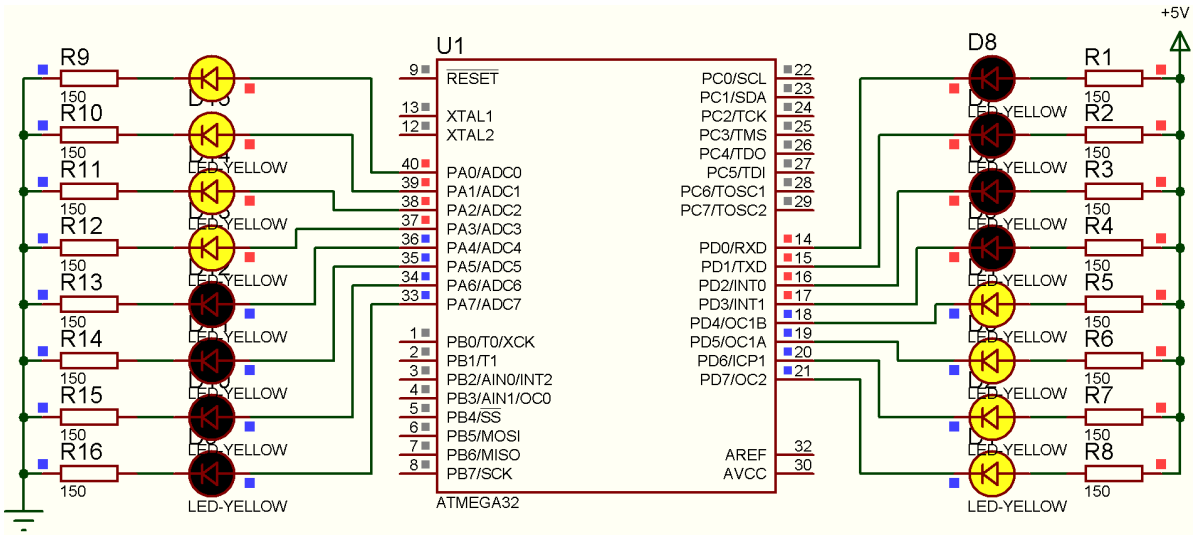



الشكل3-21 توصيل الثنائيات مع المتحكم للتجربة 1

البرنامج Exp.01.bas في بيئة BASCOM-AVR:

```
' ********************************************************************************
' * Title          : Exp.01.bas                                                  *
' * Target MCU     : ATMega32A                                                   *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : LEDs                                                        *
' * Description    : GPIOs as Outputs                                            *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'------------------------
'-----------------------[GPIO Configurations]
Config Pina.0 = Output : Led1 Alias Porta.0
Config Pind.0 = Output : Led2 Alias Portd.0
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
    '>[Turn Led1 on & Led1 off]
   Set Led1 : Set Led2 : Waitms 500
    '>[Turn Led1 off & Led2 on]
   Reset Led1 : Reset Led2 : Waitms 500
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

التجربة الثانية: تم وصل ثمانية ثنائيات ضوئية (LEDs_A) إلى البوابة PORTA للمتحكم ATmega32A بحيث أن بوابة المتحكم هي في حالة منبـع للتيار، وتم وصل ثمانية ثنائيات ضوئية أخرى (LEDs_D) إلى البوابة PORTD بحيث أن بوابة المتحكم هي في حالة مصرف للتيار، والمطلوب: كتابة برنامج بحيث تعمل الثنائيات بالتتابع ذهاباً وإياباً باستخدام الحلقات.

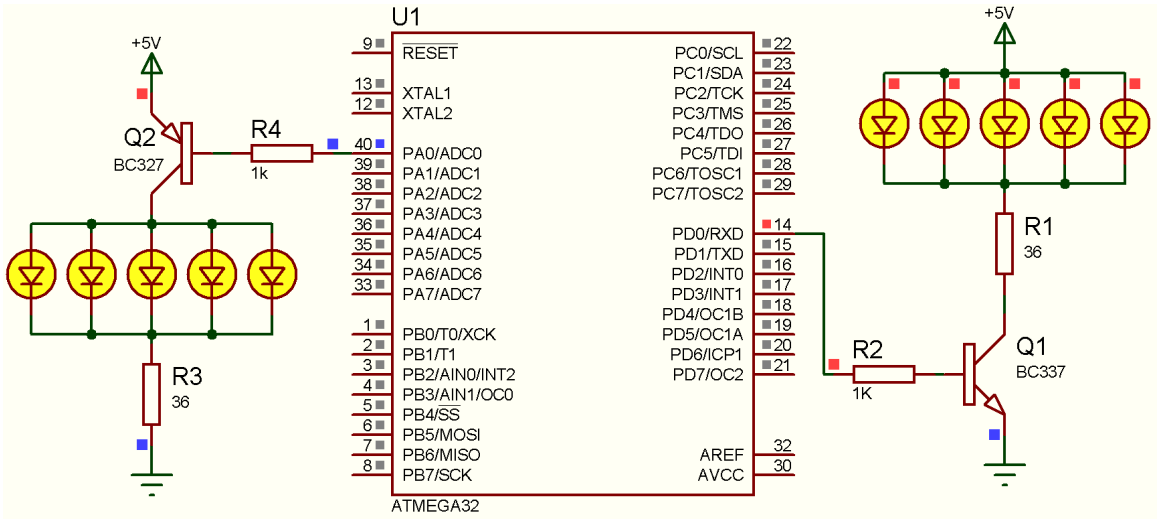



الشكل3-22 توصيل الثنائيات مع المتحكم للتجربة 2

البرنامج Exp.02.bas في بيئة BASCOM-AVR:

```
' ************************************************************************************
' * Title          : Exp.02.bas                                                      *
' * Target MCU     : ATMega32A                                                       *
' * Author         : Walid Balid                                                     *
' * IDE            : BASCOM AVR 2.0.7.3                                              *
' * Peripherals    : LEDs                                                            *
' * Description    : GPIOs as Outputs                                                *
' ************************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'-----------------------
'-----------------------[GPIO Configurations]
Config Porta = Output : Leds_a Alias Porta : Leds_a = 0      'Set All LEDs off
Config Portd = Output : Leds_d Alias Portd : Leds_d = 0      'Set All LEDs on
'-----------------------
'-----------------------[Variables]
Dim I As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   Gosub Leds_fw : Gosub Leds_rw
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[To Turn-on LEDs_A  & Turn-off LEDs_D]
Leds_fw:
   For I = 0 To 7
      Set Leds_a.i : Set Leds_d.i
      Waitms 100
   Next I
Return
'-----------------------
'--->[To Turn-off LEDs_A  & Turn-on LEDs_D]
Leds_rw:
   For I = 7 To 0 Step -1
      Reset Leds_a.i : Reset Leds_d.i
      Waitms 100
   Next I
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

التجربة الثالثة: تم وصل مجموعة من خمسة ثنائيات ضوئية (LEDs_A) إلى متحكم ATmega32A مع القطب PINA.0 عن طريق مفتاح ترانزستوري وبحيث أن الترانزستور هو من النوع PNP (فعال عند المستوى المنطقي "0")، كما تم وصل خمسة ثنائيات ضوئية أخرى (LEDs_B) مع القطب PIND.0 عن طريق مفتاح ترانزستوري وبحيث أن الترانزستور هو من النوع NPN (فعال عند المستوى المنطقي "1")، والمطلوب: كتابة برنامج خفقان بالتناوب لكلا المجموعتين كل 0.5S.




الشكل3-23 توصيل الثنائيات مع المتحكم للتجربة 3

البرنامج Exp.03.bas في بيئة BASCOM-AVR:

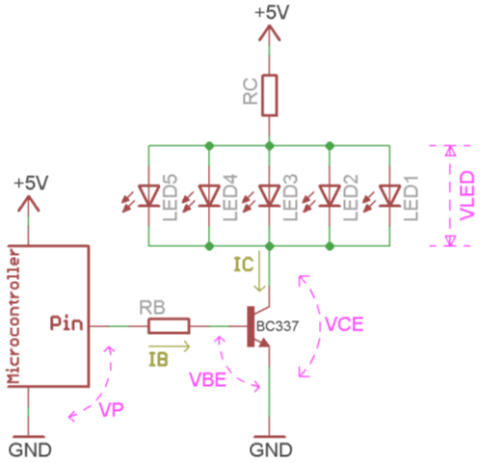
```
' ********************************************************************************
' * Title          : Exp.03.bas                                                  *
' * Target MCU     : ATMega32A                                                   *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : LEDs                                                        *
' * Description    : GPIOs as Outputs                                            *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'-----------------------
'-----------------------[GPIO Configurations]
Config Pina.0 = Output : Leds_a Alias Porta.0
Config Pind.0 = Output : Leds_d Alias Portd.0
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   '>[Turn-on LEDs_A & Turn-off LEDs_D]
   Set Leds_a : Set Leds_d : Waitms 500
   '>[Turn-on LEDs_A & Turn-off LEDs_D]
   Reset Leds_a : Reset Leds_d : Waitms 500
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

## 3-13 تصميم دارة مفتاح ترانزستوري ثنائي القطبية (BJT Control Switch Circuit Design):

لنأخذ مثالاً عملياً ونحسب قيم المقاومات والتيارات للدارة المبينة في الشكل3-24 والتي تستخدم في التجربة السابقة (Exp.03).

لتصميم دراة المفتاح قمنا باختيار الترانزستور BC337 والذي له المواصفات التالية:

$I_{C\_max}$=800mA, $V_{BE\_saturate}$=0.65V, $V_{CE\_saturate}$=0.2V, $h_{FE}$ = 100, $V_{CE\_max}$=50V




الشكل3-24 التحكم بمجموعة ثنائيات من قطب متحكم باستخدام مفتاح ترانزستوري BJT

1) نحسب تيار الحمل (IC) مع العلم أن تيار كل ثنائي ضوئي هـو: $I_{LED}$= 20mA.

$I_C$ = 5 x 20mA = 100mA

2) نحسب مقاومة تحديد التيار مع العلم أن جهد عمل الثنائي هو: $V_{LED}$=2V.

$$Rc = \frac{V_{cc} - V_{LED}}{I_C} = \frac{5-2}{100} = 30\Omega$$

3) نحسب استطاعة مقاومة تحديد التيار.

$$P_{RC} = (V_{cc} - V_{LED}) \times I_C = (5-2) \times 100 = 300mW$$

4) نحسب قيمة التيار الأصغري اللازم لقيادة الترانزستور عن طريق بوابة المتحكم:

$$I_C = hfe \times I_B \rightarrow I_B = \frac{I_C}{hfe} = \frac{100}{100} = 1mA$$

5) نحسب الاستطاعة المبددة في الترانزستور:

$$Pc_{max} = U_{CE} \times I_C = 0.2 \times 100 = 20mW$$

6) نحسب قيمة مقاومة القاعدة واستطاعتها:

$$R_B = \frac{V_P - V_{BE}}{I_B} = \frac{5-0.7}{1} = 4.3K\Omega$$

$$P_{RC} = (V_P - V_{BE}) \times I_B = (5-0.7) \times 1 = 4.3mW$$

. . . ﴿انتهت الجلسة العملية الثانية﴾ . . .

وليد بليد

- دمتم بخير ومودة ونور -

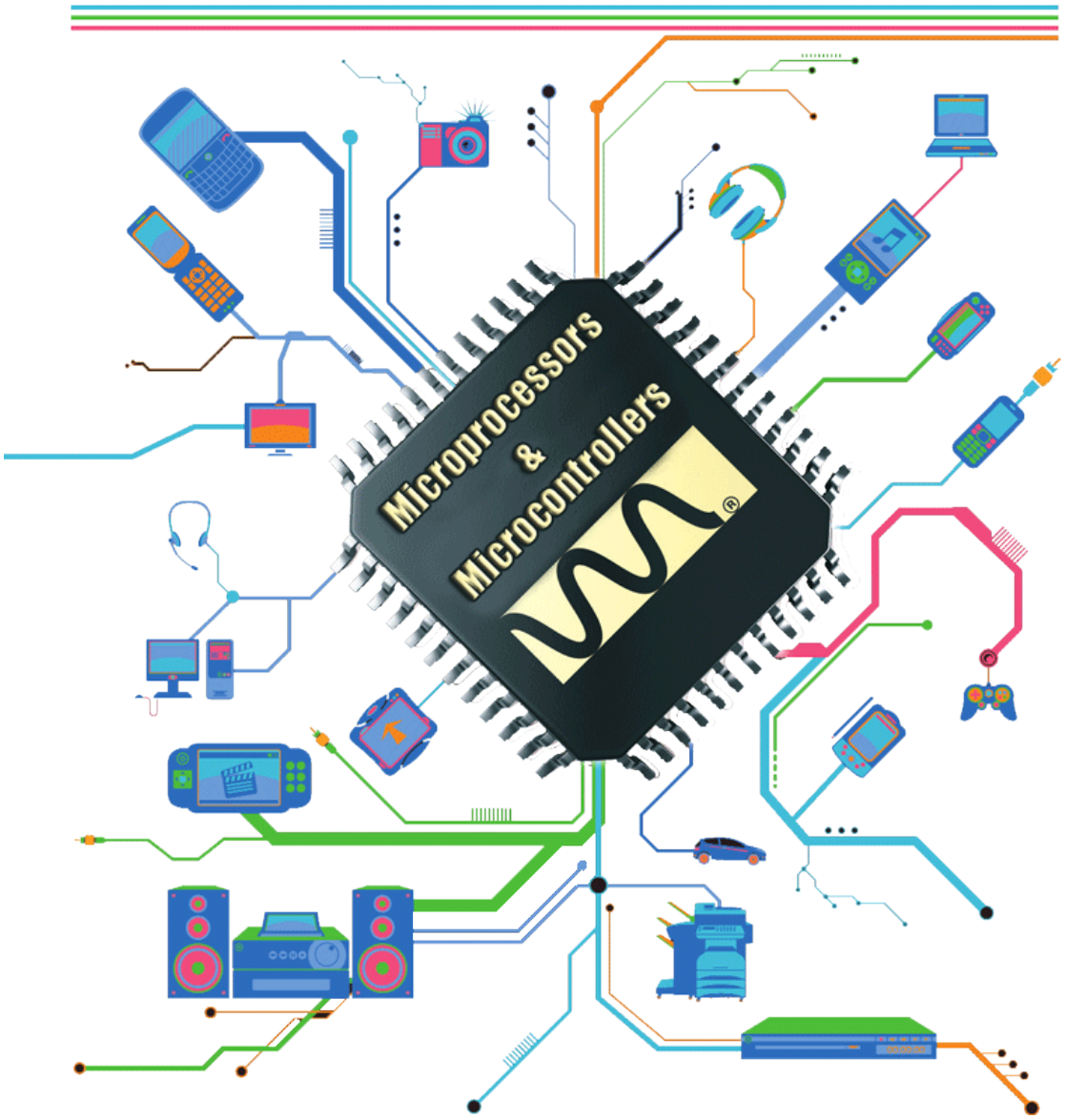
جامعـــــــــــة حلـــــــــــب
كلية الهندسة الكهربائية والإلكترونية
قسـم هندسـة التحكـم والأتمتـة

# ﴿الجلسات العملية لمادة المعالجات والمتحكمات المصغرة﴾

# Microprocessors & Microcontrollers Lab Sessions

السنة الثالثة | قسم اتصالات

## ﴿الجلسة العملية الرابعة﴾




م. وليد بليد

Copyright © 2012 Walid Balid - All rights reserved.

Wednesday, March 28, 2012

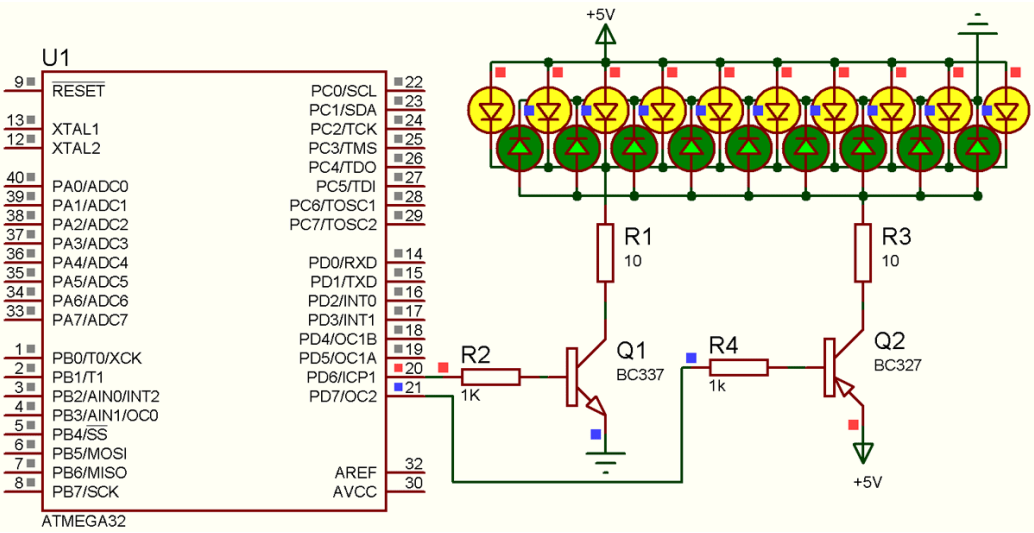
# الجلسة العملية الرابعة

## نظرة عامة (Overview):

هذه المحاضرة تقدم تطبيقاً عملياً لاستثمار أقطاب الدخل والخرج لمتحكمات AVR وبرمجتها في البيئة BASCOM-AVR ومحاكاتها في البيئة Proteus. حيث تقدم مجموعة من التجارب العملية التي تتضمن توصيل الثنائيات الضوئية وتصميم المفاتيح الترانزستورية لربط أقطاب المتحكم مع الأحمال. وكذلك ربط الواصلات الميكانيكية ولمفاتيح اللحظية.

## 1-4 برمجة بوابات الدخل والخرج في متحكمات AVR (Programming AVR MCUs GPIOs):

التجربة الرابعة: تم وصل عشرة ثنائيات ضوئية (LEDs) إلى متحكم ATmega32A على القطب PIND.6 ("1")، وعشرة أخرى إلى القطب PIND.7 ("0")، والمطلوب: كتابة برنامج خفقان لكلا المجموعتين معاً كل 0.5S وتصميم دارة المفتاح الترانزستوري.




الشكل4-1 توصيل الثنائيات مع المتحكم عن طريق مفاتيح تحكم ترانزستورية للتجربة 4

أولاً: إن الترانزستور الذي قمنا باختياره BC337 له المواصفات التالية:

$I_{Cmax}$=800mA, $V_{BE\_saturate}$=0.65V, $V_{CE\_saturate}$=0.2V, $h_{FE}$ = 100, $V_{CE\_max}$=50V

ثانياً: نحسب تيار الحمل (IC) الكلي علماً أن: $V_{LED}$ = 2.2V & $I_{LED}$ = 10mA:

$$I_C = 10 \times 10mA = 100mA$$

ثالثاً: نحسب مقاومة تحديد التيار RC واستطاعتها:

$$Rc = \frac{V_{cc} - V_{LED}}{I_C} = \frac{5 - 2.2}{100} = 28\Omega$$

$$P_{RC} = (V_{cc} - V_{LED}) \times I_C = (5 - 2.2) \times 100 = 280mW$$

رابعاً: نحسب قيمة التيار الأصغري اللازم لقيادة الترانزستور :

$$I_C = hfe \times I_B \rightarrow I_B = \frac{I_C}{hfe} = \frac{100}{100} = 1.0mA$$

$$Pc_{max} = U_{CE} \times I_C = 0.2 \times 100 = 20mW$$

خامساً: الآن يمكن حساب قيمة مقاومة القاعدة واستطاعتها من العلاقة التالية:

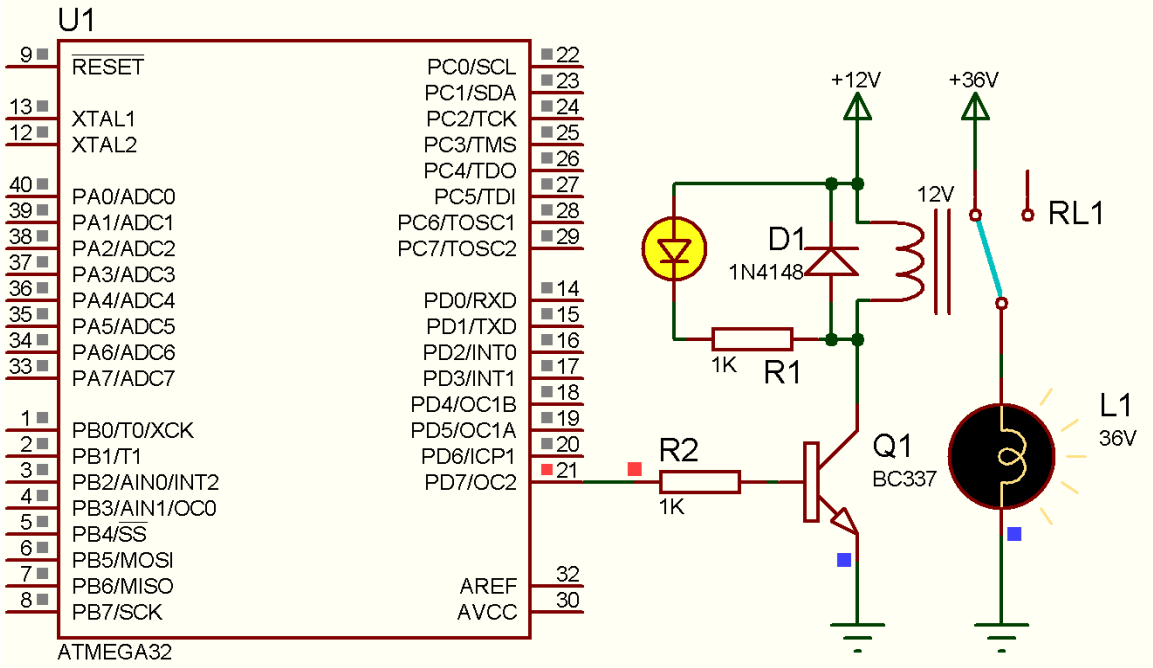
$$R_B = \frac{V_P - V_{BE}}{I_B} = \frac{5 - 0.7}{1} = 4.3K\Omega$$

$$P_{RC} = (V_P - V_{BE}) \times I_B = (5 - 0.7) \times 1 = 4.3mW$$

البرنامج Exp.04.bas في بيئة BASCOM-AVR:

```
' ********************************************************************************
' * Title          : Exp.04.bas                                                  *
' * Target MCU     : ATMega32A                                                   *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : LEDs                                                        *
' * Description    : GPIOs as Outputs                                            *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'-----------------------
'-----------------------[GPIO Configurations]
Config Pind.6 = Output
Leds_y Alias Portd.6 : Reset Leds_y

Config Pind.7 = Output
Leds_g Alias Portd.7 : Set Leds_g
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   '>[Turn Leds on]
   Set Leds_y : Set Leds_g : Waitms 500
   '>[Turn Leds off]
   Reset Leds_y : Reset Leds_g : Waitms 500
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

التجربة الخامسة: التحكم بمصباح كهربائي 36V تياره 5A عن طريق متحكم ATmega32A على القطب PIND.7، وكتابة برنامج بحيث يخفق المصباح كل 1Sec علماً أن جهد تغذية ملف الريليه هو 5V وتيار تشغيل ملف هو 100mA.




الشكل4-2 توصيل الريليه مع المتحكم عن طريق مفتاح تحكم ترانزستوري للتجربة 5

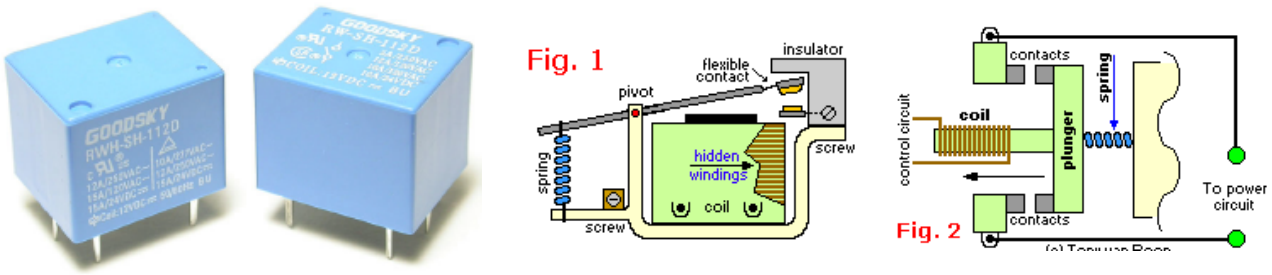
البرنامج Exp.05.bas في بيئة BASCOM-AVR:

```
' ************************************************************************************
' * Title          : Exp.05.bas                                                      *
' * Target MCU     : ATMega32A                                                       *
' * Author         : Walid Balid                                                     *
' * IDE            : BASCOM AVR 2.0.7.3                                              *
' * Peripherals    : LEDs                                                            *
' * Description    : GPIOs as Outputs                                                *
' ************************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'-----------------------
'-----------------------[GPIO Configurations]
Config Pind.7 = Output : Relay Alias Portd.7
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   Toggle Relay : Waitms 1000
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

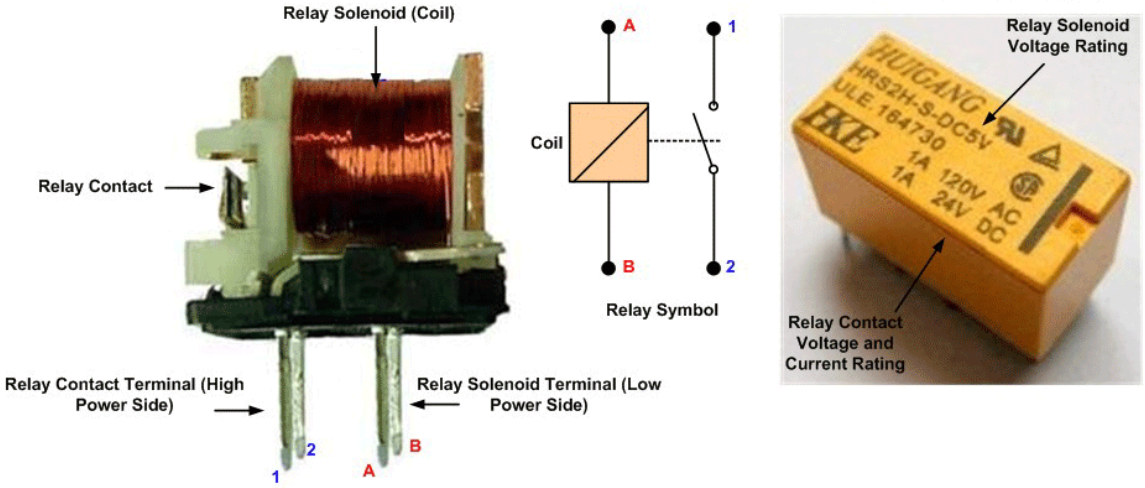
## 2-4 التحكم بالريليه باستخدام مفتاح ترانزستوري ثنائي القطبية (Driving Relays using BJT Control Switch):

تستخدم الريليه أو الوصل الميكانيكي (Relay) للتحكم بأحمال التيار المستمر والمتناوب التي لا تتطلب تحكماً سريعاً بالوصل والفصل – أي أن الريليه تستخدم كقاطع ميكانيكي متحكم به كهربائياً. تتوفر الريليه تجاري بجهود تحكم ذات مجال واسع نسبياً ومن هذه الجهود نذكر: 3V, 5V, 6V, 9V, 12V, 15V, 24V, 36V, 48V, 60V. كما أن التيار الذي يستجره ملف تشغيل الريليه يتراوح من 30mA ~ 300mA وذلك حسب حجم واستطاعة الريليه ويتناسب مع ذلك تناسباً طردياً.

تتألف الريليه من ملف (Coil) وهو مسؤول عن فصل ووصل الريليه (التحكم)، ومن تماسات استطاعية لقيادة الحمل واستطاعتها تختلف من ريليه لأخرى، ولكن معظم الريليهات المستخدمة في الدارات الإلكترونية تكون تماساتها قادرة على قيادة حمل بتيار من 3~10A عند جهد تشغيل الشبكة 220V. يبين الشكل4-3 والشكل4-4 رسماً تفصيلياً للبنية الداخلية للريليه حيث أنه عندما يتم تغذية ملف الريليه فإن الزراع الذي يحمل التماس المتحرك سوف ينجذب ويلامس التماس الثابت مؤدياً إلى وصل الدارة، وعندما يفقد الملف تهييجه تؤثر قوة النابض العكسية على الذراع وتعيده إلى وضعيته الأساسية.



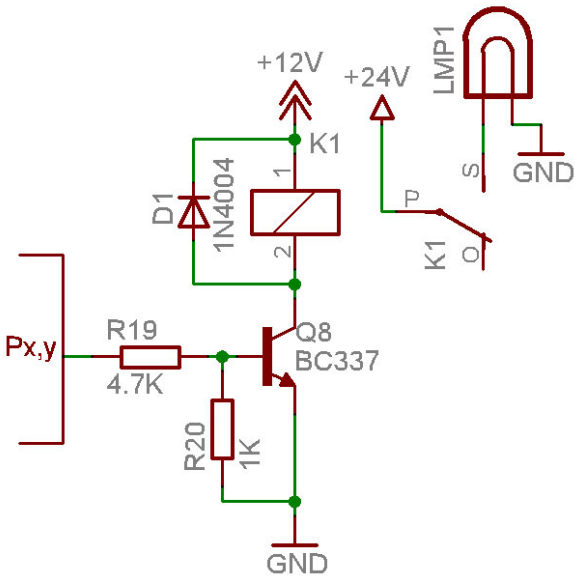

الشكل4-3 رسم تمثيلي للبنية الداخلية للريليه




الشكل4-4 رسم تفصيلي للبنية الداخلية للريليه

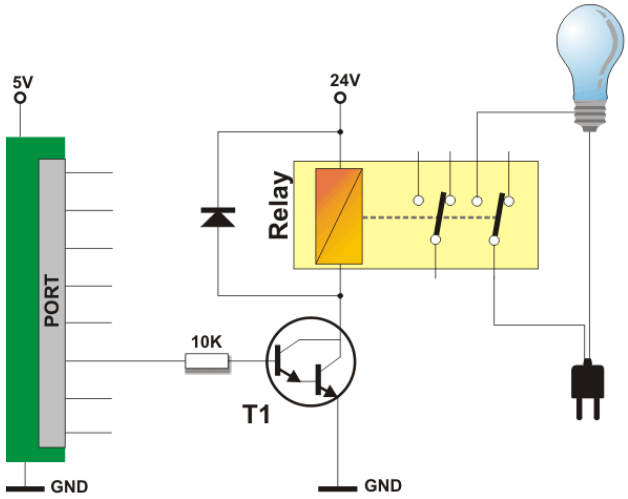
يبين الشكل4-5 دارة تحكم بريليه يعمل ملفها على جهد 12V وتتحكم بدورها بعمل مصباح تيار مستمر جهد تشغيله 24V، حيث أنه عند تطبيق "1" على قطب بوابة المتحكم فإن الترانزستور سوف يغلق مؤدياً إلى وصل النقطة الأرضية إلى الطرف الثاني من ملف الريليه، فيتهيج الملف مؤدياً بدوره إلى جذب تماس الريليه K1 وإغلاق النقطتين P, S، وعندها يضيء المصباح الكهربائي.

ملاحظة: من أجل حماية الترانزستور من أن يتم تدميره (حرقه) بسبب تيار التفريغ العكسي (Electromotive Force) لملف الريليه عند فصل الترانزستور، يتم إضافة ديود على التوازي مع ملف الريليه (D1) يسمى ديود المسار الحر والذي بدوره يشكل حلقة مغلقة لتفريغ تيار الملف عند قطع الترانزستور.




الشكل4-5 التحكم بحمل باستخدام ريليه

يبين الشكل4-6 دارة عملية لقيادة مصباح كهربائي متناوب ذو جهد 220V وتيار 1A عن طريق ريليه متحكم بها من متحكم مصغر.



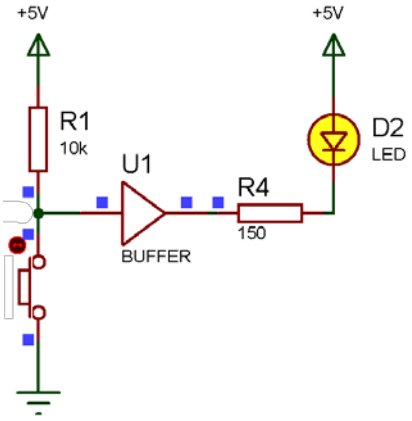

والشكل4-6 قيادة مصباح كهربائي متناوب عن طريق ريليه متحكم بها من متحكم مصغر

## 3-4 وصل المفاتيح اللحظية مع المتحكم المصغر (Interfacing Switches with MCUs):

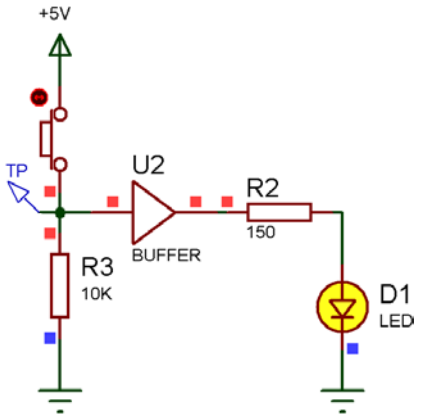
يوجد طريقتين لوصل المفاتيح مع أقطاب المتحكم المصغر:

1) المفتاح يطبق على قطب المتحكم القيمة المنطقية "0" عند ضغطه – الشكل4-7.
2) المفتاح يطبق على قطب المتحكم القيمة المنطقية "1" عند ضغطه – الشكل4-8.

على الشكل4-7 وعند ضغط المفتاح يتم توصيل النقطة الأرضية إلى قطب المتحكم، اما عند تحرير المفتاح فيتم تطبيق التغذية 5V+ على مدخل قطب المتحكم. وبالعكس تماماً تكون الحالة في طريقة التوصيل المبينة على الشكل4-8. وفي كلا الحالتين يقوم المتحكم في برنامجه الرئيسي بفحص حالة التغير على القطب المتصل مع المفتاح.

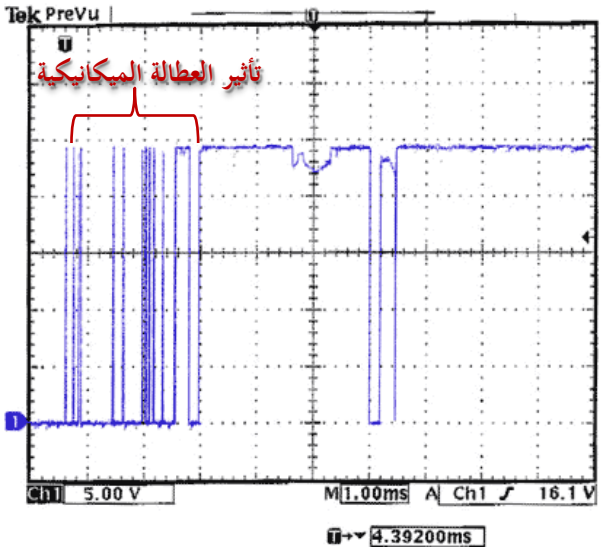



الشكل4-7 توصيل المفتاح ليكون فعال عند "0"




الشكل4-8 توصيل المفتاح ليكون فعال عند "1"

إن المفاتيح الميكانيكية لها تأثير سلبي عند ضغطها وتحريرها يسمى بالعطالة الميكانيكية للمفتاح والتي بدورها تسبب نشوء تغيرات سريعة في الإشارة على قطب المتحكم، هذه التغيرات ناتجة عن الاهتزاز الميكانيكي للمفتاح قبل أن تستقر الإشارة على الحالة المنطقية الحقيقية كما هو مبين على الشكل4-9.

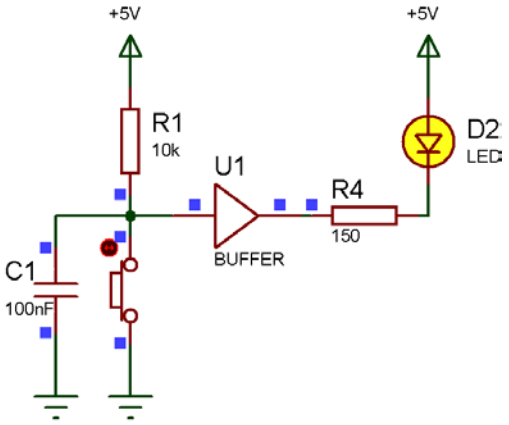



الشكل4-9 أثر العطالة الميكانيكية للمفتاح عند ضغطه وتحريره

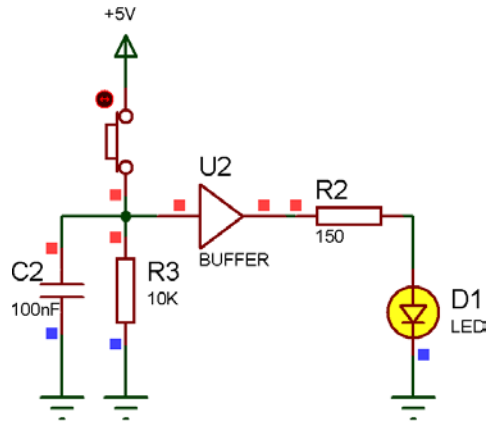
بشكل عام يوجد طريقتين للتخلص من العطالة الميكانيكية للمفاتيح وهما:

1) استخدام مكثف بقيمة تتراوح من 100nF – 1uF على التوازي مع المفتاح والذي سيقوم بدوره على تأخير التذبذب الناشئ كما هو مبين على الشكل4-10 والشكل4-11.

2) معالجة هذه الحالة برمجياً في برنامج المتحكم بفحص حالة المفتاح، وعند تحقق الشرط يتم توليد تأخير زمني 25~100ms وبعدها يتم فحص الحالة من جديد، فإذا بقيت الحالة مستقرة على الشرط المطلوب فيتم التنفيذ.




الشكل4-10 التخلص من الاهتزاز الميكانيكي للمفتاح ("0")




الشكل4-11 التخلص من الاهتزاز الميكانيكي للمفتاح ("1")

## 4-4 مقاومات الرفع والسحب (Pull-up & Pull-down Resistors):

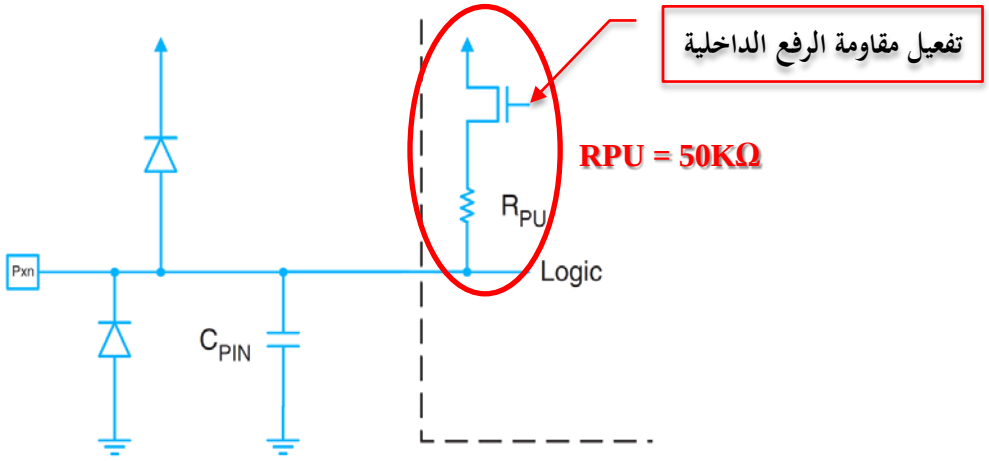
بالعودة إلى الشكل4-7 أو الشكل4-10 والذي تم فيهما توصيل المفتاح ليكون فعالاً عند "0" – أي بالضغط على المفتاح سيتم تطبيق صفر منطقي على قطب الدخل – فإن المقاومة R1 تمثل مقاومة رفع وظيفتها تأمين القيمة المنطقية "1" (+5V) على مدخل القطب عندما يكون المفتاح غير مضغوط، وبدونها ستكون الحالة على قطب الدخل غير معرّفة.

بالعودة إلى الشكل4-8 أو الشكل4-11 والذي تم فيهما توصيل المفتاح ليكون فعالاً عند "1" – أي بالضغط على المفتاح سيتم تطبيق واحد منطقي على قطب الدخل – فإن المقاومة R3 تمثل مقاومة سحب وظيفتها تأمين القيمة المنطقية "0" (GND) على مدخل القطب عندما يكون المفتاح غير مضغوط، وبدونها ستكون الحالة على قطب الدخل غير معرّفة.

قيمة مقاومات الرفع أو السبح تتراوح عادة بين القيمة 10KΩ ~ 50KΩ.

**هل يمكن الاستغناء عن مقاومة الرفع الخارجية في الشكل4-7 والشكل4-10 عند وصل المفتاح إلى قطب متحكم AVR؟؟**

تمتلك أقطاب متحكمات AVR عند استخدامها كأقطاب دخل مقاومات رفع داخلية قيمتها $R_{PU}$ = 50KΩ كما هو مبين على الشكل4-12 وهو مخطط البنية الداخلية لقطب دخل/خرج لمتحكم AVR. يمكن تفعيل أو إلغاء تفعيل هذه المقاومة لكل قطب دخل على حدى وبشكل افتراضي تكون هذه المقاومات غير مفعلة. ملاحظة: لا تملك متحكمات AVR مقاومات سحب داخلية وبالتالي يجب وضع مقاومات سحب خارجية من أجل التوصيلات في الشكل4-8 والشكل4-11.

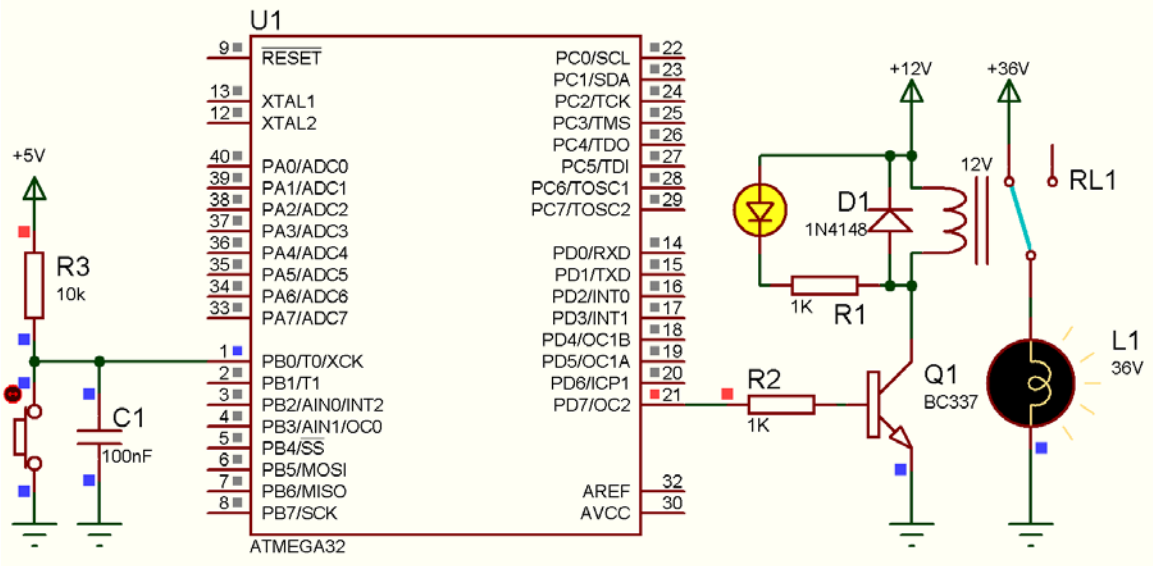



الشكل4-12 مخطط البنية الداخلية لقطب دخل/خرج لمتحكم AVR

## 4-5 تعليمات تعريف الأقطاب كمداخل في Bascom-AVR (Input Configuration Instructions in Bascom):

| وظيفة التعليمة | شكل التعليمة |
|---|---|
| تعريف البوابة C كبوابة دخل | `Config PORTC = Input` |
| تعريف القطب رقم 5 من البوابة C كقطب دخل | `Config PINC.5 = Input` |
| تفعيل مقاومات الرفع الداخلية للبوابة C كاملةً | `PORTC = 255` |
| إلغاء تفعيل مقاومات الرفع الداخلية للبوابة C كاملةً | `PORTC = 0` |
| تفعيل مقاومة الرفع الداخلية للقطب رقم 5 فقط من البوابة C | `PINC.5 = 1` |
| إلغاء تفعيل مقاومة الرفع الداخلية للقطب رقم 5 فقط من البوابة C | `PINC.5 = 0` |
| تفعيل بعض مقاومات الرفع الداخلية للبوابة C (PIN.4,5,6,7) | `PORTC = &B11110000` |
| يمكن استخدام هذا الشكل لتعريف الأقطاب من البوابة كدخل/خرج حيث أن (0) تعني قطب دخل، والـ (1) تعني قطب خرج. | `Config PORTC = &B11110000` |
| يصرح إلى أن PINC سوف يشار إليها أثناء البرنامج بالاسم (SWs) | `SWs Alias PINC` |
| يصرح إلى أن القطب PINC.5 سوف يشار إليه بالاسم (SW) | `SW Alias PINC.0` |

التجربة السادسة: المطلوب التحكم بتشغيل وفصل الريليه في التجربة الخامسة باستخدام مفتاح لحظي موصول إلى القطب PINB.0، بحيث أنه عند ضغط المفتاح تعمل الريليه وعند تحرير المفتاح تفصل الريليه – باستخدام مقاومة رفع خارجية.




الشكل4-13 التحكم بالريليه عن طريق مفتاح لحظي

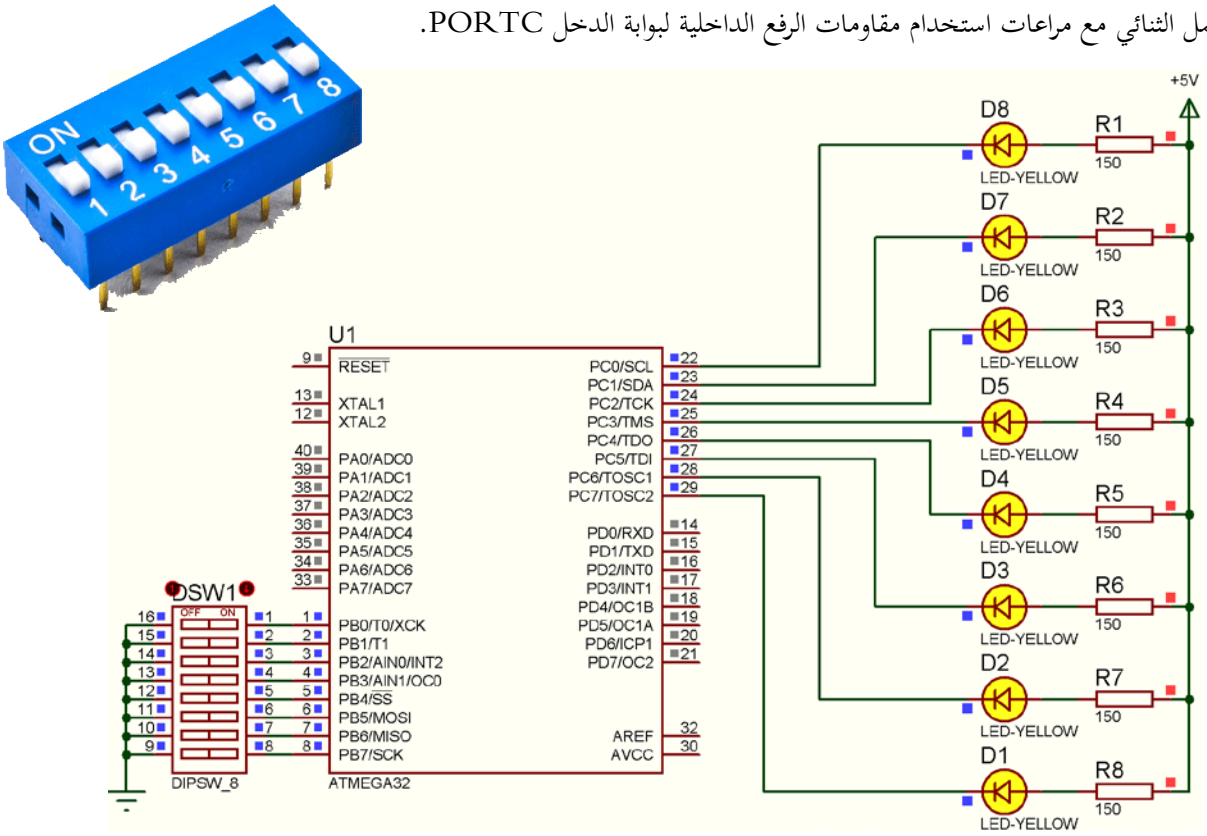
البرنامج Exp.06.bas في بيئة BASCOM-AVR:

```
' ********************************************************************************
' * Title          : Exp.06.bas                                                  *
' * Target MCU     : ATMega128A                                                  *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : Relay - Switch                                              *
' * Description    : GPIOs as Output/Input                                       *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'-----------------------
'-----------------------[GPIO Configurations]
Config Pind.7 = Output : Relay Alias Portd.7
Config Pinb.0 = Input : Switch Alias Pinb.0
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   If Switch = 0 Then Set Relay Else Reset Relay
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

**ملاحظة**: من أجل الاستغناء عن مقاومة الرفع الخارجية وتفعيل مقاومة الرفع الداخلية فإن كل ما نحتاجه هو إضافة تعليمة تفعيل مقاومة الرفع الداخلية للقطب PINB.0 بعد تعليمة تفعيل القطب وهي:

```
PORTB.0 = 1
```

التجربة السابعة: المطلوب التحكم بتشغيل وفصل ثمانية ثنائيات ضوئية موصولة إلى البوابة PORTC باستخدام مفتاح DIP-Switch موصول إلى البوابة PINB، بحيث أنه عند تفعيل المفتاح (on) يعمل الثنائي الموافق لرقم المفتاح وعند إلغاء تفعيل المفتاح (off) سوف يتوقف عمل الثنائي مع مراعات استخدام مقاومات الرفع الداخلية لبوابة الدخل PORTC.




الشكل4-14 توصيل المفاتيح الانزلاقية DIP-Switch

البرنامج Exp.07.bas في بيئة BASCOM-AVR:

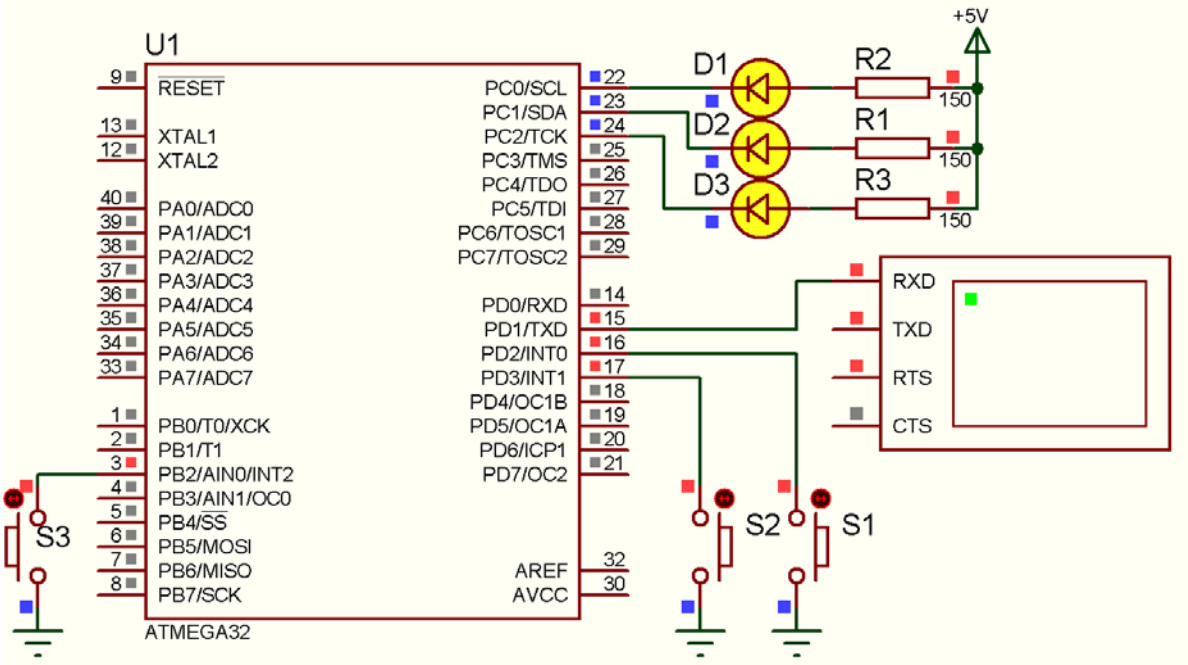
```
' ********************************************************************************
' * Title          : Exp.07.bas                                                  *
' * Target MCU     : ATMega128A                                                  *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : DIP-Switch                                                  *
' * Description    : GPIOs as Output/Input                                       *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'-----------------------
'-----------------------[GPIO Configurations]
Config Portc = Output : Leds Alias Portc
Config Portb = Input : Switchs Alias Pinb : Portb = &B11111111
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   Leds = Switchs
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

## 6-4 تعليمات قراءة حالة مفتاح موصول إلى قطب دخل في Bascom (Reading a Port PIN connected to a switch):

إن التجربة السادسة تضمنت قراءة حالة مفتاح لحظي من خلال قراءة القيمة الموجودة على القطب المتصل معه المفتاح، ومبدأ التخلص من العطالة الميكانيكية استخدم مكثف خارجي مع القطب. وتضمنت الفقرة 4-3 طريقتان للتخلص من العطالة الميكانيكية للمفاتيح، حيث أن الطريقة الثانية تستخدم تأخير زمني برمجي. التعليمات التالية يمكن استخدامها كتعليمات برمجية جاهزة في البيئة Bascom تسهل إلى حد كبير التعامل مع المفاتيح اللحظية وإضافة تأخير زمني.

| وظيفة التعليمة | شكل التعليمة |
|---|---|
| يراقب حالة القطب المحدد في Px.y كلما مر عليه، وعندما تصبح حالته موافقة للحالة المحددة في state، سوف يقفز إلى البرنامج الفرعي عند اللافتة label وينفذ البرنامج ويعود. | `Debounce Px.y , state , label , Sub`<br>`Ex. Debounce Key1 , 0 , Sw1 , Sub` |
| تهيئة زمن تأخير (ميلي ثانية) عن استعمال تعليمة Debounce للتخلص من العطالة الميكانيكية للمفتاح. | `Config Debounce = time` |
| سوف يقف البرنامج عند هذه التعليمة وينتظر أن تصبح حالة القطب صفر (reset) أو واحد (set) منطقي ليكمل البرنامج. | `Bitwait x , Set/reset`<br>`Ex. Bitwait Pinb.7 , reset` |

التجربة الثامنة: يوجد على اللوحة التعليمية Mini-Phoenix ثلاث مفاتيح لحظية موصولة إلى الأقطاب Pind2, Pind3, Pinb.2، كما يوجد ثمانية ثنائيات ضوئية (Leds) موصولة إلى البوابة PortC، والمطلوب: باستخدام التعليمة الشرطية If تغيير حالة عمل (Toggle) الثنائي D1 عن الضغط على المفتاح S1، وتغيير حالة D2 عند الضغط على S2، وتغيير حالة D3 عند الضغط على S3.




الشكل4-15 توصيل المفاتيح اللحظية الثنائيات مع المتحكم ATmega32A على اللوحة Mini-Phoenix

البرنامج Exp.08.bas في بيئة BASCOM-AVR:

```
' ********************************************************************************
' * Title          : Exp.08.bas                                                  *
' * Target Board   : Mini-Phoenix - REV 1.00                                     *
' * Target MCU     : ATMega32A                                                   *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : Pull-Up Resistors                                           *
' * Description    : GPIOs as Input; Active Low (GND)                            *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 9600
'-----------------------
'-----------------------[GPIO Configurations]
Config Portc = &B00000111
Led1 Alias Portc.0 : Led2 Alias Portc.1 : Led3 Alias Portc.2
Set Led1 : Set Led2 : Set Led3

Config Pind.2 = Input : Sw_1 Alias Pind.2 : Portd.2 = 1  'PU Internal Resistor
Config Pind.3 = Input : Sw_2 Alias Pind.3 : Portd.3 = 1
Config Pinb.2 = Input : Sw_3 Alias Pinb.2 : Portb.2 = 1
'-----------------------
'-----------------------[Variables]
Dim Count1 As Byte , Count2 As Byte , Count3 As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Print "Hello!"
Do
   If Sw_1 = 0 Then Gosub Sw_r1
   If Sw_2 = 0 Then Gosub Sw_r2
   If Sw_3 = 0 Then Gosub Sw_r3
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Print]
Sw_r1:
   Toggle Led1 : Count1 = Count1 + 1
   Print "Sw1 has Pressed! > " ; Count1
Return
'---<
Sw_r2:
   Toggle Led2 : Count2 = Count2 + 1
   Print "Sw2 has Pressed! > " ; Count2
Return
'---<
Sw_r3:
   Toggle Led3 : Count3 = Count3 + 1
   Print "Sw3 has Pressed! > " ; Count3
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

التجربة التاسعة: المطلوب تعديل البرنامج في التجربة الثامنة باستبدال التعليمة الشرطية If بالتعليمة Debounce.

البرنامج Exp.09.bas في بيئة BASCOM-AVR:

```
' ********************************************************************************
' * Title          : Exp.09.bas                                                  *
' * Target Board   : Mini-Phoenix - REV 1.00                                     *
' * Target MCU     : ATMega32A                                                   *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : Pull-Up Resistors                                           *
' * Description    : GPIOs as Input; Active Low (GND)                            *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 9600
'-----------------------
'-----------------------[GPIO Configurations]
Config Portc = &B00000111
Led1 Alias Portc.0 : Led2 Alias Portc.1 : Led3 Alias Portc.2
Set Led1 : Set Led2 : Set Led3

Config Pind.2 = Input : Sw_1 Alias Pind.2 : Portd.2 = 1    'PU Internal Resistor
Config Pind.3 = Input : Sw_2 Alias Pind.3 : Portd.3 = 1
Config Pinb.2 = Input : Sw_3 Alias Pinb.2 : Portb.2 = 1

Config Debounce = 50
'-----------------------
'-----------------------[Variables]
Dim Count1 As Byte , Count2 As Byte , Count3 As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Print "Hello!"
Do
   Debounce Sw_1 , 0 , Sw_r1 , Sub
   Debounce Sw_2 , 0 , Sw_r2 , Sub
   Debounce Sw_3 , 0 , Sw_r3 , Sub
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Print]
Sw_r1:
   Toggle Led1 : Count1 = Count1 + 1
   Print "Sw1 has Pressed! > " ; Count1
Return
'---<
Sw_r2:
   Toggle Led2 : Count2 = Count2 + 1
   Print "Sw2 has Pressed! > " ; Count2
Return
'---<
Sw_r3:
   Toggle Led3 : Count3 = Count3 + 1
   Print "Sw3 has Pressed! > " ; Count3
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

التجربة العاشرة: المطلوب تعديل البرنامج في التجربة الثامنة باستبدال التعليمة الشرطية If بالتعليمة Bitwait.

البرنامج Exp.10.bas في بيئة BASCOM-AVR:

```
' ************************************************************************
' * Title          : Exp.10.bas                                          *
' * Target Board   : Mini-Phoenix - REV 1.00                             *
' * Target MCU     : ATMega32A                                           *
' * Author         : Walid Balid                                         *
' * IDE            : BASCOM AVR 2.0.7.3                                  *
' * Description    : GPIOs as Input; Active Low (GND)                    *
' ************************************************************************
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 9600
'-----------------------
'-----------------------[GPIO Configurations]
Config Portc = &B00000111
Led1 Alias Portc.0 : Led2 Alias Portc.1 : Led3 Alias Portc.2
Set Led1 : Set Led2 : Set Led3

Config Pind.2 = Input : Sw_1 Alias Pind.2 : Portd.2 = 1    'PU Internal Resistor
Config Pind.3 = Input : Sw_2 Alias Pind.3 : Portd.3 = 1
Config Pinb.2 = Input : Sw_3 Alias Pinb.2 : Portb.2 = 1
'-----------------------
'-----------------------[Variables]
Dim Count1 As Byte , Count2 As Byte , Count3 As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   Print "PC is waiting for Sw1"
   Bitwait Sw_1 , Reset : Gosub Sw_r1

   Print "PC is waiting for Sw2"
   Bitwait Sw_2 , Reset : Gosub Sw_r2

   Print "PC is waiting for Sw3"
   Bitwait Sw_3 , Reset : Gosub Sw_r3
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Print]
Sw_r1:
   Toggle Led1 : Count1 = Count1 + 1
   Print "Sw1 has Pressed! > " ; Count1
Return
'---<
Sw_r2:
   Toggle Led2 : Count2 = Count2 + 1
   Print "Sw2 has Pressed! > " ; Count2
Return
'---<
Sw_r3:
   Toggle Led3 : Count3 = Count3 + 1
   Print "Sw3 has Pressed! > " ; Count3
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

ملاحظة: شرح البرامج موجود في ملف الفيديو للمحاضرة الخامسة.

. . . ﴿انتهت الجلسة العملية الرابعة﴾ . . .

وليد بليد - دمتم بخير ومودة ونور -

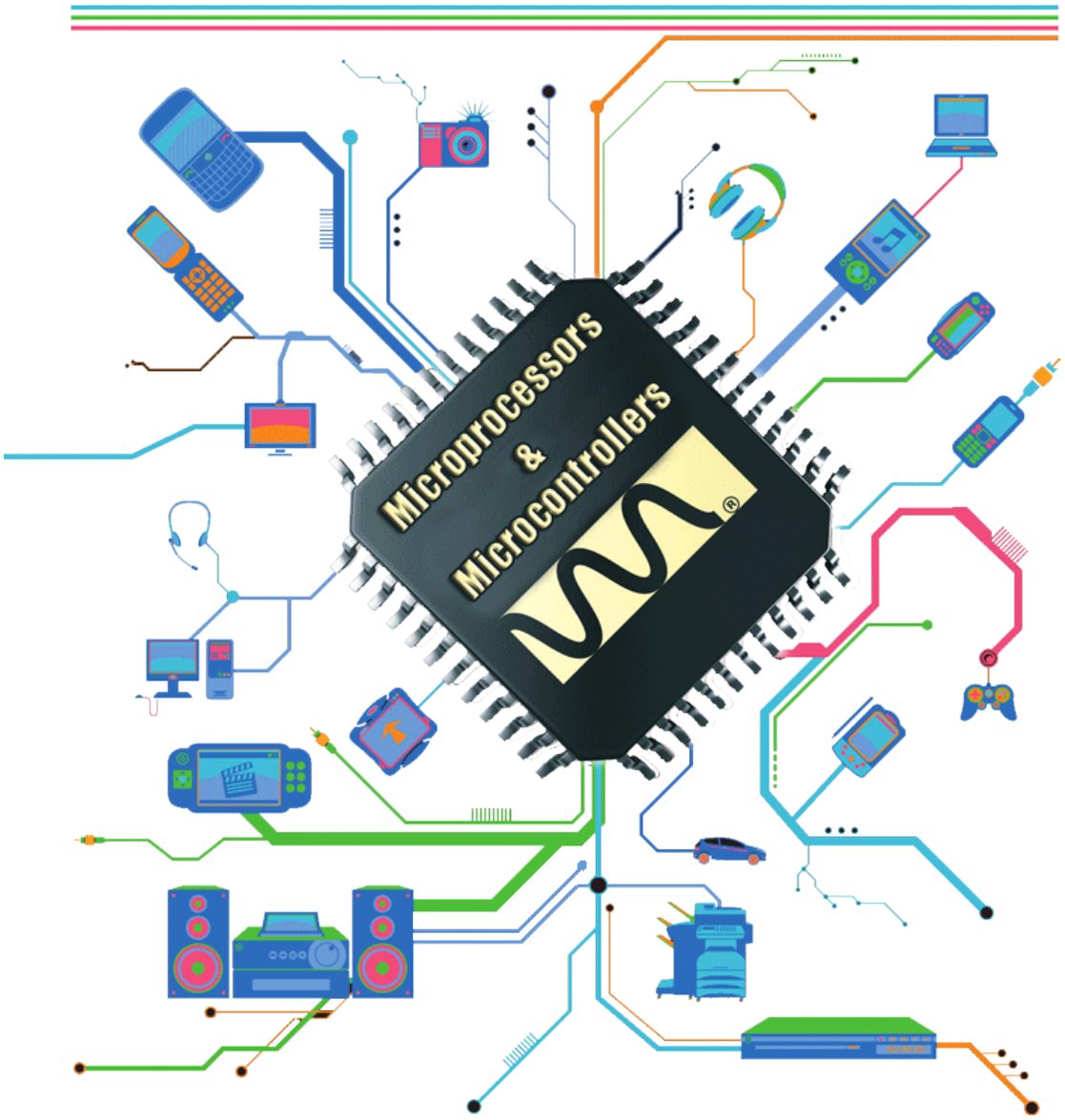
جامعـــــــــــة حلـــــــــــب
كلية الهندسة الكهربائية والإلكترونية
قسـم هندسـة التحكـم والأتمتـة

# ﴿الجلسات العملية لمادة المعالجات والمتحكمات المصغرة﴾

# Microprocessors & Microcontrollers Lab Sessions

السنة الثالثة | قسم اتصالات

## ﴿الجلسة العملية الخامسة﴾




م. وليد بليد

Wednesday, April 04, 2012

Copyright © 2012 Walid Balid - All rights reserved.

# الجلسة العملية الخامسة

## نظرة عامة (Overview):

هذه المحاضرة تشرح تعليمات الإزاحة والدوران وتقدم مثالاً تطبيقياً عليها. ثم تقدم المقاطعات في متحكمات AVR وأنواعها وتشرع في المقاطعات الخارجية ومبدأ عملها ولمحة عن المسجلات الداخلية للمقاطعات الخارجية. ثم تقدم تطبيقاً عملياً لاستثمار المقاطعات الخارجية في متحكمات AVR وبرمجتها في البيئة BASCOM-AVR ومحاكاتها في البيئة Proteus. وأخيراً طريقة توصيل لوحة مفاتيح مصفوفية ومنهجية المسح.

## 1-5 عمليات الإزاحة والتدوير (Shifting and Rotating):

تستخدم تعليمات الإزاحة والدوران بهدف إزاحة بت أو أكثر – من بايت أو أكثر – إلى اليمين أو إلى اليسار؛ وهناك فرق بين عملية الإزاحة وعملية الدوران لقيمة ما حيث:

- في الإزاحة كل بت يخرج (من اليمين أو اليسار) يدخل مكانه صفر. مثال ذلك: إذا تم إزاحة القيمة &B11111111 ثمان مرات إلى اليمين أو اليسار فستصبح القيمة عندها &B00000000.

مثال:

```
A = &B11011011
Shift A , Right , 1
```

| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

```
Shift A , Right , 4
```

| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

- في الدوران كل بت يخرج (من اليمين أو اليسار) يدخل من الطرف الآخر – أي يتم تدوير القيمة. مثال ذلك: إذا تم تدوير القيمة &B11111111 ثمان مرات إلى اليمين أو اليسار فستبقى القيمة على حالها. وإذا تم تدوير القيمة &B00001111 أربع مرات إلى اليمين أو اليسار فستصبح &B11110000.

مثال:

```
A = &B11011011
Rotate A , Right , 1
```

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

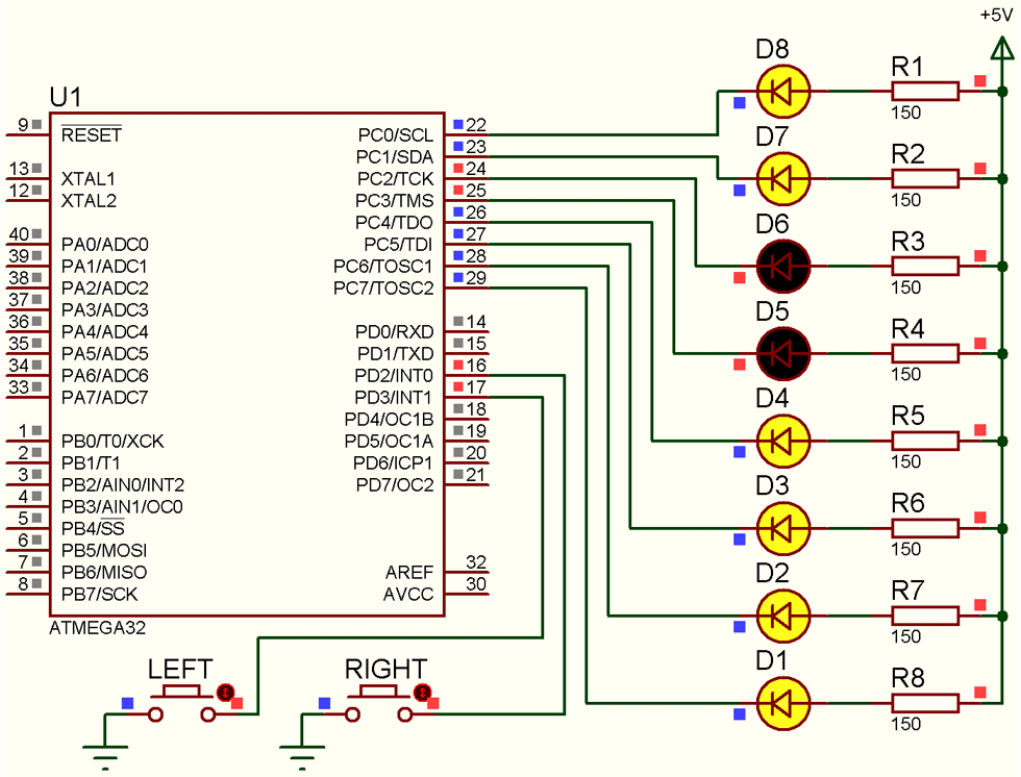
```
Rotate A , Right , 4
```

| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

## 2-5 تعليمات الإزاحة والتدوير في Bascom (Shifting and Rotating Instructions in BASCOM-AVR):

| شرح التعليمة | التعليمة البرمجية |
|---|---|
| إزاحة بت من متحول (var) إلى اليمين أو اليسار وعدد خانات الإزاحة محددة بـ [, shift] | `Shift var , Right/Left [, shift]` |
| تدوير بت من متحول (var) إلى اليمين أو اليسار وعدد خانات الدوران محددة بـ [, shift] | `Rotate var , Right/Left [, rotate]` |

التجربة الحادية عشرة: استخدم المفاتيح اللحظية S1-S2 (PIND.3, PIND.2) على اللوحة التعليمية لإزاحة وتدوير قيمة تظهر على الثنائيات الضوئية الثمانية (LEDs) الموصولة إلى البوابة PORTC.




الشكل5-1 توصيل الثنائيات والمفاتيح مع المتحكم للتجربة 11

البرنامج Exp.11.bas في بيئة BASCOM-AVR:

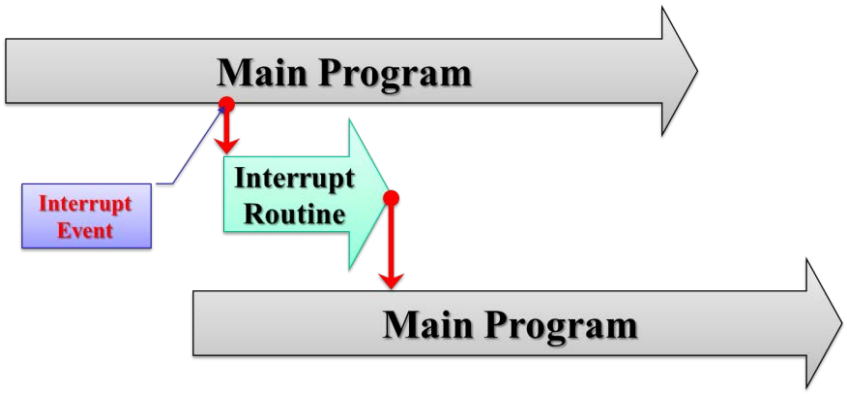
```
' ********************************************************************************
' * Title          : Exp.11.bas                                                  *
' * Target MCU     : ATMega128A                                                  *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : DIP-Switch                                                  *
' * Description    : Shift/Rotate                                                *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'------------------------
'-----------------------[GPIO Configurations]
Config Portc = Output : Leds Alias Portc

Config Pind.2 = Input : Sw_1 Alias Pind.2 : Portd.2 = 1    'PU Internal Resistor
Config Pind.3 = Input : Sw_2 Alias Pind.3 : Portd.3 = 1
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Leds = &B11011011
Do
   Debounce Sw_1 , 0 , Shift_r , Sub
   Debounce Sw_2 , 0 , Shift_l , Sub
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Shift LEDs to Right]
Shift_r:
  Shift Leds , Right , 1
   'Rotate Leds , Right , 1
Return
'---<
'--->[Shift LEDs to Left]
Shift_l:
   Shift Leds , Left , 1
   'Rotate Leds , Left , 1
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

## 5-3 المقاطعات في متحكمات AVR (Interrupts in AVR MCUs):

تعرف المقاطعة (Interrupt) بأنها آلية إعلام داخلية (دون تدخل المتحكم في آلية عمل المقاطعة داخلياً) تعلم وحدة المعالجة المركزية بوجود حدث يجب معالجته مما يسبب تغير في سير البرنامج الرئيسي لإنجاز برنامج فرعي يسمى برنامج خدمة المقاطعة. لتوضيح الفكرة نأخذ على سبيل المثال برنامج قراءة حالة مفتاح موصول إلى قطب المتحكم، وبالتالي يوجد لحالتين لبرمجة عمل المفتاح:

1) الطريقة التقليدية (Polling): يمكن أن نطلق على هذه الطريقة المقاطعة البرمجية (Software Interrupts) وتتم بالفحص الدوري لحالة القطب (... **If** Sw = 0 **Then**) من أجل اكتشاف تغير حالة المفتاح ويتم هذا بشكل برمجي، وبالتالي سوف يشغل المتحكم في عملية الفحص الدوري المتكرر للتحقق من حالة المفتاح بشكل دائم، الأمر الذي سيؤدي إلى ضياع في قدرة المعالجة للمتحكم واستهلاك في الطاقة.

2) طريقة المقاطعة (Interrupt): يمكن أن نطلق على هذه الطريقة مقاطعة الكيان الصلب (Hardware Interrupts) وتتم من خلال آلية مستقلة مبنية ضمن المتحكم تقوم على مقاطعة المتحكم عندما تتحقق الحالة المطلوبة فقط، وبالتالي لن ينشغل المتحكم بتفحص المفاتيح من أجل معرفة فيما إذا تغيرت حالة المفتاح أم لا، وإنما عندما تتغير الحالة المنطقية للمفتاح على قطب المقاطعة سوف يتم مقاطعة المتحكم ويقفز إلى برنامج خدمة المقاطعة الخارجية المتعينة من أجل تنفيذها.




الشكل5-2 تمثيل عملية المقاطعة والقفز من البرنامج الرئيسي إلى برنامج المقاطعة والعودة إلى البرنامج الرئيسي

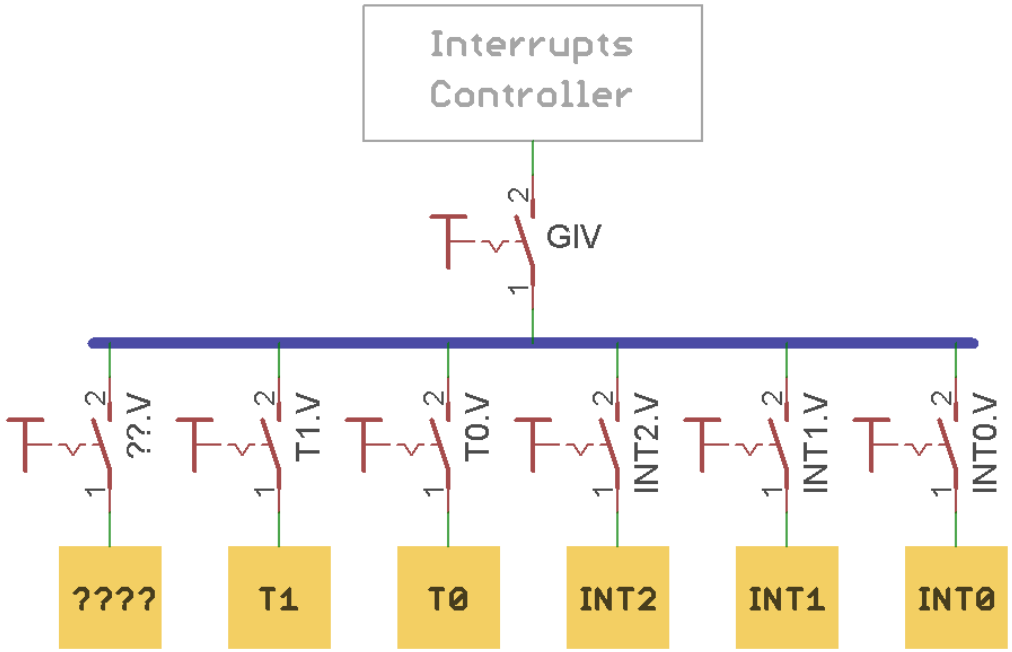
## 5-4 مصادر المقاطعات في متحكمات AVR (AVR MCU Interrupts Sources):

تمتلك العائلة AVR مجموعة كبيرة من مصادر المقاطعة المختلفة، وتمتلك كل من هذه المقاطعات عنوان مستقل في حيز ذاكرة البرنامج، ولكل مقاطعة خانة تمكين مستقلة، فعندما نرغب بتفعيل إحدى المقاطعات فإنه يتوجب علينا تفعيل الخانة المخصصة لها في مسجل التحكم بالمقاطعة المعنية إلى جانب تفعيل خانة تمكين المقاطعة العامة I في مسجل الحالة SREG.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **SREG** | **I** | **T** | **H** | **S** | **V** | **N** | **Z** | **C** |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

الشكل5-3 الخانة 7 شعاع المقاطعات العام (Global Interrupt Vector) في مسجل الحالة SREG

يمكن تمثيل شعاع المقاطعات العام (Global Interrupt Vector) بقاطع رئيسي، وباقي المقاطعات كقواطع فرعية، وبالتالي لا يكفي تفعيل المقاطعة الفرعية وإنما يجب أيضاً تفعيل شعاع المقاطعات العام معها كما هو مبين على الشكل التالي.




الشكل5-4 تمثيل لحالة شعاع المقاطعات العام (Global Interrupt Vector) والمقاطعات الفرعية الأخرى

## 5-5 تصنيف المقاطعات في متحكمات AVR (AVR MCU Interrupts Classification):

يمكن تقسيم المقاطعات في العائلة AVR إلى مجموعتين رئيسيتين، تضم كل مجموعة من المجموعتين مجموعات فرعية أخرى:

- مقاطعات خارجية (External Interrupts): لها ارتباط مباشر مع الأقطاب الفيزيائية للمتحكم وتستجيب لأحداث خارجية مطبقة على أقطاب المتحكم وهي:
  - مقاطعة التصفير (Reset).
  - مقاطعات الطلب الخارجي (INT0 ~ INT7).
- مقاطعات داخلية (Internal Interrupts): لها ارتباط مع الوحدات المحيطية الداخلية فقط للمتحكم وهي:
  - مقاطعات المؤقتات (OV, COMP).
  - مقاطعات حادثة المسك للمؤقتات/عدادات (ICP).
  - مقاطعات العدادات.
  - مقاطعة اكتمال التحويل للADC.
  - مقاطعة اكتمال الإرسال للنافذة التسلسلية SPI (STC).
  - مقاطعات النافذة التسلسلية USART (RX,TX,UDR).

- مقاطعة المقارن التشابهي (ANALOG COMP).
- مقاطعة اكتمال كتابة المعطيات إلى الذاكرة EEPROM.
- مقاطعة النافذة التسلسلية TWI.
- وغيرها...

## 6-5 مبدأ عمل المقاطعات في متحكمات AVR (AVR MCU Interrupts Basic):

تتمتع وحدة المقاطعات بمسجلات تحكم خاصة في المساحة المخصصة للدخل/الخرج بالإضافة إلى خانة تفعيل خاصة (I) ضمن مسجل الحالة، وتتمتع كل مقاطعة بشعاع مقاطعة منفصل في جدول أشعة المقاطعات، ويكون لكل مقاطعة أولوية متناسبة مع موقع شعاعها ضمن الجدول، فكلما كان عنوان شعاع المقاطعة أدنى كلما كانت ذات أولوية أعلى.

| Vector No. | Program Address(2) | Source | Interrupt Definition |
|---|---|---|---|
| 1 | $000(1) | RESET | External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset |
| 2 | $002 | INT0 | External Interrupt Request 0 |
| 3 | $004 | INT1 | External Interrupt Request 1 |
| 4 | $006 | INT2 | External Interrupt Request 2 |
| 5 | $008 | TIMER2 COMP | Timer/Counter2 Compare Match |
| 6 | $00A | TIMER2 OVF | Timer/Counter2 Overflow |
| 7 | $00C | TIMER1 CAPT | Timer/Counter1 Capture Event |
| 8 | $00E | TIMER1 COMPA | Timer/Counter1 Compare Match A |
| 9 | $010 | TIMER1 COMPB | Timer/Counter1 Compare Match B |
| 10 | $012 | TIMER1 OVF | Timer/Counter1 Overflow |
| 11 | $014 | TIMER0 COMP | Timer/Counter0 Compare Match |
| 12 | $016 | TIMER0 OVF | Timer/Counter0 Overflow |
| 13 | $018 | SPI, STC | Serial Transfer Complete |
| 14 | $01A | USART, RXC | USART, Rx Complete |
| 15 | $01C | USART, UDRE | USART Data Register Empty |
| 16 | $01E | USART, TXC | USART, Tx Complete |
| 17 | $020 | ADC | ADC Conversion Complete |
| 18 | $022 | EE_RDY | EEPROM Ready |
| 19 | $024 | ANA_COMP | Analog Comparator |
| 20 | $026 | TWI | Two-wire Serial Interface |
| 21 | $028 | SPM_RDY | Store Program Memory Ready |

الشكل5-5 عناوين أشعة المقاطعات في المتحكم ATmega32A

الشكل5-5 يبين القائمة الكاملة للمقاطعات في المتحكم ATmega32A وعناوين الأشعة لهذه المقاطعات، كما تحدد هذه القائمة أيضاً مستويات الأولوية للمقاطعات، فالعنوان الأخفض (000$) هو الشعاع ذو الأولوية الأعلى، فلمقاطعة التصفير مثلاً الأولوية الأعلى ومن ثم المقاطعة الخارجية INT0 وهكذا...

عندما تحدث مقاطعة ما فإنه يتم **تلقائياً** تصفير خانة تمكين المقاطعة العامة (I = 0)، وبالتالي تحجب جميع المقاطعات الأخرى إلى حين الانتهاء من المقاطعة الحالية، إلا أن المبرمج يستطيع أن يُفعّل خانة تمكين المقاطعة العامة (I = 1) داخل برنامج خدمة المقاطعة في حال أريد الإبقاء على المقاطعات الأخرى. وعندما ينفذ المتحكم تعليمة العودة RETURN الواقعة في نهاية برنامج خدمة المقاطعة، فإنه يتم **تلقائياً** تفعيل خانة تمكين المقاطعة العامة (I = 1).

## 7-5 ملاحظات هامة حول المقاطعات في متحكمات AVR:

- عند استخدام أي مقاطعة فإنه يجب تفعيل شعاع المقاطعات العام I والذي يمكن تمثيله كقاطع رئيسي لجميع المقاطعات.
- عندما تحدث مقاطعة ما يتم تلقائياً تصفير شعاع المقاطعات العام I في مسجل الحالة وبذلك يتم إلغاء الاستجابة لجميع المقاطعات الأخرى، وتتم عملية إعادة تفعيل الخانة I بعد الانتهاء من تنفيذ أية مقاطعة تلقائياً أيضاً.
- إذا تحقق شرط إحدى المقاطعات أو أكثر، وكانت خانة تمكين المقاطعة العامة غير مفعلة (I = 0)، فإن أعلام المقاطعة التي حدثت ستفعل (''1'') تلقائياً وتبقى كذلك إلى أن يتم تأهيل خانة المقاطعة العامة (I = 1)، فإذا ما تم تفعيل خانة المقاطعة العامة (I = 1) عندها يبدأ المتحكم بتنفيذ برامج خدمة المقاطعة بحسب أولويات أشعتها.
- في حال كان المتحكم يقوم بتنفيذ برنامج خدمة مقاطعة ما، وفي نفس الوقت حصلت مقاطعة أخرى، فإن المتحكم سوف يكمل المقاطعة الجارية ويقوم بتخزين المقاطعة الطارئة حتى إذا انتهى من المقاطعة الجارية عاد إلى البرنامج الرئيسي ونفذ تعليمة واحدة على الأقل من البرنامج الرئيسي ثم سيستدعى المقاطعة الطارئة ويقوم بتنفيذها. وأما في حال حصلت عدة مقاطعات أثناء عمل المتحكم في برنامج خدمة مقاطعة ما، فإنه يقوم بمراكمتها حسب أولويتها ويقوم بتنفيذها وفق تسلسل الأولوية بعد انتهائه من برنامج خدمة المقاطعة الجارية.
- عندما ينتهي تنفيذ برنامج خدمة مقاطعة ما، فإن سيتم العودة إلى البرنامج الرئيسي وينفذ المتحكم تعليمة واحدة على الأقل قبل أن ينتقل لتنفيذ مقاطعة أخرى في حال وجود مقاطعات متراكمة أثناء برنامج خدمة المقاطعة الأخيرة.
- ينصح بأن يكون برنامج خدمة المقاطعة قصيراً جداً (يمكن تفعيل علم تحقق المقاطعة وتفحص العلم في البرنامج الرئيسي وتنفيذ جملة تعليمات تبعاً لحالة علم المقاطعة) وجميع المعالجَات تتم في البرنامج الرئيسي من أجل الاستجابة المباشرة للمقاطعات الأخرى حال حصولها.

### 8-5 زمن استجابة المقاطعة (Interrupt Response Time):

إن الاستجابة الزمنية عند تنفيذ المقاطعات بالنسبة لمتحكمات عائلة AVR هي على الأقل أربع دورات ساعة (4-Cycle) يتم خلالها دفع (Push) محتوى عداد البرنامج PC (Program Counter) إلى المكدس SP ويستهلك الدفع 2-Cycle، ومن ثم يقفز البرنامج إلى برنامج خدمة المقاطعة ويستهلك القفز 2-Cycle. وإذا حدثت المقاطعة أثناء تنفيذ إحدى التعليمات التي زمن تنفيذها أكبر من دورة واحدة، فإنه يتم استكمال تنفيذ التعليمة قبل الانتقال إلى برنامج خدمة المقاطعة. إن العودة من برنامج خدمة المقاطعة تستهلك أربع دورات ساعة (4-Cycle) أيضاً يتم خلالها سحب (Pull) قيمة عداد البرنامج PC من المكدس SP ويستهلك السحب 2-Cycle، ومن ثم يقفز إلى البرنامج الرئيسي ويستهلك القفز 2-Cycle، وينفذ ابتداءً من التعليمة التالية للتعليمة التي حدثت عندها المقاطعة.

### 9-5 المقاطعات الخارجية في متحكمات AVR (AVR MCU External Interrupts):

تمتلك متحكمات العائلة AVR أقطاب مخصصة للمقاطعات الخارجية والتي يرمز لها INT0, INT1, ….., INT7. الهدف من هذه المقاطعات الخارجية هو الاستجابة لأحداث معينة تطبق على أقطاب هذه المقاطعات. تملك هذه المقاطعات الخارجية أنماط استجابة متعددة للجبهات المطبقة عليها وكذلك يمكن تفعيلها أو إلغاء تفعيلها من خلال مجموعة من مسجلات التحكم الخاصة بهذه المقاطعات.

#### 9-5-1 مسجلات التحكم بالمقاطعات الخارجية في متحكمات AVR (AVR MCUs External Interrupt Registers):

تملك المقاطعات الخارجية ثلاث مسجلات تحكم وهي:

- مسجل التحكم بنمط عمل المقاطعة الخارجية MCUCR (MCU Control Register).
- مسجل التحكم بالمقاطعات الخارجية GICR (Global Interrupt Mask Register).
- مسجل أعلام المقاطعات الخارجية GIFR (Global Interrupt Flag Register).

1) **مسجل التحكم بنمط عمل المقاطعة الخارجية MCUR**: يتم من خلاله التحكم بحساسية أو نمط استجابة المقاطعة للحدث الخارجي المطبق على قطب المقاطعة ويوجد أربع حالات وهي:
   1) تقدح عن الجبهة الصاعدة (Rising Edge).
   2) تقدح عن الجبهة الهابطة (Falling Edge).
   3) تقدح عن مستوى الجبهة (Low Level).
   4) تقدح عن تغير المستوى (Level Change).

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | SE | SM2 | SM1 | SM0 | ISC11 | ISC10 | ISC01 | ISC00 | MCUCR |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

الشكل5-6 مسجل التحكم بنمط عمل المقاطعة الخارجية **MCUR** للمقاطعات [INT0/INT1]

على اعتبار وجود أربعة حالات لنمط عمل كل مقاطعة من المقاطعات الخارجية فإن هذا سيحتاج إلى خانتين في مسجل التحكم بنمط عمل المقاطعة الخارجية لكل مقاطعة حيث تمثل الخانتين ISC01|ISC00 خانتي التحكم بنمط عمل المقاطعة الخارجية INT0، وتمثل الخانتين ISC11|ISC10 خانتي التحكم بنمط عمل المقاطعة الخارجية INT1 وهكذا باقي المقاطعات الأخرى.

| ISC11 | ISC10 | Description |
|---|---|---|
| 0 | 0 | The low level of INT1 generates an interrupt request. |
| 0 | 1 | Any logical change on INT1 generates an interrupt request. |
| 1 | 0 | The falling edge of INT1 generates an interrupt request. |
| 1 | 1 | The rising edge of INT1 generates an interrupt request. |

الشكل5-7 اختيار حالات نمط استجابة المقاطعة للحدث للمقاطعة INT1

2) **مسجل التحكم بالمقاطعات الخارجية GICR**: تمثل كل خانة من الخانات الثلاث (5,6,7) في المسجل خانة لتفعيل طلب مقاطعة خارجية من المقاطعات الخارجية الثلاثة (INT0, INT1, INT2) للمتحكم ATmega32A، حيث أنه عند وضع القيمة "**1**" في خانة المسجل **GICR**.n فإنه يتم تمكين المقاطعة الموافقة لهذه الخانة بشرط أن تكون الخانة I في مسجل الحالة SREG مفعلة ("1")، أما عند وضع "**0**" في خانة المسجل **GICR**.n فإنه يتم إلغاء تمكين المقاطعة الموافقة (n).

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | INT1 | INT0 | INT2 | – | – | – | IVSEL | IVCE | GICR |
| Read/Write | R/W | R/W | R/W | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

الشكل5-8 مسجل التحكم المقاطعات الخارجية GICR

3) **مسجل أعلام المقاطعات الخارجية GIFR**: تمثل كل خانة من الخانات الثلاث (5,6,7) في المسجل علم يشير لحدوث مقاطعة خارجية (GIFR.n = "1") من المقاطعات الثلاث (INT0, INT1, INT2) للمتحكم ATmega32، وبالتالي سوف يقفز المتحكم إلى شعاع المقاطعة المتوضع عند العنوان المحدد في ذاكرة البرنامج لينفذ برنامج خدمة المقاطعة، وعند العودة من برنامج خدمة المقاطعة سيتم تصفير هذا العلم بشكل تلقائي من قبل الكيان الصلب.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | INTF1 | INTF0 | INTF2 | – | – | – | – | – | GIFR |
| Read/Write | R/W | R/W | R/W | R | R | R | R | R | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

الشكل5-9 مسجل أعلام المقاطعات الخارجية GIFR

## 10-5 برمجة المقاطعات الخارجية في BASCOM-AVR (Programming External Interrupts in BASCOM):

بشكل عام فإنه من أجل برمجة المقاطعات الخارجية فإنه يجب:

✓ تحديد نمط عمل (State: `Rising | Falling | Low | Level`) المقاطعة الخارجية (`INTx`).

```
Config INTx = State
```

✓ تحديد اسم البرنامج الفرعي (Label) للمقاطعة (`INTx`).

```
On INTx Label
```

✓ تفعيل (`Enable`) شعاع المقاطعة المطلوبة تشغيلها.

```
Enable INTx
```

✓ تفعيل شعاع المقاطعات العام.

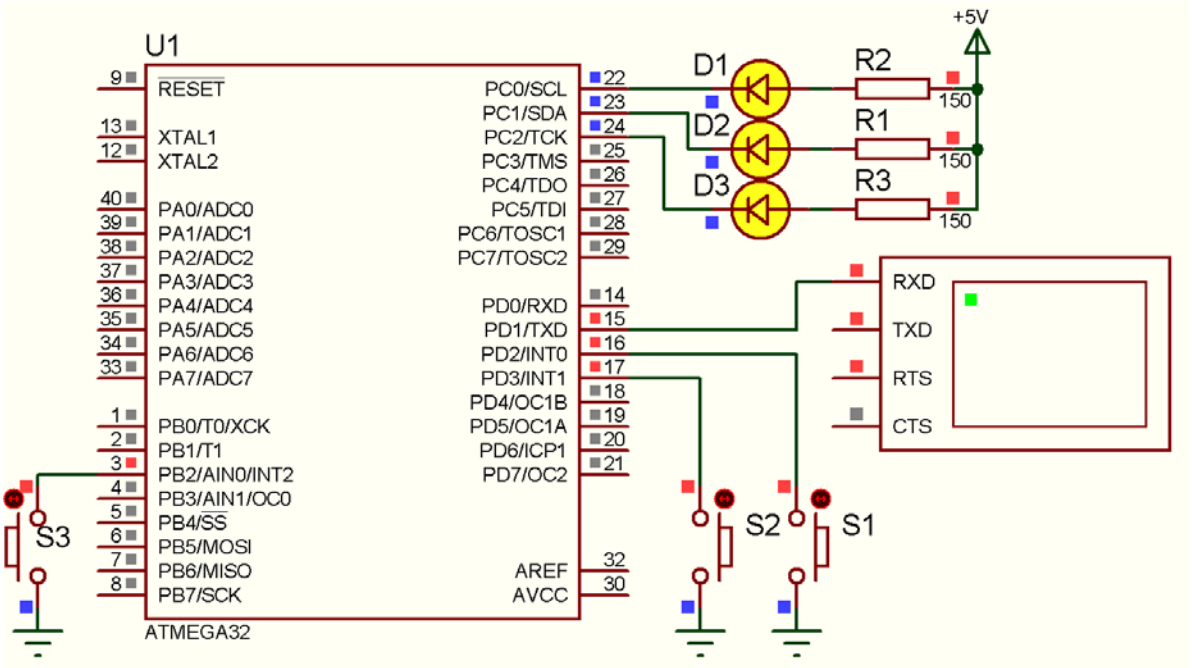
```
Enable Interrupts
```

ملاحظة1: يمكن أثناء عمل البرنامج إلغاء تفعيل أي من المقاطعات الخارجية من خلال التعليمة: `Disable INTx`

ملاحظة2: يمكن أثناء عمل البرنامج إلغاء تفعيل شعاع المقاطعات العام من خلال التعليمة: `Disable Interrupts`

ملاحظة3: التعليمة `Print` تستخدم لطباعة البيانات على النافذة التسلسلية (UART) في حال الوصل مع الحاسب.

التجربة الثانية عشرة: المطلوب تعديل التجربة الثامنة لتعمل المفاتيح اللحظية الثلاث (S1, S2, S3) الموصولة إلى أقطاب المقاطعات الخارجية INT2, INT1, INT0، على تغيير حالة عمل (Toggle) الثنائي D1 عن الضغط على المفتاح S1، وتغيير حالة D2 عند الضغط على S2، وتغيير حالة D3 عند الضغط على S3 – باستخدام المقاطعات الخارجية بدلاً من الفحص الدوري لحالة المفاتيح.



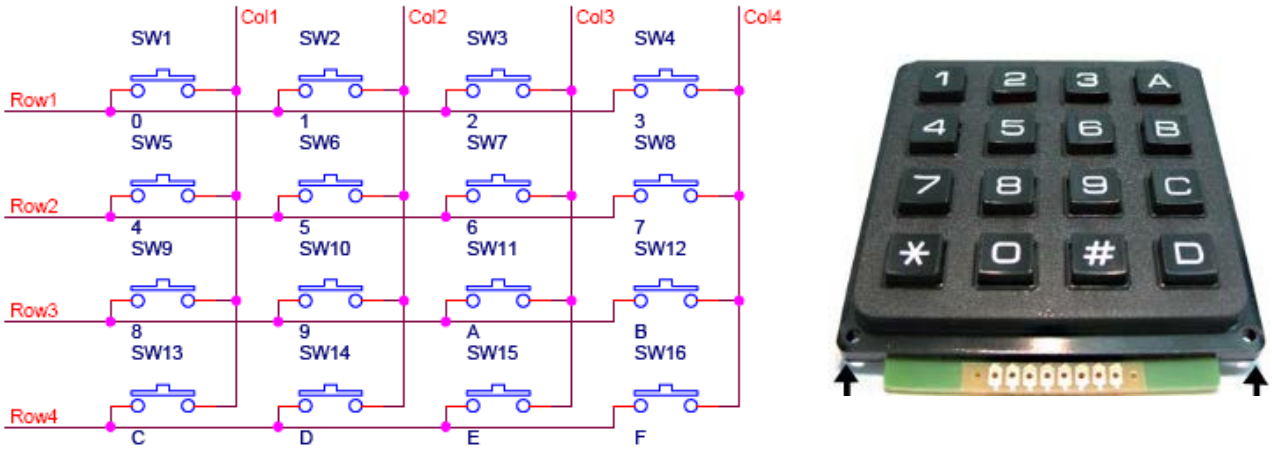

الشكل5-10 توصيل المفاتيح اللحظية الثنائيات مع المتحكم ATmega32A على اللوحة Mini-Phoenix للتجربة 12

البرنامج Exp.12.bas في بيئة BASCOM-AVR:

```
' ********************************************************************************
' * Title         : Exp.12.bas                                                   *
' * Target Board  : Mini-Phoenix - REV 1.00                                      *
' * Author        : Walid Balid                                                  *
' * IDE           : BASCOM AVR 2.0.7.3                                           *
' * Peripherals   : Pull-Up Resistors                                            *
' * Description   : External Interrupts                                          *
' ********************************************************************************
' Set the SW Jumbers to GND (Active Low)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 9600
'-----------------------
'-----------------------[GPIO Configurations]
Config Portc = &B00000111
Led1 Alias Portc.0 : Led2 Alias Portc.1 : Led3 Alias Portc.2
Set Led1 : Set Led2 : Set Led3
'-----------------------
'-----------------------[External Interrupts Configurations]
Config Int0 = Falling : On Int0 Sw_r1 : Enable Int0 : Portd.2 = 1 'PU Resistor
Config Int1 = Falling : On Int1 Sw_r2 : Enable Int1 : Portd.3 = 1
Config Int2 = Falling : On Int2 Sw_r3 : Enable Int2 : Portb.2 = 1

Enable Interrupts
'-----------------------[Variables]
Dim Count1 As Byte , Count2 As Byte , Count3 As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Print "Hello!"
Do

Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Print]
Sw_r1:
   Toggle Led1 : Count1 = Count1 + 1
   Print "Sw1 has Pressed! > " ; Count1
Return
'---<
Sw_r2:
   Toggle Led2 : Count2 = Count2 + 1
   Print "Sw2 has Pressed! > " ; Count2
Return
'---<
Sw_r3:
   Toggle Led3 : Count3 = Count3 + 1
   Print "Sw3 has Pressed! > " ; Count3
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

## 11-5 توصيل وبرمجة لوحة مفاتيح مصفوفية مع متحكم AVR (Interfacing AVR MCU with Matrix-Keypad):

من أجل ربط عدد كبير من المفاتيح اللحظية مع متحكم مصغر فإنه ليس من المجدي ربط كل مفتاح إلى قطب كما مر معنا في التجارب السابقة لأن عدد الأقطاب المستهلكة من المتحكم ستساوي عدد المفاتيح التي تم ربطها مع تلك الأقطاب. لذلك يتم ربط المفاتيح مع بعضها بطريقة مصفوفية – أي يتم توصيل النقطة الأولى للمفاتيح المتوضعة على سطر واحد مع بعضها لتشكل قطب واحد يمثل السطر، كذلك يتم توصيل النقطة الثانية للمفاتيح المتوضعة على عمود واحد مع بعضها لتشكل قطب واحد يمثل العمود... وهكذا كما هو مبين على الشكل5-11.

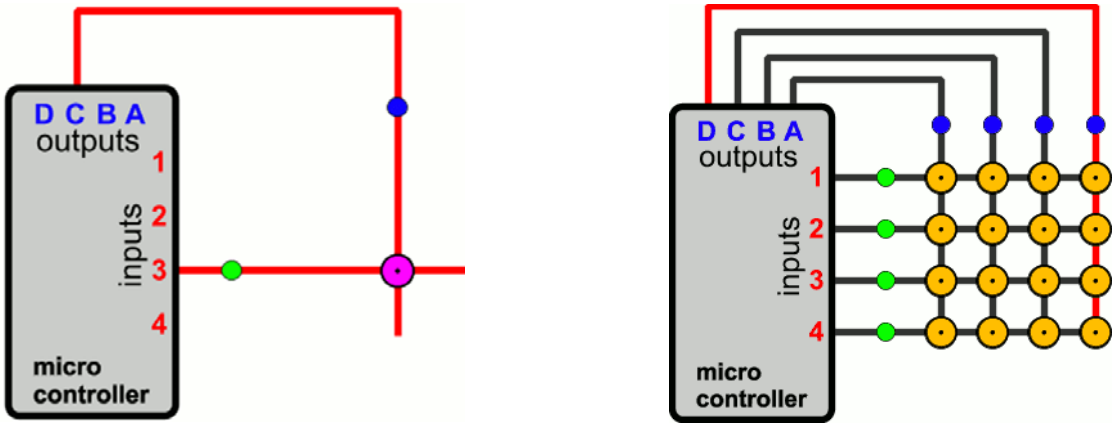



الشكل5-11 لوحة المفاتيح المصفوفية المؤلفة من 16-key

تستخدم لوحة المفاتيح المصفوفية بشكل أساسي في الهواتف، ويمكن أن تكون مؤلفة من 9 مفاتيح (3×3) أو 16 مفتاح (4×4) أو أكثر... ويكون دائماً عدد أقطاب توصيل اللوحة مساوياً إلى مجموع الأسطر والأعمدة (8-Lines < 4×4).

يتم توصيل لوحة المفاتيح مع أقطاب المتحكم المصغر مباشرة، ومنهجية مسح اللوحة لمعرفة المفتاح المضغوط تتم على الشكل التالي:

1) يجب وصل العمود الأول إلى القطب الأول من البوابة (مثلاً: PINB.0) والعمود الثاني إلى القطب الثاني وهكذا... ثم يتم توصيل السطر الأول إلى القطب التالي من نفس البوابة... ففي حال لوحة مفاتيح 4×4 فإن التوصيل سيكون كما هو مبين على الشكل5-13.
2) يتم تعريف أقطاب المتحكم الموصولة مع الأعمدة كأقطاب خرج، ويتم تعريف الأقطاب الموصولة مع الأسطر كأقطاب دخل.
3) يبدأ المسح بكتابة القيمة "1" على العمود الأول (على اعتبار أن الأقطاب الموصلة مع الأعمدة هي أقطاب خرج) وقراءة القيمة الظاهرة على الأسطر (على اعتبار أن الأقطاب الموصلة مع الأسطر هي أقطاب دخل). في حال لم يكن هناك أي مفتاح مضغوط فإن القيمة على الأسطر ستكون "0000". وفي حال كان هناك مفتاح مضغوط فإن السطر الذي ضغط فيه المفتاح ستظهر عليه القيمة المطبقة على العمود "1" وبالتالي يمكن معرفة المفتاح المضغوط. ثم ينتقل المسح إلى العمود الثاني ويكرر العملية السابقة ثم الثالث فالرابع وهكذا حتى يعود للعمود الأول ضمن دورة مسح لانهائية كما في الشكل5-12.




الشكل5-12 توصيل مجموعة المفاتيح مع المتحكم المصغر وحالة المسح

من أجل قراءة لوحة مفاتيح ست عشرية في البيئة BASCOM-AVR فإننا نحتاج إلى تعليميتين أساسيتين:

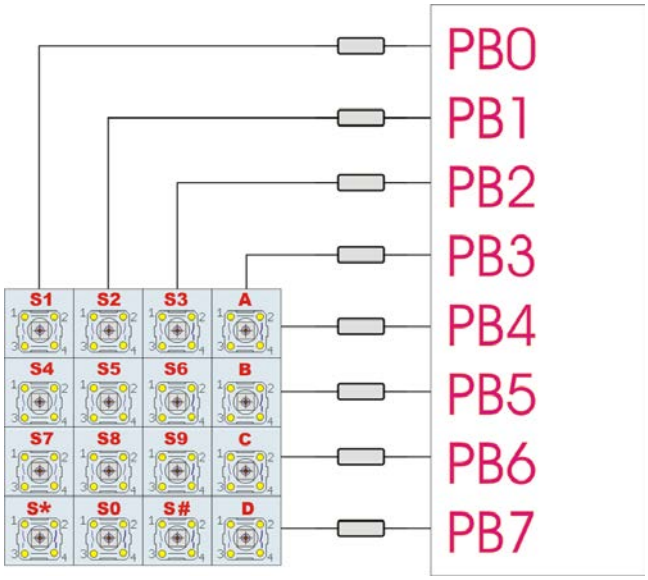
1) تعريف البوابة الموصول معها لوحة المفاتيح وتعريف زمن التأخير (**Debounce**) لتفادي أثر العطالة الميكانيكية للمفاتيح.

```
Config Kbd = Portb , Debounce = 100 , Delay = 100
```

2) قراءة حالة المفاتيح.

```
Var = Getkbd()
```

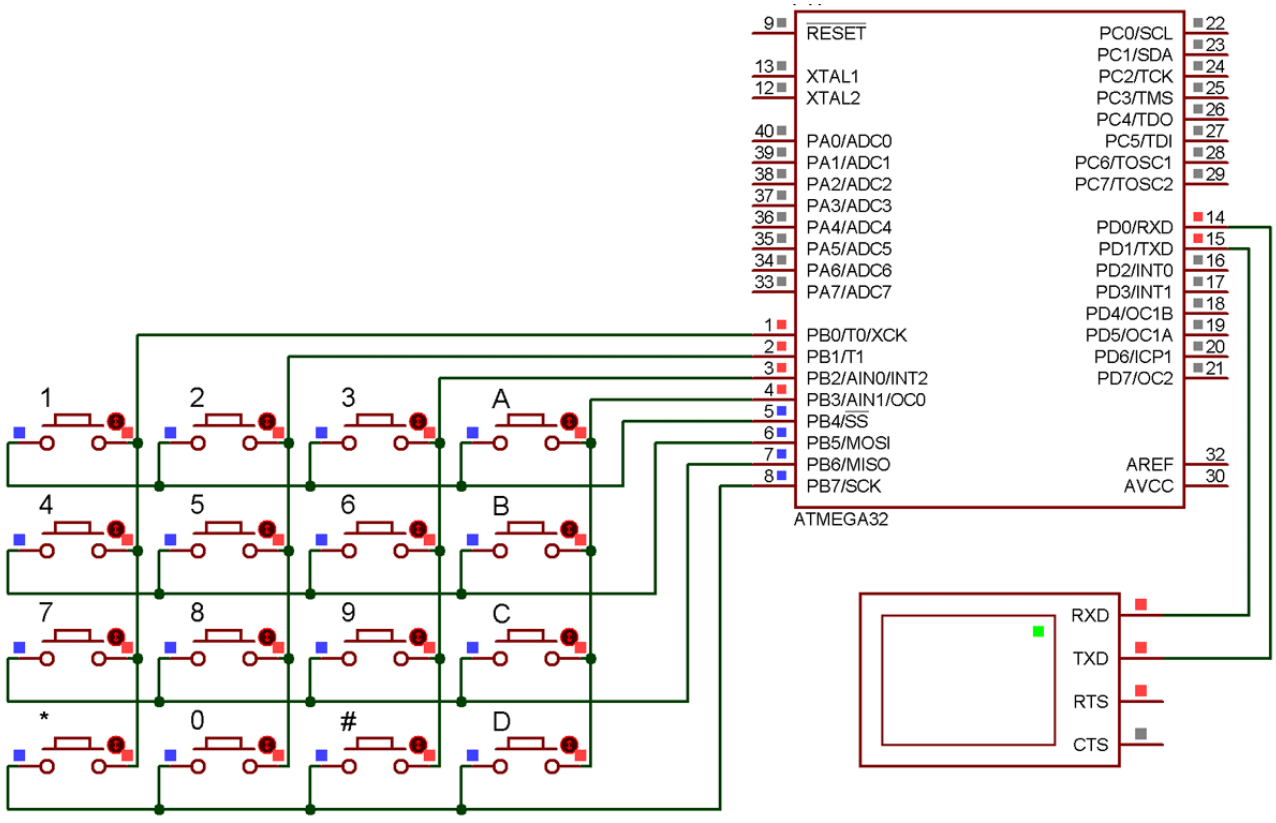
التابع "**Getkbd**" سيعود بقيمة عددية تتراوح بين **0 – 16** تمثل المفتاح المضغوط. حيث يعود هذا التابع بالقيمة **Var = 16** إذا لم يكن هناك أي مفتاح مضغوط. وأما إذا كان هناك مفتاح مضغوط فسيعود بقيمة المفتاح المضغوط كما يلي:




```
Var(S1) = 0
Var(S2) = 1
Var(S3) = 2
Var(A ) = 3
Var(S4) = 4
Var(S5) = 5
Var(S6) = 6
Var(B ) = 7
Var(S7) = 8
Var(S8) = 9
Var(S9) = 10
Var(C ) = 11
Var(S*) = 12
Var(S0) = 13
Var(S#) = 14
Var(D ) = 15
```

الشكل5-13 توصيل مجموعة المفاتيح مع المتحكم المصغر والقيم التي يعود بها التابع "**Getkbd**"

التجربة الثالثة عشرة: المطلوب بكتابة برنامج لقراءة حالة لوحة مفاتيح موصلة بشكل مصفوفي إلى البوابة PORTB لمعرفة المفتاح المضغوط وطباعة اسم المفتاح المضغوط على النافذة التسلسلية UART كما هو مبين على الشكل5-14 – ثم يطلب تطبيقها على اللوحة التعليمية مباشرة.




الشكل5-14 توصيل مجموعة من المفاتيح مع المتحكم ATmega32A لتشكيل لوحة مفاتيح 16-key/Keypad للتجربة 13

البرنامج Exp.13.bas في بيئة BASCOM-AVR:

```
' ************************************************************************
' * Title          : Exp.13.bas                                          *
' * Target Board   : Mini-Phoenix - REV 1.00                             *
' * Target MCU     : ATMega32A                                           *
' * Author         : Walid Balid                                         *
' * IDE            : BASCOM AVR 2.0.7.3                                  *
' * Peripherals    : Keypad                                              *
' * Description    : GPIOs as Input/Keypad                               *
' ************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 9600
'-----------------------
'-----------------------[Keypad Configurations]
Config Kbd = Portb , Debounce = 100 , Delay = 100
'-----------------------
'-----------------------[Variables]
Dim Var As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
'--->[Main Program]
Do
   Var = Getkbd()
   If Var < 16 Then Gosub Check_number
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Print the Key Number]
Check_number:
   Select Case Var
      Case 00 : Print "Key Pressed is (1)"
      Case 01 : Print "Key Pressed is (2)"
      Case 02 : Print "Key Pressed is (3)"
      Case 03 : Print "Key Pressed is (A)"
      Case 04 : Print "Key Pressed is (4)"
      Case 05 : Print "Key Pressed is (5)"
      Case 06 : Print "Key Pressed is (6)"
      Case 07 : Print "Key Pressed is (B)"
      Case 08 : Print "Key Pressed is (7)"
      Case 09 : Print "Key Pressed is (8)"
      Case 10 : Print "Key Pressed is (9)"
      Case 11 : Print "Key Pressed is (C)"
      Case 12 : Print "Key Pressed is (*)"
      Case 13 : Print "Key Pressed is (0)"
      Case 14 : Print "Key Pressed is (#)"
      Case 15 : Print "Key Pressed is (D)"
   End Select
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

. . . ﴿انتهت الجلسة العملية الخامسة﴾ . . .

وليد بليد

- دمتم بخير ومودة ونور -

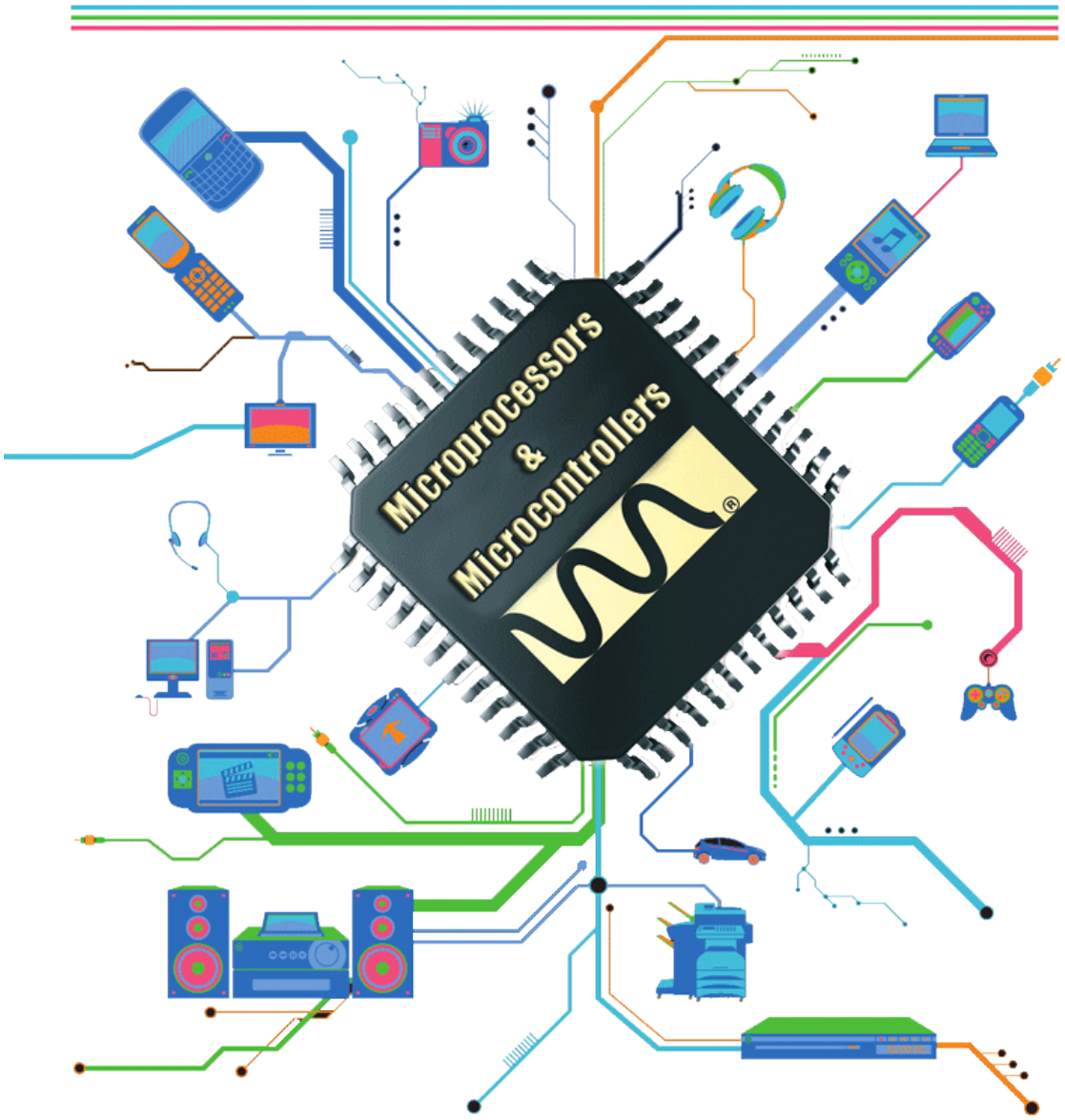
جامعـــــــــــة حلـــــــــــب
كلية الهندسة الكهربائية والإلكترونية
قسـم هندسـة التحكـم والأتمتـة

# ﴿الجلسات العملية لمادة المعالجات والمتحكمات المصغرة﴾

# Microprocessors & Microcontrollers Lab Sessions

السنة الثالثة | قسم اتصالات

## ﴿الجلسة العملية السادسة﴾




م. وليد بليد

Wednesday, April 11, 2012

Copyright © 2012 Walid Balid - All rights reserved.

# الجلسة العملية السادسة

## نظرة عامة (Overview):

هذه المحاضرة تشرح طريقة ربط وبرمجة شاشات الإظهار الكريستالية المحرفية مع متحكمات AVR. ثم تقدم تطبيقاً عملياً لبرمجة شاشة الإظهار المحرفية في البيئة BASCOM-AVR ومحاكاتها في البيئة Proteus في نمطي العمل 4bit, 8bit.

## 1-6 شاشة الإظهار الكريستالية المحرفية (Character Liquid Crystal Display):

إن شاشة الإظهار الكريستالية LCD هي عبارة عن مصفوفة نقطية تستخدم لعرض المعلومات والنتائج، ويمكن من خلالها إظهار جميع رموز الآسكي تقريباً والتي يبلغ عددها 189 رمزاً مختلفاً.

تعتمد شاشة الإظهار LCD على البلورات السائلة (Liquid Crystal)، حيث تم اكتشاف البلورات السائلة أول مرة في عام 1888 من قبل عالم النبات النمساوي فريريك رينتيزير الذي لاحظ أنه عندما يتم صهر الكوليسترول النباتي يصبح غير صافٍ، ومن ثم يأخذ بالصفو عندما ترتفع درجة حرارته. وبالاعتماد على التبريد يبدأ السائل (الكوليسترول) بالتحول إلى اللون الأزرق قبل التبلور الأخير له. في عام 1968 وبعد مرور ثمانين سنة، صنعت شركة RCA شاشة الLCD الأولى.

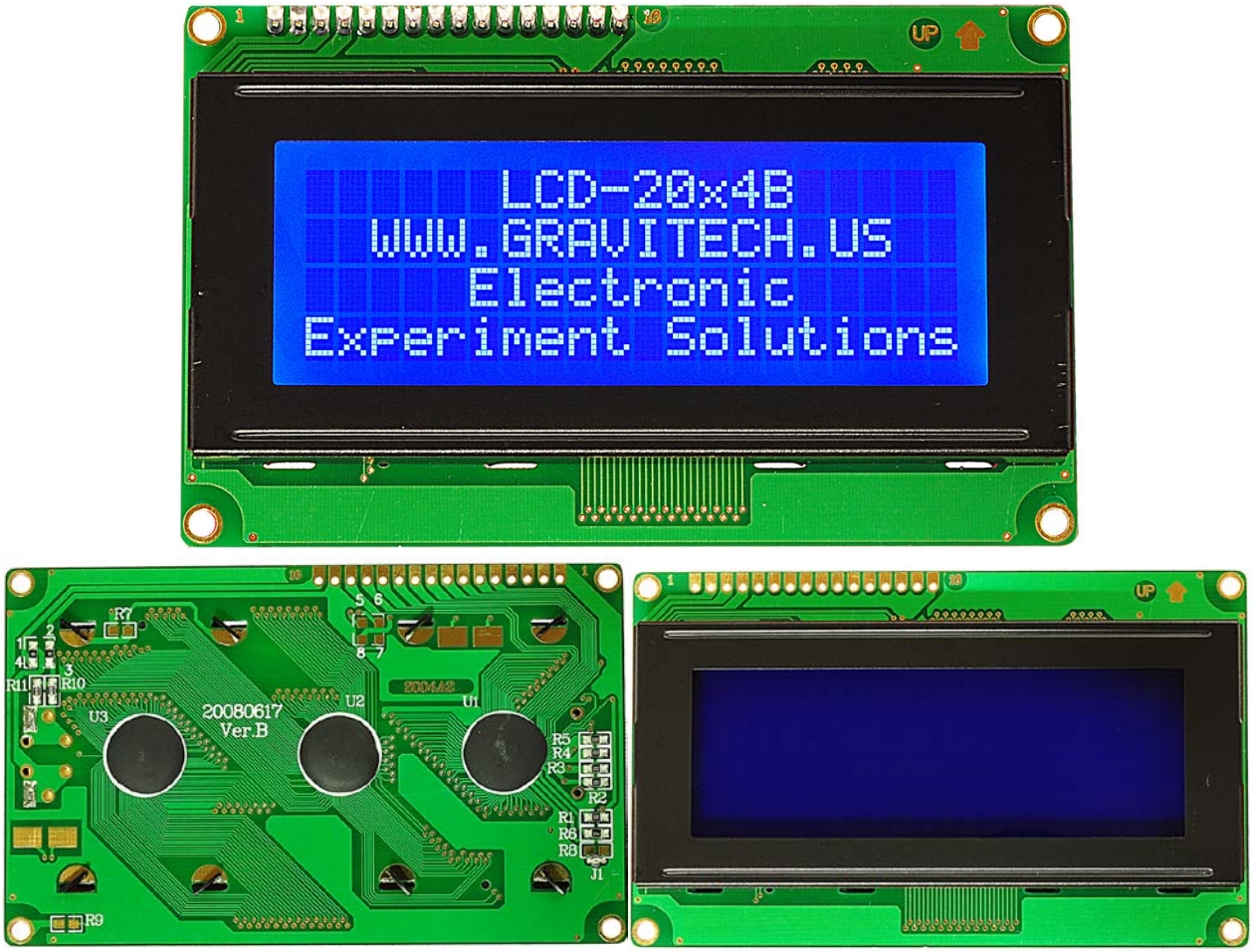
تحتوي شاشة الإظهار LCD على شريحة معالج إظهار خاص مصنع بتقنية CMOS ويحمل في أغلب شرائح شاشات الإظهار الرقم HD44780 المصنع من قبل شركة Hitachi اليابانية، فتوفر بذلك على المستخدم القيام بالعديد من العمليات الشاقة والمعقدة، كما تزود شاشة الإظهار LCD بذاكرة داخلية خاصة تقسم بدورها إلى قسمين: 1) ذاكرة المعطيات DD-RAM، 2) ذاكرة مولد الرموز CG-RAM. تقوم هذه الذواكر بالاحتفاظ بالرموز المراد إظهارها وتمكن المبرمج من إعادة إظهارها بدون الحاجة إلى إرسالها مرة أخرى. إضافة إلى ذلك تحتوي الشاشة LCD على دارات قيادة (Drivers) لخانات شاشة الإظهار.

تأتي شاشة LCD بمقاسات مختلفة من عدد الأسطر والأعمدة (المحارف)، حيث يمكن أن تكون مؤلفة من سطر حتى أربعة أسطر، ويحتوي كل سطر على عدد من الخانات (المحارف) يتراوح من 16 وحتى 40 الخانة؛ والخانة هي عبارة عن مربع صغير يتم فيه إظهار محرف واحد فقط؛ وأكثر الشاشات شيوعاً هي الشاشات ذات القياسات التالية:

Chars × Lines: 16 x 1 | 16 x 2 | 16 x 4 | 20 x 2 | 20 x 4 | 40 x 2 | 40 x 4

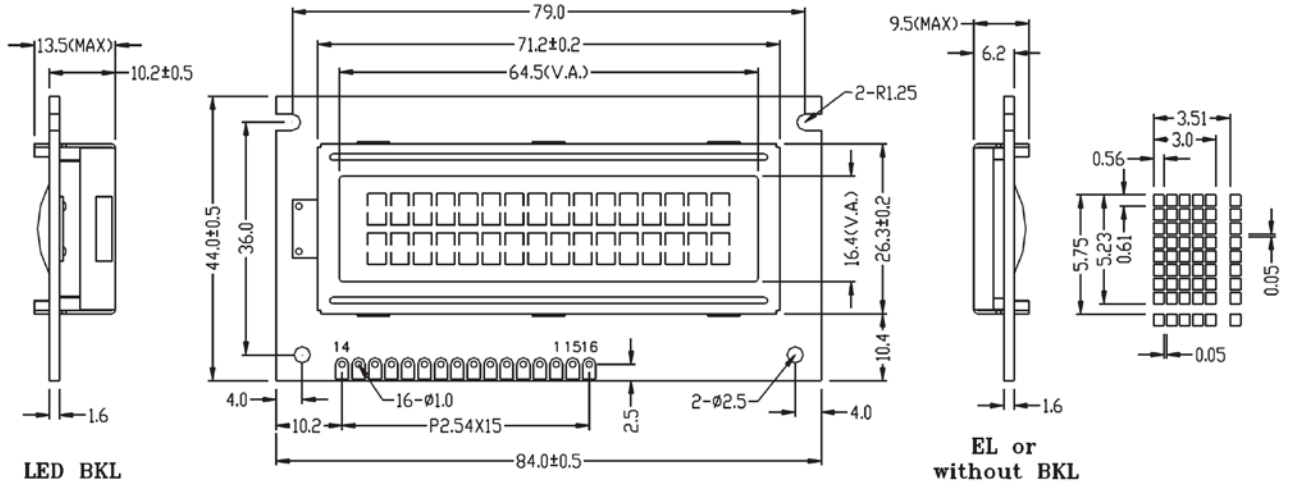
تملك شاشات LCD بشكل عام نفس أقطاب التحكم مع وجود بعض الاختلافات البسيطة. يبين الشكل التالي أقطاب التحكم لشاشة LCD ذات سطرين و 16 عمود وفيما يلي أسماء هذه الأقطاب ووظيفتها.

الشكل6-1 شاشة إظهار كريستالية محرفية ذات قياس 2 x 16




الشكل6-2 شاشة إظهار كريستالية محرفية ذات قياس 4 x 20

الشكل6-3 الرموز التي يمكن إظهارها على شاشة الإظهار المحرفية




| PIN NO | Symbol | Function |
|---|---|---|
| 1 | VSS | GND |
| 2 | VDD | +5V |
| 3 | V0 | Contrast adjustment |
| 4 | RS | H/L Register select signal |
| 5 | R/W | H/L Read/Write signal |
| 6 | E | H/L Enable signal |
| 7 | DB0 | H/L Data bus line |
| 8 | DB1 | H/L Data bus line |
| 9 | DB2 | H/L Data bus line |
| 10 | DB3 | H/L Data bus line |
| 11 | DB4 | H/L Data bus line |
| 12 | DB5 | H/L Data bus line |
| 13 | DB6 | H/L Data bus line |
| 14 | DB7 | H/L Data bus line |
| 15 | A | +4.2V for BKL |
| 16 | K | Power supply for BKL(0V) |

الشكل6-4 المحيط الخارجي لشاشة الإظهار المحرفية وتوزع الأقطاب ووظائفها

- القطب Vss: قطب التغذية السالب للشاشة GND.
- القطب Vdd : قطب التغذية الموجب للشاشة +5V.
- القطب Vo : قطب جهد التباين، ويقصد بالتباين حدة ظهور الرمز على الشاشة. عند أقل قيمة تباين لا يمكن أن تظهر الرموز على الشاشة ويكون هذا عند تطبيق (+5v) على هذا القطب. أعلى تباين للشاشة يكون عند تطبيق (GND) على هذا القطب ويمكن التحكم بتباين الشاشة عن طريق وصل قطب التباين (V0) إلى مقاومة متغيرة 10K.
- القطب RS: قطب مسجل اختيار الدخل للشاشة؛ من أجل إرسال أمر تحكم، يتم وضع أمر التحكم على أقطاب D0-D7 ويتم تطبيق "0" منطقي على هذا القطب؛ ومن أجل إرسال معطيات إلى الشاشة فيتم وضع المعطيات على أقطاب D0-D7 ويتم تطبيق "1" منطقي على هذا القطب.

- القطب R/W: ويتم تطبيق "1" منطقي على هذا القطب للقراءة (R) من ذاكرة الشاشة، ويتم تطبيق "0" منطقي على هذا القطب للكتابة (W) إلى الشاشة.
- القطب E: إن تأكيد عملية إرسال أمر تحكم أو معطيات إلى الشاشة يتم من خلال نبضة تمكين عند الجبهة الهابطة على القطب E.
- الأقطاب DB0 ~ DB7: هي أقطاب المعطيات (DATA)، حيث يتم كتابة المعطيات أو قراءتها أو كتابة كلمات التحكم إلى شاشة LCD عبر هذه الخطوط.
- القطبين A & K: تملك بعض الشاشات إضاءة خلفية (Backlight) وظيفتها تأمين الإضاءة الكافية للشاشة ليتمكن المستخدم من رؤية العبارات المكتوبة عليها في الليل؛ يتم تشغيل الإضاءة بتطبيق "+5V" على A و"GND" على K.

## 2-6 أنماط عمل شاشات الإظهار المحرفية

تملك شاشة الإظهار الكريستالية نمطي عمل:

❖ نمط العمل 4-bit: وفيه يتم استخدام أربعة خطوط من خطوط المعطيات (DB0 ~ DB7) وهي DB4 ~ DB7 ويتم تجاهل (عدم توصيل) باقي خطوط المعطيات (DB0 ~ DB3). وفي هذه الحالة يتم البيانات عبر هذه الخطوط على دفعتين – أي يتم إرسال النصف الأدنى من البايت ثم النصف الأعلى من البايت. ويستخدم هذا النمط بهدف توفير في عدد أقطاب المتحكم المطلوبة وبالتالي سنحتاج إلى ستة أقطاب فقط من المتحكم للتوصيل: DB4, DB5, DB6, DB7, E, RS.

❖ نمط العمل 8-bit: وفيه يتم توصيل جميع خطوط المعطيات مع المتحكم وبالتالي سنحتاج إلى عشرة أقطاب من المتحكم لتوصيل الشاشة: DB0, DB1, DB2, DB3, DB4, DB5, DB6, DB7, E, RS.

## 3-6 برمجة شاشة الإظهار المحرفية في Bascom-AVR (Programming LCD in BASCOM-AVR):

تعليمات التعامل مع شاشة الإظهار الكريستالية LCD في البيئة Bascom-AVR على قسمين:

1) تعليمات التهيئة (Configuration).
2) تعليمات الإظهار (Display).

تعليمات التهيئة (Configuration) تتضمن:

- تحديد أبعاد الشاشة؟

```
Config Lcd = 16 * 2
```

- تحديد نمط العمل والأقطاب الموصولة مع الشاشة

1) **نمط العمل 4-bit**: Db4 ~ Db7 تمثل الأقطاب الموصولة مع خطوط المعطيات للشاشة، E, Rs خطوط التحكم.

```
Config Lcdpin = Pin , Db4 = Portc.2 , Db5 = Portc.3 , Db6 = Portc.4 ,
Db7 = Portc.5 , E = Portd.3 , Rs = Portd.4
```

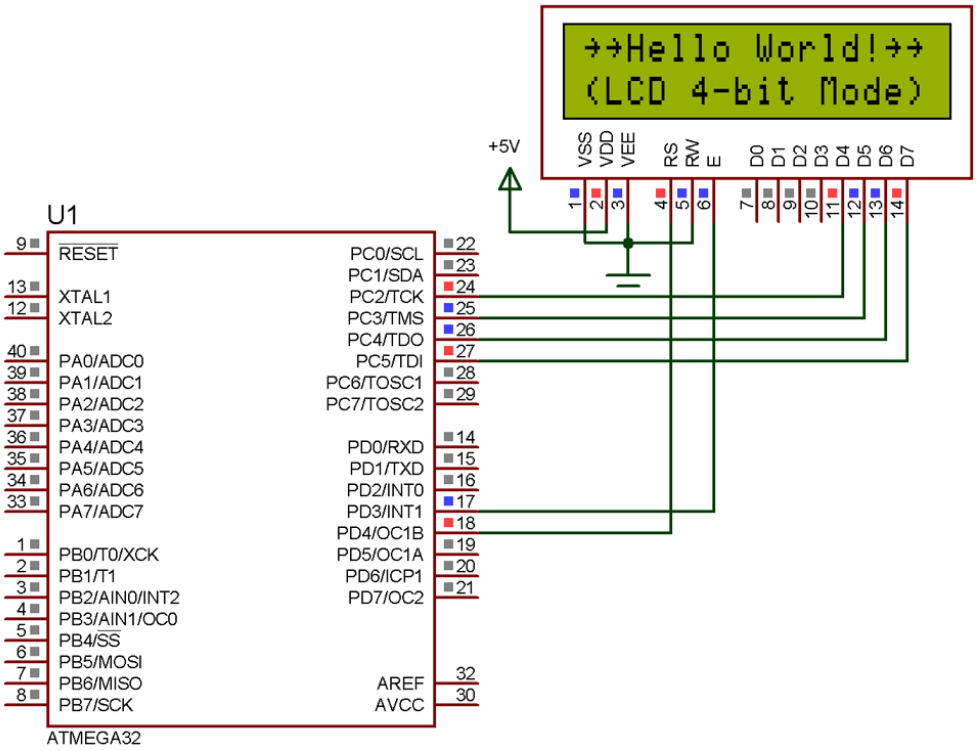
2) **نمط العمل 8-bit**: Port تمثل أقطاب البوابة الموصولة مع خطوط المعطيات للشاشة، E, Rs خطوط التحكم.

```
Config Lcdpin = Pin , Port = Portc , E = Portd.3 , Rs = Portd.4
```

تعليمات الإظهار (Display) تتضمن مجموعة من التعليمات:

| | |
|---|---|
| **Lcd** var | ▪ عرض متحول |
| **Lcd** "Hello World" | ▪ عرض عبارة نصية |
| **Display Off** | ▪ إطفاء الشاشة |
| **Display On** | ▪ تشغيل الشاشة |
| **Shiftlcd Right** | ▪ إزاحة المحتوى إلى اليمين خانة |
| **Shiftlcd Left** | ▪ إزاحة المحتوى إلى اليسار خانة |
| **Cursor On** [**Blink**] | ▪ تفعيل مؤشر الكتابة – خفقان |
| **Cursor Off** | ▪ إخفاء مؤشر الكتابة |
| **Shiftcursor Right** | ▪ إزاحة مؤشر الكتابة خانة إلى اليمين |
| **Shiftcursor Left** | ▪ إزاحة مؤشر الكتابة خانة إلى اليسار |
| **Locate** X , Y | ▪ وضع مؤشر الكتابة عند نقطة محددة (سطر/عمود) |
| **Home Upper** | ▪ الانتقال إلى السطر/العمود الأول (نقطة البداية) |
| **Lowerline** | ▪ تحريك مؤشر الكتابة إلى السطر التالي |
| **Thirdline** | ▪ الانتقال إلى السطر الثالث |
| **Fourthline** | ▪ الانتقال إلى السطر الرابع |
| **Home Third** | ▪ وضع مؤشر الكتابة في بداية السطر الثالث |
| **Home Fourth** | ▪ وضع مؤشر الكتابة في بداية السطر الرابع |
| **Deflcdchar** 0 , 14 , 17 ,... | ▪ تعريف محرف إضافي باستخدام الأداة LCD Designer |
| **Lcd Chr**(x) | ▪ إظهار المحرف الإضافي على الشاشة |

التجربة الرابعة عشرة: المطلوب كتابة برنامج لتشغيل شاشة إظهار محرفية كريستالية في النمط 4-bit موصولة إلى متحكم مصغر ATmega32A وفقاً لمخطط التوصيل للوحة التعليمية – ملاحظة: هذا البرنامج يمكن تشغيله مباشرة على اللوحة التعليمية.



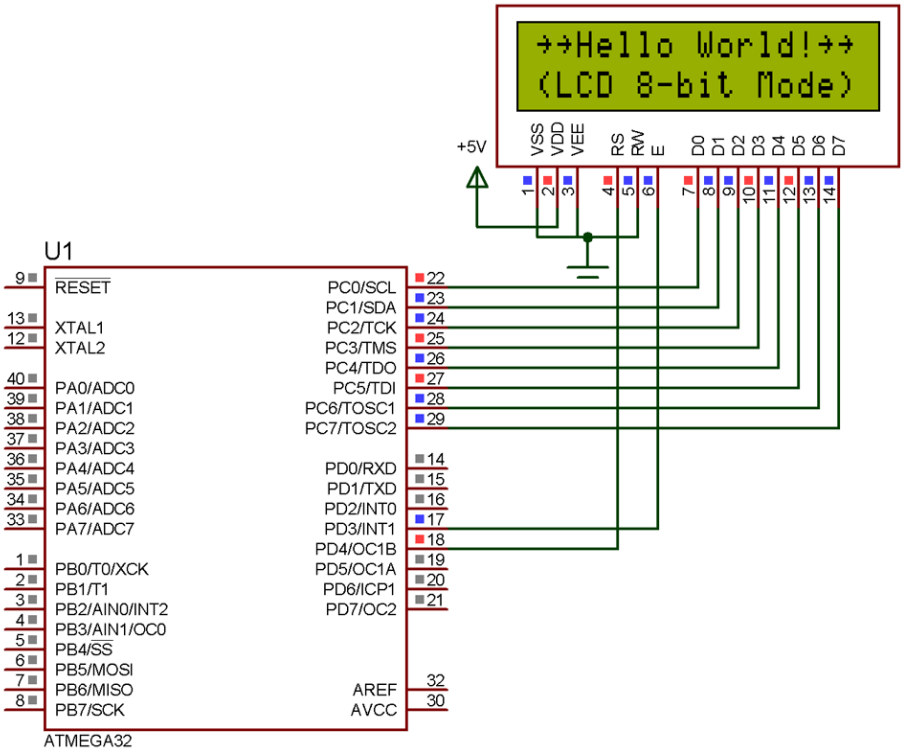

الشكل6-5 توصيل شاشة LCD مع المتحكم ATmega32 للتجربة 14

البرنامج Exp.14.bas في بيئة BASCOM-AVR:

```
' ****************************************************************************
' * Title          : Exp.14.bas                                              *
' * Target Board   : Mini-Phoenix - REV 1.00                                 *
' * Target MCU     : ATMega32A                                               *
' * Author         : Walid Balid                                             *
' * IDE            : BASCOM AVR 2.0.7.3                                      *
' * Peripherals    : 16 x 2 LCD                                              *
' * Description    : 4 bit LCD Mode                                          *
' ****************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'-----------------------
'-----------------------[LCD Configurations]
Config Lcdpin = Pin , Db4 = Portc.2 , Db5 = Portc.3 , Db6 = Portc.4 , Db7 =
Portc.5 , E = Portd.3 , Rs = Portd.4
Config Lcd = 16 * 2
'-----------------------
'-----------------------[Variables]
Dim I As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   Cls
```

```
   Upperline : Lcd "~~Hello World!~~" : Wait 1
   Lowerline : Lcd "(LCD 4-bit Mode)" : Wait 1

   Gosub Shift2right : Gosub Shift2left

   Locate 1 , 8 : Lcd ":" : Wait 1
   Locate 2 , 1 : Lcd ">" : Wait 1

   Shiftcursor Right : Wait 1 : Shiftcursor Left

   Cursor Off Noblink : Wait 1 : Cursor On Blink

   Display Off : Wait 1 : Display On

   Home Upper : Wait 1 : Cls

   Deflcdchar 0 , 32 , 32 , 10 , 21 , 17 , 10 , 4 , 32
   Deflcdchar 1 , 4 , 10 , 17 , 10 , 10 , 17 , 10 , 4
   Locate 1 , 9 : Lcd Chr(0) : Wait 1
   Locate 2 , 9 : Lcd Chr(1) : Wait 1
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Shift LCD Char to Right]
Shift2right:
   For I = 1 To 8
      Shiftlcd Right : Waitms 500
   Next I
Return
'-----------------------
'--->[Shift LCD Char to Left]
Shift2left:
   For I = 1 To 8
      Shiftlcd Left : Waitms 500
   Next I
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

التجربة الخامسة عشرة: المطلوب كتابة برنامج لتشغيل شاشة إظهار محرفية كريستالية في النمط 4-bit موصولة إلى متحكم مصغر ATmega32A وفقاً لمخطط التوصيل للوحة التعليمية – ملاحظة: هذا البرنامج لا يمكن تشغيله على اللوحة التعليمية.




الشكل6-6 توصيل شاشة LCD مع المتحكم ATmega32 للتجربة 15

البرنامج Exp.15.bas في بيئة BASCOM-AVR:

```
' ********************************************************************************
' * Title          : Exp.15.bas                                                  *
' * Target Board   : Phoenix - REV 1.00                                          *
' * Target MCU     : ATMega32A                                                   *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : 16 x 2 LCD                                                  *
' * Description    : 8 bit LCD Mode                                              *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
'-----------------------
'-----------------------[LCD Configurations]
Config Lcdpin = Pin , Port = Portc , E = Portd.3 , Rs = Portd.4
Config Lcd = 16 * 2
'-----------------------
'-----------------------[Variables]
Dim I As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   Cls
   Upperline : Lcd "~~Hello World!~~" : Wait 1
```

```
   Lowerline : Lcd "(LCD 8-bit Mode)" : Wait 1

   Gosub Shift2right : Gosub Shift2left

   Locate 1 , 8 : Lcd ":" : Wait 1
   Locate 2 , 1 : Lcd ">" : Wait 1

   Shiftcursor Right : Wait 1 : Shiftcursor Left

   Cursor Off Noblink : Wait 1 : Cursor On Blink

   Display Off : Wait 1 : Display On

   Home Upper : Wait 1 : Cls

   Deflcdchar 0 , 32 , 32 , 10 , 21 , 17 , 10 , 4 , 32
   Deflcdchar 1 , 4 , 10 , 17 , 10 , 10 , 17 , 10 , 4
   Locate 1 , 9 : Lcd Chr(0) : Wait 1
   Locate 2 , 9 : Lcd Chr(1) : Wait 1
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Shift LCD Char to Right]
Shift2right:
   For I = 1 To 8
      Shiftlcd Right : Waitms 500
   Next I
Return
'------------------------
'--->[Shift LCD Char to Left]
Shift2left:
   For I = 1 To 8
      Shiftlcd Left : Waitms 500
   Next I
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

. . . ﴿انتهت الجلسة العملية السادسة﴾ . . .

وليد بليد

- دمتم بخير ومودة ونور -

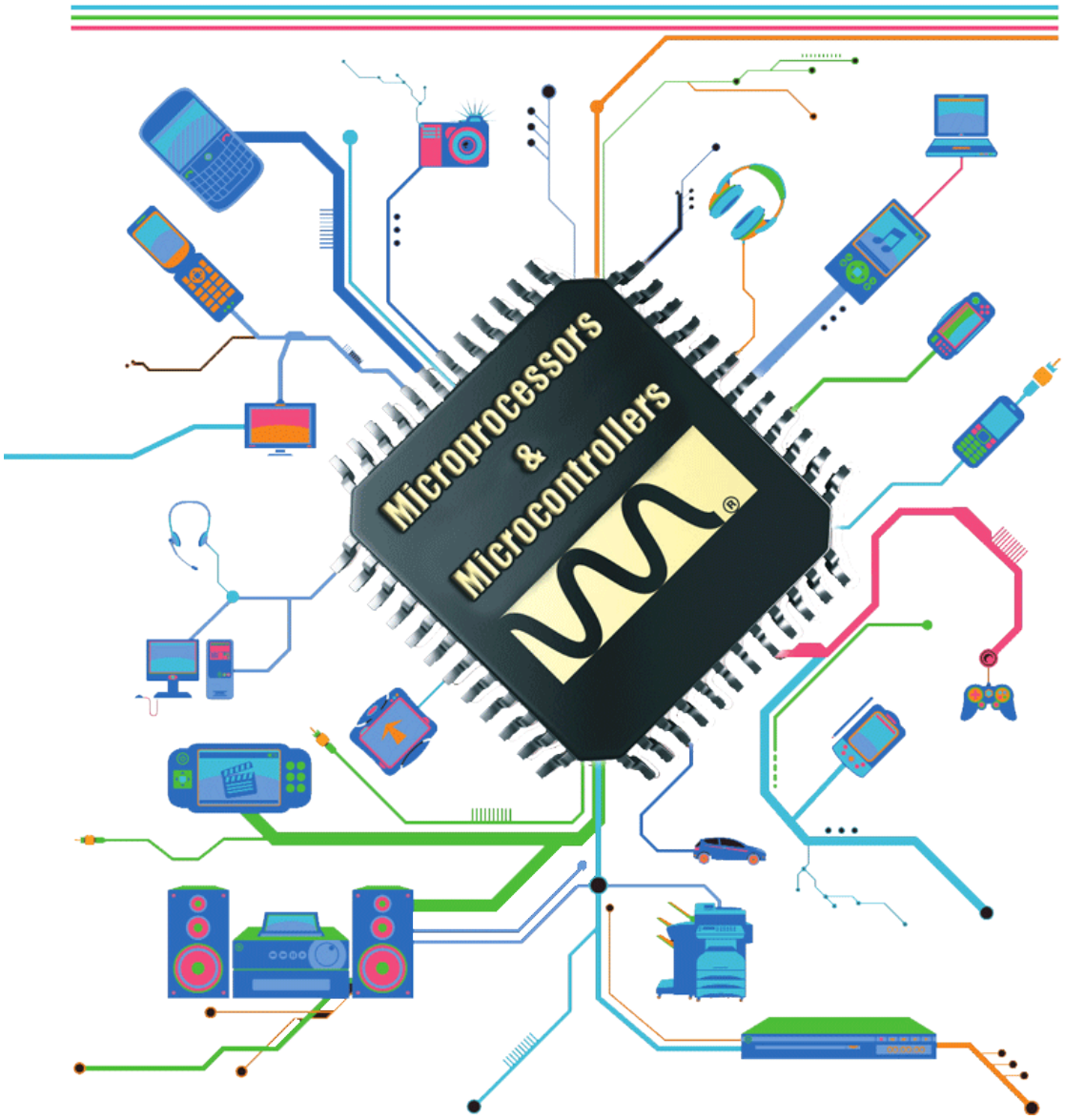
جامعـــــــــــة حلـــــــــــب
كلية الهندسة الكهربائية والإلكترونية
قسـم هندسـة التحكـم والأتمتـة

# ﴿ الجلسات العملية لمادة المعالجات والمتحكمات المصغرة ﴾

# Microprocessors & Microcontrollers Lab Sessions

## السنة الثالثة | قسم اتصالات

## ﴿ الجلسة العملية السابعة ﴾




م. وليد بليد

Copyright © 2012 Walid Balid - All rights reserved.

Wednesday, April 18, 2012

# الجلسة العملية السابعة

نظرة عامة (Overview):

هذه المحاضرة تشرح بنية البروتوكول RC5 المستخدم في أجهزة التحكم بالأشعة تحت الحمراء. ثم تقدم تطبيقاً عملياً لربط مستقبل أشعة تحت الحمراء يدعم البروتوكول المذكور وطرقة قراءة البيانات من المستقبل. ثم طريقة تصميم وبرمجة جهاز تحكم لإرسال أوامر تحكم باستخدام الأشعة تحت الحمراء والبروتوكول RC5.

## 1-7 البروتوكول RC5 وأجهزة التحكم بالأشعة تحت الحمراء (RC5 Code & The IR Remote Controls):

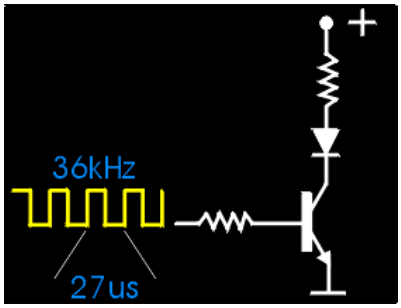
التساؤل الأول الذي يتبادر للذهن هو تساؤل عن ماهية الأشعة تحت الحمراء؟ فيأتي التعريف بأنها عبارة عن طاقة إشعاع ضوئي غير مرئي يقع تحت حزمة الترددات المرئية لأعيننا. في الحقيقة إن الأشعة تحت الحمراء هي ضوء طبيعي يبلغ طول الموجه لهذه الأشعة 950nm وهي موجة قصيرة جداً لهذا لا يمكن للعين أن ترى الضوء المنبعث من مرسل الأشعة تحت الحمراء.

تعتبر الأشعة تحت الحمراء من أرخص الطرق وأسهلها للتحكم عن بعد بالأجهزة وذلك ضمن مجال مرئي، وتستخدم بكثرة في الأجهزة الكهربائية المنزلية وأجهزة التسجيل الرقمي والعرض المرئي. بالإضافة إلى سهولة توليدها، كما أنها لا تعاني من التدخل الكهرومغناطيسي، ولكنها في نفس الوقت يمكن أن تتصادم مع إشعاعات تحت حمراء أخرى كأشعة الشمس مثلاً تحوي على مجال طيف عريض من الإشعاعات التي منها الأشعة تحت الحمراء، وهذا سيؤثر بدوره على فعالية الإرسال.

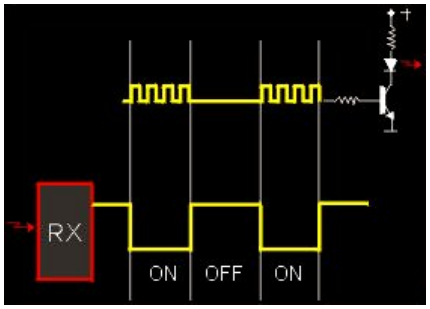
إن كثير من الأشياء يمكن أن تولد الأشعة تحت الحمراء، وخوصاً الأجسام التي تصدر حرارة كأجسادنا مثلاً: المصابيح، الأفران، الماء الحار، لذلك يجب استخدام مفتاح أو عنوان للجهاز المرسل لتفادي الأشعة المزيفة الصادرة عن الأجسام التي لها إصدار حراري وليخبر المستقبل عن البيانات الحقيقية التي يجب أن يستجيب لها نظام التحكم، وهذا ما سوف نوضحه لاحقاً ويعبر عنه بـ العنوان (Address).

إن حزمة ترددات الأشعة تحت الحمراء تتراوح بين 30KHZ – 60KHZ ومجال الأشعة الأفضل هو ضمن 36KHZ والحزم التي حوله (38KHZ). لذلك تستخدم أجهزة التحكم بالأشعة تحت الحمراء الحزمتين 38KHZ, 36KHZ لإرسال المعلومات وهذا يعني أن الثنائي المرسل للأشعة تحت الحمراء سوف يتذبذب 36~38 ألف مرة خلال دور قدره واحد ثانية من أجل القيمة واحد منطقي، وسيكون ساكن من أجل قيمة صفر منطقي.

إن مسألة إرسال تردد 36KHZ, 38KHZ هي مسألة سهلة، لكن الصعوبة تكمن في استقبال هذه الترددات وخصوصاً أن هذه الترددات انتقلت عبر الهواء وتراكبت معها ترددات الضجيج المحيط، لهذا السبب تقوم بعض الشركات بإنتاج مستقبلات الأشعة تحت الحمراء التي تحوي في بنيتها على مرشحات الحزمة ودارات فك التشفير ودارات القص للحزم الغير مرغوبة، وهذا بدوره يساعد على استخلاص الإشارة الحقيقية. الشكل التالي يبين دارة إرسال بسيطة من أجل إرسال تردد 36KHZ، وذلك بتطبيق إشارة مربعة 27uS على قاعدة الترانزستور الشكل1. إن المستقبل سيقوم باستلام الإشارة المرسلة وتعديلها كما في الشكل2.



الشكل1



الشكل2

نلاحظ أن دارة التعديل الموجودة داخل المستقبل قد عكست المستوى المنطقي للإشارة.

## 2-7 ماهي معايير التحكم باستخدام الأشعة تحت الحمراء:

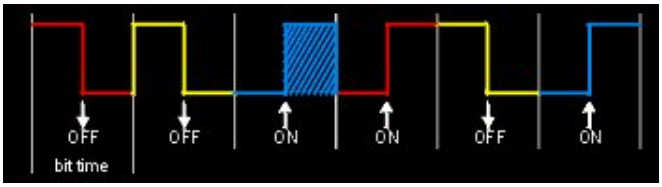
هناك الكثير من معايير التحكم (بروتوكولات) التي تعمل عليها المستقبلات، منها: NEC، SIRCS، RC5، JAPAN، SAMSUNG، Sony. وتختلف هذه البروتوكولات عن بعضها في شكل موجة الإرسال وبنيتها (Waveforms).

## 3-7 المعيار RC5:

إن اهتمامنا ينصب بشكل كلي على معيار RC5 الذي طورته شركة فيلبس ويتلخص بإرسال قطار من 14 نبضة في كل مرة يتم فيها الضغط على أحد أزرار جهاز التحكم وبزمن 1.728mS عند التردد 36KHz أو بزمن 1.4mS عند التردد 38KHz لكل نبضة، وهذا القطار من النبضات يتكرر كل 130mS إذا أبقيت المفتاح مضغوطاً. ولفهم مبدأ عمل هذا البروتوكول يجب التعرف إلى البارامترات التالية:

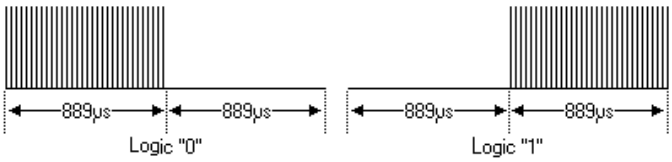
- طول العنوان (Address Length).
- طول أمر التحكم (Command Length).
- تردد الناقل ( Carrier Frequency).
- زمن نبضة بداية الإرسال (Start Bit).
- زمن نبضة الإرسال للمستوى المنطقي "1" (High-Bit-Time).
- زمن نبضة الإرسال للمستوى المنطقي "0" (Low-Bit-Time).

إن هذه البارامترات تختلف حسب نوع المستقبل. إن كل نبضة من قطار النبضات هي بت واحد منقسم إلى قسمين: له نصف يميني ونصف يساري، ولكل منهما مستوى منطقي معاكس للآخر دائماً. فإذا كان البت المرسل من طرف الإرسال هو واحد منطقي، فإن القسم اليميني من البت سيكون واحد منطقي، بينما القسم اليساري سيكون صفر منطقي، وإذا كان البت المرسل هو صفر منطقي، فستكون عكس الحالة السابقة تماماً. بمعنى آخر، يمكنك أن تستنتج أن القسم اليميني من البت المستقبل، سيكون له نفس المستوى المنطقي للبت المرسل، من الشكل السابق تجد النبضة الزرقاء لها مستوى واحد منطقي، وهذا يعني أن البت المرسل هو واحد منطقي أيضا، ولكن القسم اليساري سيكون عكسه.



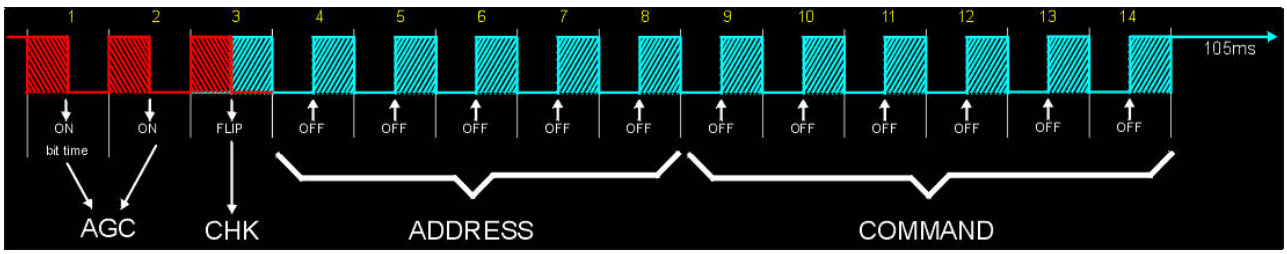

الشكل3 يبين المنطق الحقيقي الذي سوف تستقبله

في هذا البروتوكول هناك عدد محدد من النبضات التي دور كل منها 27µs (عند التردد 36KHz) أو 18.75µS (عند التردد 38KHz) يجب أن تصل إلى دارة فاك التشفير الموجودة داخل المستقبل (demodulator) ليفهم أن التردد المستقبل هو التردد الصحيح ومن ثم نقله إلى الخرج، هذا العدد من النبضات لمستقبلات شركة فيلبس هو 32 نبضة لكل قسم من كل بت من بتات الإرسال، وبالتالي 64 نبضة لكل بت. وعليه فإنه من أجل إرسال "0" فإنه سيكون لدينا في طرف المستقبل في مرحلة فك التعديل 32 نبضة مربعة دور كل منها 27µs (عند التردد 36KHz) أو 18.75µS (عند التردد 38KHz) ثم يليها 32 silence pulse. بينما من أجل إرسال "1" سيكون لدينا الحالة المعاكسة تماماً، 32 silence pulse ثم يليها 32 نبضة مربعة دور كل منها كل منها 27µs (عند التردد 36KHz) أو 18.75µS (عند التردد 38KHz).




الشكل4 المنطق "1" والمنطق "0" في الإرسال عند التردد 36KHz

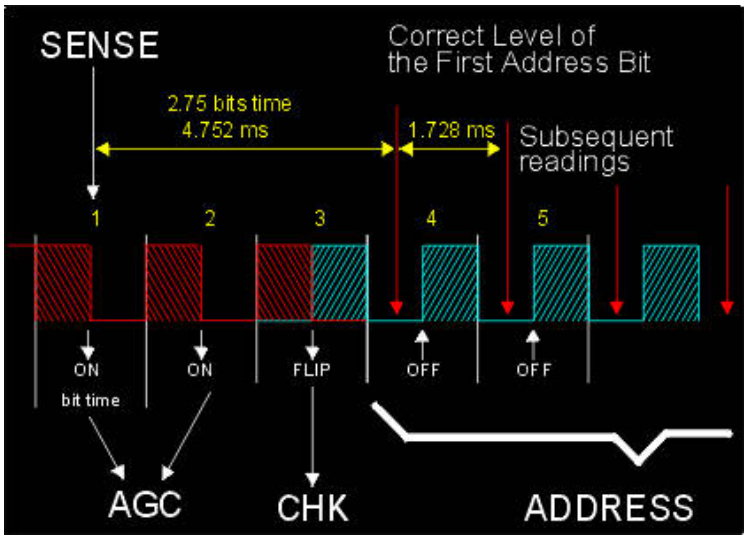
يتكون بروتوكول RC5 من 14Bits ثنائي (أي له نصفين) كما هو مبين على الشكل5.




الشكل5 بروتوكول الإرسال RC5

| Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 | Bit8 | Bit9 | Bit10 | Bit11 | Bit12 | Bit13 | Bit14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| start bits | | control | Address | | | | | Command | | | | | |

- Bits1-2: هي بتات بداية الإرسال (Start Bits or AGC "Automatic Gain Control") وهي دائماً تملك القيمة "1". وهي تساعد هذه البتات في معايرة التحكم الآلي بربح مستقبل الأشعة وكذلك لإعلام المستقبل ببدء عملية الإرسال.




الشكل6 بتات التحكم في بروتوكول الإرسال RC5

- Bit3: هو بت التحكم CHECK bit (Control Bit or Toggle Bit)، هذا البت تتغير قيمته بين الصفر والواحد منطقي في كل مرة يتم فيها ضغط أحد أزرار التحكم. هذا يفيد جهاز التحكم ليفهم إذا مازلت تضغط على أحد الأزرار ويتكرر الأمر – تصور انك تضغط الرقم واحد وتستمر بالضغط، فلولا هذا البت فإن الجهاز سيفهم انك تريد اختيار القناة 11 بدلاً من القناة واحد لأنه سيرسل قطارين من النبضات لهما القيمة نفسها.
- Bits4-8: هي بتات العنوان، هذه البتات الخمسة تسمح لي باختيار نوع الجهاز الذي يجب أن يستجيب للأوامر، وهي تحقق لي عنونة لـ 32 جهاز (2^5=32) وهي على الشكل التالي:

| SYSTEM ADDRESS | EQUIPMENT |
|---|---|
| 0 | TV SET 1 |
| 1 | TV SET 2 |
| 2 | VIDEOTEXT |
| 3 | EXPANSION FOR TV 1 AND 2 |
| 4 | LASER VIDEO PLAYER |
| 5 | VIDEO RECORDER 1 (VCR 1) |
| 6 | VIDEO RECORDER 2 (VCR 2) |
| 7 | RESERVED |
| 8 | SAT 1 |

| | |
|---|---|
| **9** | EXPANSION FOR VCR 1 OR 2 |
| **10** | SAT 2 |
| **11** | RESERVED |
| **12** | CD VIDEO |
| **13** | RESERVED |
| **14** | CD PHOTO |
| **15** | RESERVED |
| **16** | AUDIO PREAMPLIFIER 1 |
| **17** | RECEIVER / TUNER |
| **18** | TAPE / CASSETE RECORDER |
| **19** | AUDIO PREAMPLIFIER 2 |
| **20** | CD |
| **21** | AUDIO RACK |
| **22** | AUDIO SAT RECEIVER |
| **23** | DCC RECORDER |
| **24** | RESERVED |
| **25** | RESERVED |
| **26** | WRITABLE CD |
| **26–31** | RESERVED |

- Bits9-14: هي بتات الأوامر الوظيفية، هذه البتات الستة تحتوي عن عنوان الأمر المرسل تبعاً للزر الموجود على جهاز التحكم، وهي تحقق لي استخدام 64 مفتاح وظيفي (2^6=64) وهي بالنسبة للأجهزة القياسية على الشكل التالي:

| COMMAND | DESCRIPTION of FUNCTION |
|---|---|
| 0-9 | NUMERIC KEYS 0 - 9 |
| 12 | STANDBY |
| 13 | MUTE |
| 14 | PRESETS |
| 16 | VOLUME UP |
| 17 | VOLUME DOWN |
| 18 | BRIGHTNESS + |
| 19 | BRIGHTNESS - |
| 20 | COLOR SATURATION + |
| 21 | COLOR SATURATION - |
| 22 | BASS UP |
| 23 | BASS DOWN |
| 24 | TREBLE + |
| 25 | TREBLE - |
| 26 | BALANCE RIGHT |

| | |
|---|---|
| 27 | BALANCE LEFT |
| 48 | PAUSE |
| 50 | FAST REVERSE |
| 52 | FAST FORWARD- |
| 53 | PLAY |
| 54 | STOP |
| 55 | RECORD |
| 63 | SYSTEM SELECT |
| 71 | DIM LOCAL DISPLAY |
| 77 | LINEAR FUNCTION (+) |
| 78 | LINEAR FUNCTION (-) |
| 80 | STEP UP |
| 81 | STEP DOWN |
| 82 | MENU ON |
| 83 | MENU OFF |
| 84 | DISPLAY A/V SYS STATUS |
| 85 | STEP LEFT |
| 86 | STEP RIGHT |
| 87 | ACKNOWLEDGE |
| 88 | PIP ON/OFF |
| 89 | PIP SHIFT |
| 90 | PIP MAIN SWAP |
| 91 | STROBE ON/OFF |
| 92 | MULTI STROBE |
| 93 | MAIN FROZEN |
| 94 | 3/9 MULTI SCAN |
| 95 | PIP SELECT |
| 96 | MOSAIC MULTI PIP |
| 97 | PICTURE DNR |
| 98 | MAIN STORED |
| 99 | PIP STROBE |
| 100 | RECALL MAIN PICTURE |
| 101 | PIP FREEZE |
| 102 | PIP STEP UP |
| 103 | PIP STEP DOWN |
| 118 | SUB MODE |
| 119 | OPTIONS BUS MODE |
| 123 | CONNECT |
| 124 | DISCONNECT |

## 4-7 ربط مستقبل IR إلى معالج مصغر:

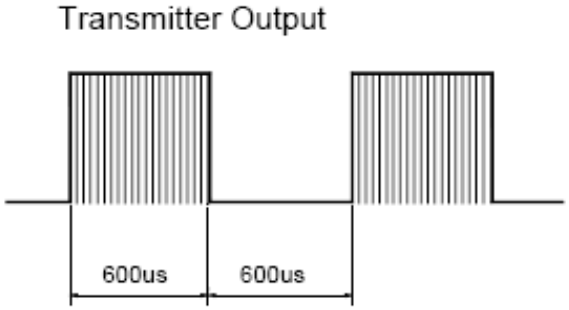
توضح هذه الفقرة بعض الأمور التي يجب مراعاتها عند وصل مستقبل أشعت تحت الحمراء مع متحكم مصغر.

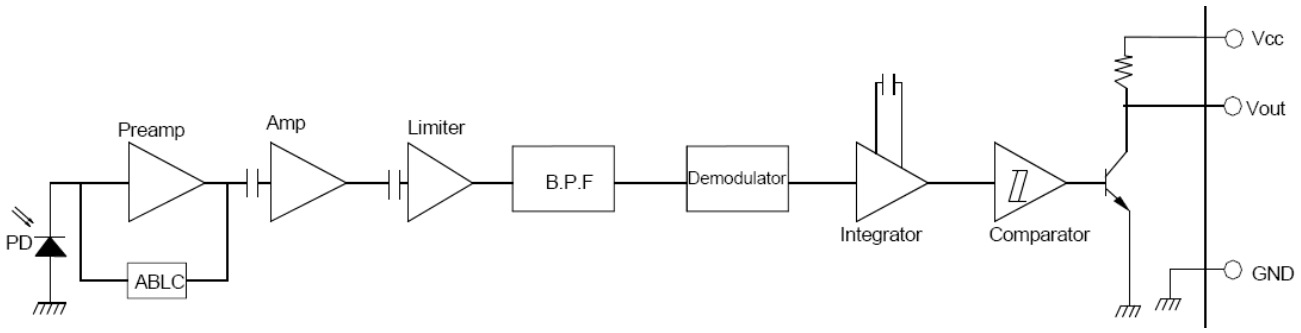
1) مستقبل الأشعة تحت الحمراء سوف يعكس المستوى المنطقي للنبضات – "1" = off | "0"=On.
2) في حال عدم الإرسال فإن خرج المستقبل سيكون على المستوى "1".
3) يمكن ربط خرج المستقبل إلى أي قطب من أقطاب المايكرو أو إلى قطب مقاطعة خارجية ومراقبة حالة القطب حتى تتغير حالته إلى المستوى المنخفض دلالةً على وجود حالة إرسال، حينها تبدأ باستقبال الشيفرة المؤلفة من 14 بت.

## 5-7 مستقبل الأشعة تحت الحمراء CLRM-2038S:

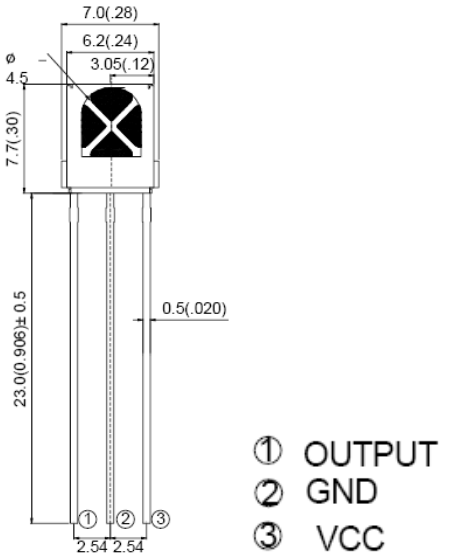
إن مستقبل الأشعة المستخدم في مشروعنا هو من النموذج CLRM-2038S وله المواصفات الأساسية التالية:

1) مستقبل أشعة تحت الحمراء ومضخم إشارة في نفس الوقت.
2) مرشح تمرير داخل غلاف المستقبل من أجل ترددات PCM.
3) مناعة عالية ضد التأثر بالأضواء المحيطة.
4) درع مطور للمناعة ضد اضطرابات الحقل الكهربائي.
5) استهلاك طاقة منخفض ضمن مجال العمل 2.7V~5.5V.
6) متوافق مع متطلبات المستوى المنطقي TTL, CMOS.
7) متوافق مع معايير NEC code,RC5 code.
8) تردد الحامل 38KHZ.
9) مسافة الاستقبال حتى 12m.
10) يمكن استخدامه من أجل التطبيقات التالية:
   - ✓ مفتاح ضوئي (Optical switch).
   - ✓ تطبيقات التحكم بالأجهزة مثل: Audio, TV, VCR, CD, MD, DVD, etc.
   - ✓ التحكم بالأجهزة المنزلية مثل: Air-conditioner, Fan, CATV, etc





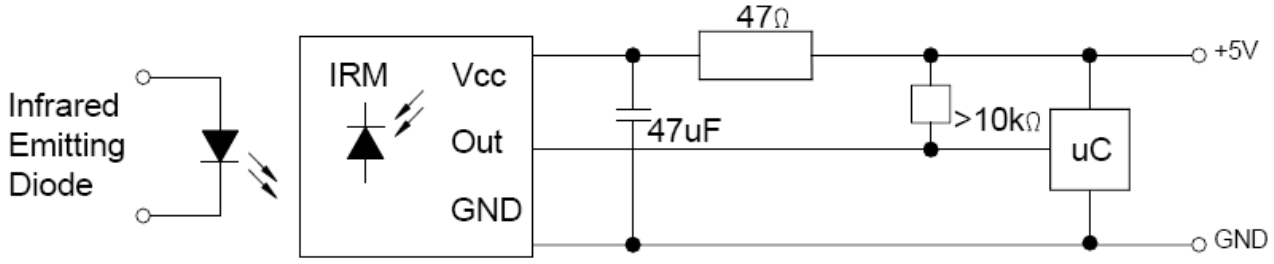

الشكل7 البنية الداخلية للمستقبل CLRM-2038S.



الشكل8 توزع الأقطاب للمستقبل CLRM-2038S.

## 6-7 دارة الملائمة لمستقبل الأشعة تحت الحمراء:

عند ربط مستقبل أشعة تحت الحمراء مع معالج، فإنه يجب وضع مكثف 4.7uF على التوازي مع أقطاب التغذية للمستقبل وأقرب ما يمكن إلى تلك الأقطاب، وإلا لن يعمل في الغالب. الشكل التالي يوضح دارة الملائمة لهذا المستقبل.




الشكل9 دارة الملائمة لمستقبل الأشعة تحت الحمراء CLRM-2038S

## 7-7 عناوين جهاز التحكم AL-AWAIL:

بالنسبة لجهاز التحكم المستخدم والموضح على الشكل10 فقد تم تصنيعه وتصميمه خصيصاً لشركة الأوائل للهندسة الإلكترونية وفق دلائل وظيفية خاصة. لذلك فإن لهذا الجهاز عنوان خاص وهو: RC5 Address = 27، وأما بالنسبة لأوامر المفاتيح على الجهاز فهي موضحة على الشكل في الطرف الأيمن باللون الأزرق علماً أن القيم هي بصيغة Hex.

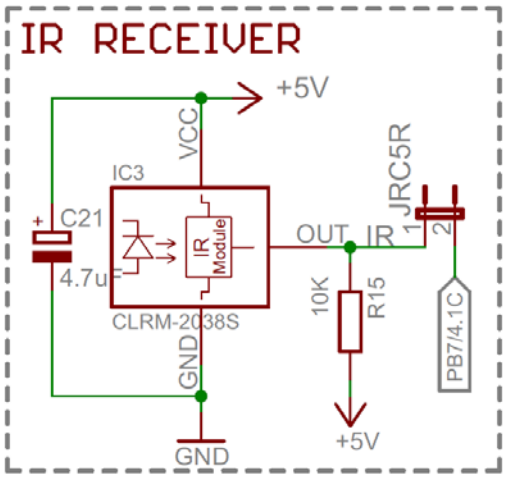
الشكل10 جهاز التحكم بالأشعة تحت الحمراء

## 8-7 تجربة توصيل وبرمجة مستقبل أشعة تحت الحمراء (CLRM-2038S) مع متحكم AVR.

المطلوب كتابة برنامج لاستقبال أوامر مرسلة من أجهزة التحكم بالأشعة تحت الحمراء والتي تعمل وفق البروتوكول RC5، وفي هذه الحالة سوف نستخدم التعليمات المخصصة للتعامل مع مستقبلات الأشعة تحت الحمراء التي تعتمد RC5 في البيئة Bascom-AVR. سوف يقوم البرنامج باستدعاء مكتبة التابع RC5 الموجودة في البيئة البرمجية والتي تحوي على بروتوكول الاستقبال RC5. يتم فحص حالة المستقبل باستخدام التعليمة Getrc5 والتي تقوم بتشغيل المؤقت Timer0 لعد النبضات بشكل آلي.

| شرح التعليمة | التعليمة البرمجية |
|---|---|
| تعريف القطب الموصول مع خرج مستقبل IR. | `Config Rc5 = Pinb.7 , Wait = 2000` |
| استحصال العنوان والأمر من المستقبل. | `Getrc5(address , Command)` |

الشكل11 يبين طريقة توصيل مستقبل أشعة تحت الحمراء (CLRM-2038S) مع المتحكم على اللوحة التعليمية Mini-Phoenix.




الشكل11 توصيل مستقبل أشعة تحت الحمراء (CLRM-2038S) مع المتحكم للتجربة 16

البرنامج Exp.16.bas في بيئة BASCOM-AVR:

```
' ************************************************************************
' * Title          : Exp.16.bas                                          *
' * Target Board   : Mini-Phoenix - REV 1.00                             *
' * Target MCU     : ATMega32A                                           *
' * Author         : Walid Balid                                         *
' * IDE            : BASCOM AVR 2.0.7.3                                  *
' * Peripherals    : RC5 Receiver;                                       *
' * Description    : Receiving RC5 Code from Remote Control              *
' ************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 9600
```

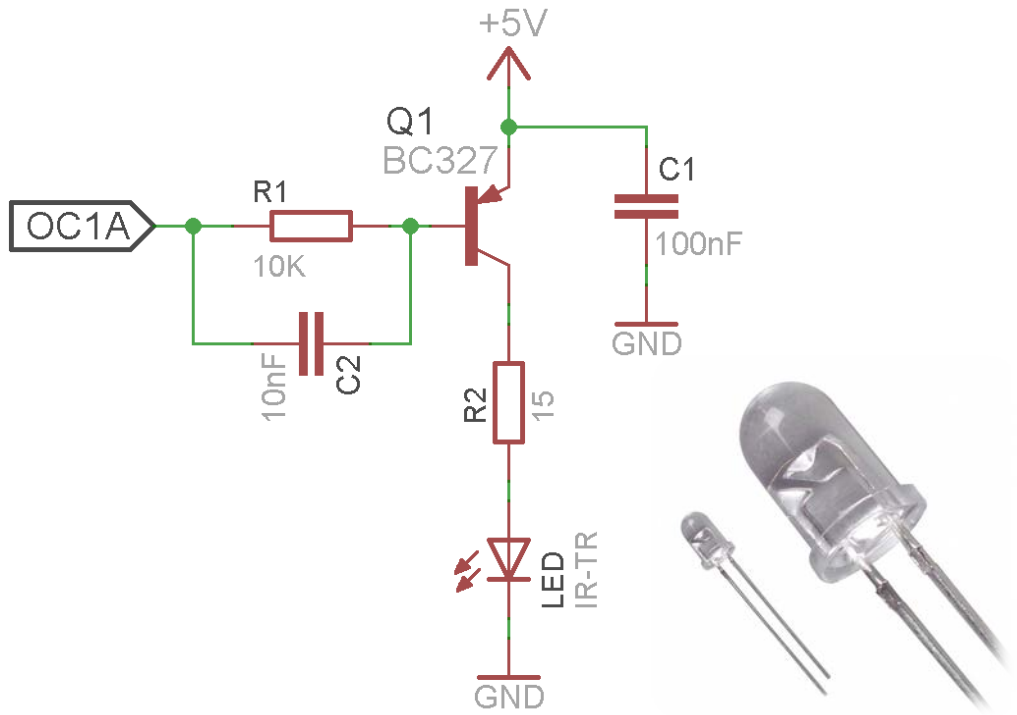
```
'-----------------------
'-----------------------[RC5 Receiver Configurations]
Config Rc5 = Pinb.7 , Wait = 2000
'-----------------------
'-----------------------[Variables]
Dim Rc5_address As Byte , Rc5_command As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Enable Interrupts
Do
   Gosub Read_rc5 : Waitms 100
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Read RC5 Code]
Read_rc5:
   Getrc5(rc5_address , Rc5_command)
   If Rc5_address <> 255 Then
      Rc5_command = Rc5_command And &B01111111

      Print "Address is: " ; Rc5_address
      Print "Command is: " ; Rc5_command
   End If
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

## 9-7 تجربة توصيل وبرمجة مرسل أشعة تحت الحمراء (IR LED) مع متحكم AVR.

المطلوب كتابة برنامج لإرسال أوامر تحكم بالأشعة تحت الحمراء تعمل وفق البروتوكول RC5، وفي هذه الحالة سوف نستخدم التعليمات المخصصة للتعامل مع مستقبلات الأشعة تحت الحمراء التي تعتمد RC5 في البيئة Bascom-AVR. سوف يقوم البرنامج باستدعاء مكتبة RC5 الموجودة في البيئة البرمجية Bascom-AVR والتي تحوي على بروتوكول الإرسال المطلوب. يتم إرسال البروتوكول باستخدام التعليمة RC5SEND والتي تقوم بتشغيل المؤقت Timer1 لحساب زمن النبضات بشكل آلي.

| التعليمة البرمجية | شرح التعليمة |
|---|---|
| `Rc5send Togbit , Address , Command` | تعليمة إرسال بت الحالة والعنوان والأمر على القطب OC1A وفق RC5 |

الشكل12 يبين طريقة توصيل مرسل أشعة تحت الحمراء إلى القطب OC1(A) مع المتحكم على اللوحة التعليمية Mini-Phoenix.




الشكل12 توصيل مرسل أشعة تحت الحمراء مع المتحكم للتجربة 17

البرنامج Exp.17.bas في بيئة BASCOM-AVR:

```
' ************************************************************************
' * Title          : Exp.17.bas                                          *
' * Target Board   : Mini-Phoenix - REV 1.00                             *
' * Target MCU     : ATMega32A                                           *
' * Author         : Walid Balid                                         *
' * IDE            : BASCOM AVR 2.0.7.3                                  *
' * Peripherals    : RC5 Sender                                          *
' * Description    : Sending RC5 Code using IR LED                       *
' ************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'----------------------[Definitions]
```

```
$regfile = "m32def.dat"
$crystal = 8000000
'-----------------------
'-----------------------[GPIO Configurations]
Config Pinb.2 = Input : Portb.2 = 1 : Send_ir Alias Pinb.2
Config Debounce = 500
'-----------------------
'-----------------------[Variables]
Dim Togbit As Byte , Command As Byte , Address As Byte
'---------------------
Command = 18 : Togbit = 0 : Address = 0
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   Debounce Send_ir , 0 , Power_command , Sub                'OC1A pin
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Send RC5 Code]
Power_command:
   If Togbit = 0 Then Togbit = 32 Else Togbit = 0
   Rc5send Togbit , Address , Command
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

. . . ﴿انتهت الجلسة العملية السابعة﴾ . . .

وليد بليد

- دمتم بخير ومودة ونور -

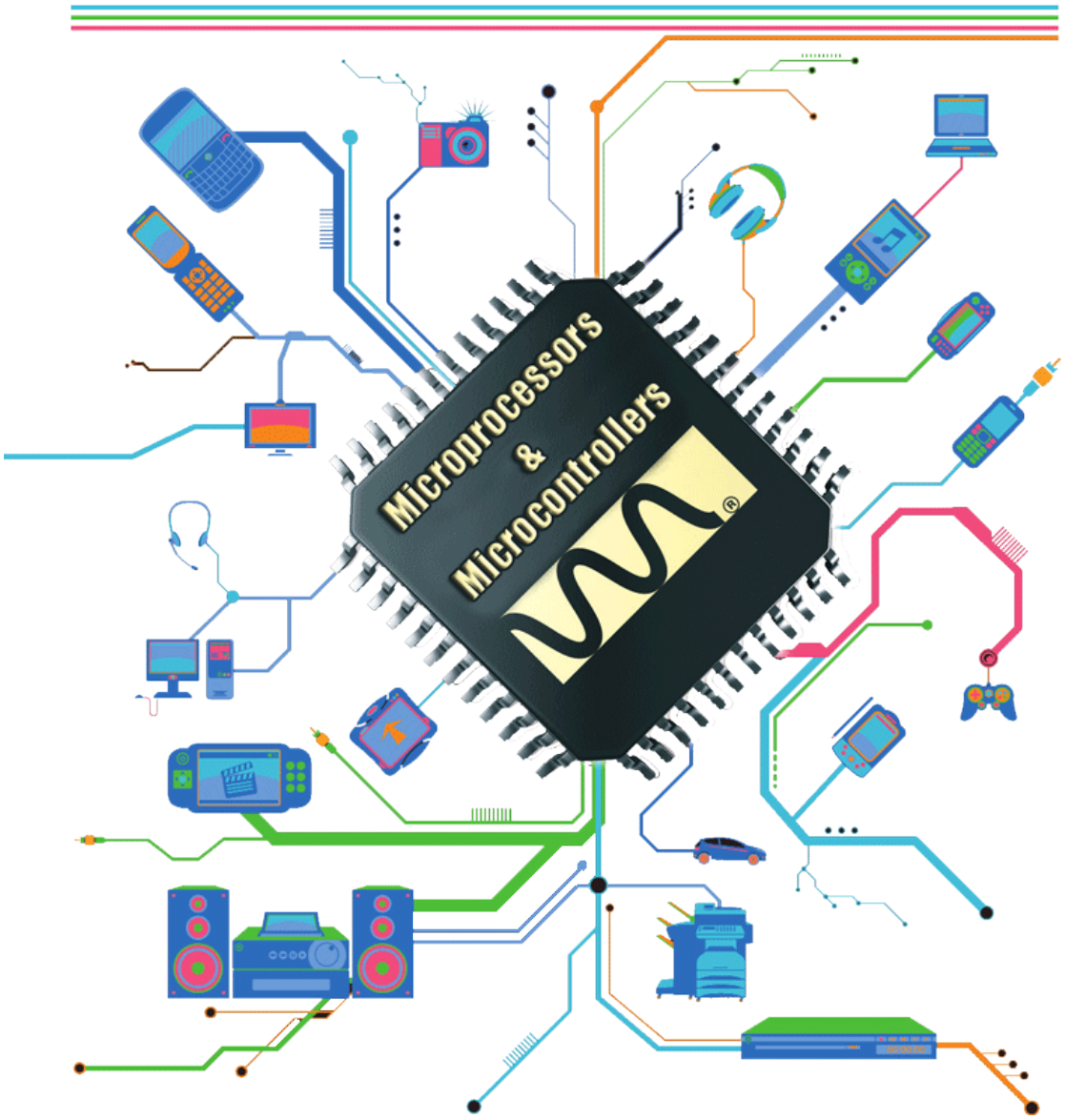
جامعـــــــــــة حلـــــــــــب
كلية الهندسة الكهربائية والإلكترونية
قسـم هندسـة التحكـم والأتمتـة

# ﴿ الجلسات العملية لمادة المعالجات والمتحكمات المصغرة ﴾

# Microprocessors & Microcontrollers Lab Sessions

## السنة الثالثة | قسم اتصالات

## ﴿ الجلسة العملية الثامنة ﴾




م. وليد بليد

Copyright © 2012 Walid Balid - All rights reserved.

Wednesday, April 25, 2012

# الجلسة العملية الثامنة

## نظرة عامة (Overview):

هذه المحاضرة تشرح مبادئ الاتصالات التسلسلية والنافذة التسلسلية اللامتزامنة UART.ثم برمجة النافذة UART في تطبيقات عدة منها: ربط متحكمات في شبكة سلكية، إرسال البيانات لاسلكياً باستخدام الأشعة تحت الحمراء، إرسال البيانات لاسلكياً باستخدام الليزر، إرسال البيانات لاسلكياً باستخدام الترددات الراديوية، وأخيراً ربط موديول GPS مع النافذة UART واستحصال الوقت والتاريخ والإحداثيات الجغرافية.

## 1-8 بروتوكولات الاتصال (Communication Protocols):

تتفرع بروتوكولات الاتصال بشكل عام إلى فرعين رئيسيين:

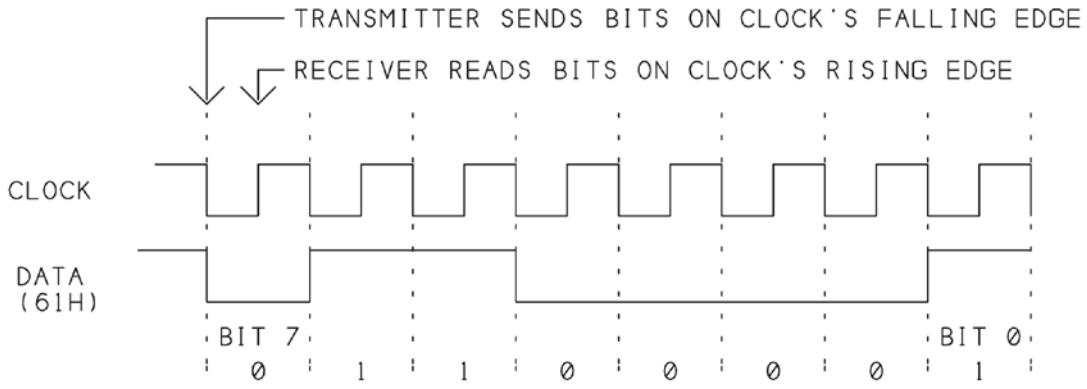
1) اتصالات تفرعية.. 2) اتصالات تسلسلية.

يختصر استخدم الاتصالات التفرعية من أجل نقل البيانات بسرعات عالية جداً ولمسافات قصيرة جداً، والسبب في محدودية المسافة هو تشكل السعات الطفيلية والضجيج العالي على مسارات خطوط النقل التفرعية عند ازدياد طول الناقل، كما أن حجم الناقل سيكون كبير وبالتالي فإن كلفة الناقل ستكون كبيرة أيضاً. في حين تستخدم الاتصالات التسلسلية على نطاق أوسع بكثير من الاتصالات التفرعية وتمتاز بمناعة عالية ضد الضجيج ونقل لمسافات بعيدة، كما أن حجم الناقل سيكون صغير وكلفته ضئيلة نسبياً مقارنة مع الناقل التفرعية.

| Serial Communications | | Parallel Communications |
|---|---|---|
| Asynchronous | Synchronous | |
| • Morse code telegraphy<br>• RS-232 (COM Port)<br>• RS-423<br>• RS-485<br>• Universal Serial Bus (USB)<br>• FireWire<br>• Ethernet<br>• Fiber Channel<br>• InfiniBand<br>• MIDI<br>• DMX512<br>• Serial ATA<br>• SpaceWire<br>• PCI Express<br>• SONET and SDH<br>• T-1, E-1 | o I2C<br>o SPI<br>o PS2 | ▪ LPT<br>▪ ISA<br>▪ EISA<br>▪ VESA<br>▪ ATA<br>▪ SCSI<br>▪ PCI<br>▪ PCMCIA<br>▪ IEEE-1284<br>▪ IEEE-488 |

## 2-8 مفاهيم أساسية في الاتصالات التسلسلية غير المتزامنة (Asynchronous):

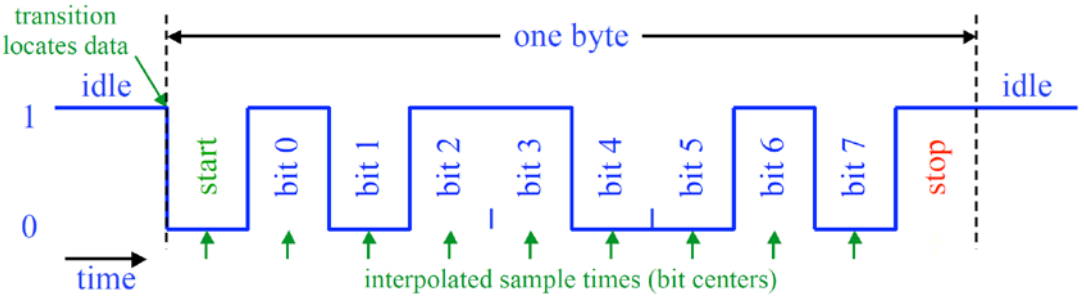
**أولاً: الاتصالات المتواقتة (المتزامنة):** يكون فيها بروتوكول الإرسال مؤلف من خطين على الأقل أحدهما خط التزامن (clock)، وبالتالي فإن سرعة إرسال البيانات تتحدد من خلال تردد إشارة التزامن بحيث يتم إرسال كل بت من البتات تسلسلياً عند جبهة التزامن (صاعدة أو هابطة).




الشكل1 إشارة البيانات وإشارة التزامن في بروتوكول إرسال متزامن

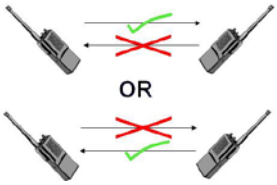
ملاحظة: بازدياد المسافة بين الطرفيتين فإنه يحصل انحراف\انزياح بين إشارة التواقت وبين إشارة البيانات مما يؤدي إلى فشل عملية النقل.

**ثانياً: اتصالات غير متواقتة (غير متزامنة):** لا تحوي على خط تزامن وإنما يتم بدء عملية الإرسال بإرسال بت بدء الإرسال (Start Bit) والذي بدوره يعلم المستقبل أن الذي يليه هو بايت البيانات، وبعدها يتم إرسال البايت المطلوب وتنتهي عملية إرسال البايت بإرسال بت التوقف (Stop Bit) والذي بدوره يعلم المستقبل أن عملية إرسال البايت قد انتهت ويجب تخزين البايت في مسجل نافذة الاستقبال والتحضر لاستقبال البايت التالي إن وجد.




الشكل2 إشارة البيانات في بروتوكول إرسال غير متزامن

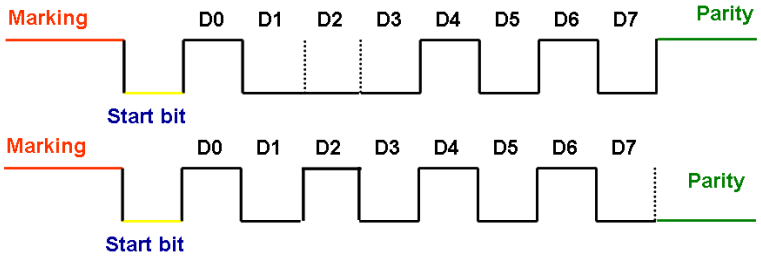
ملاحظة: بخلاف الاتصالات المتواقتة فإن ازدياد المسافة بين الطرفيتين لا يؤدي إلى فشل عملية النقل، كما أن هذه الطريقة أقل كلفة وأبسط بنية وأسهل برمجة.




**الإرسال أحادي الاتجاه (Half-Duplex):** تتم فيه عملية الاتصال بين الطرفيتين باتجاه واحد فقط في نفس اللحظة الزمنية، فإما أن تكون في حالة إرسال أو استقبال.

**الإرسال ثنائي الاتجاه (Full-Duplex):** يمكن أن تكون الوحدة الطرفية في حالة إرسال واستقبال في نفس اللحظة الزمنية.

**خانة الإيجابية (Parity Bit):** خانة يضيفها المرسل ويستخدمها المستقبل لضمان عدم ضياع المعلومات، وتتعلق خانة الإنجابية بعدد الواحدات في البايت المرسل.




الشكل3 توضع خانة الإيجابية في إشارة البيانات لبروتوكول إرسال غير متزامن

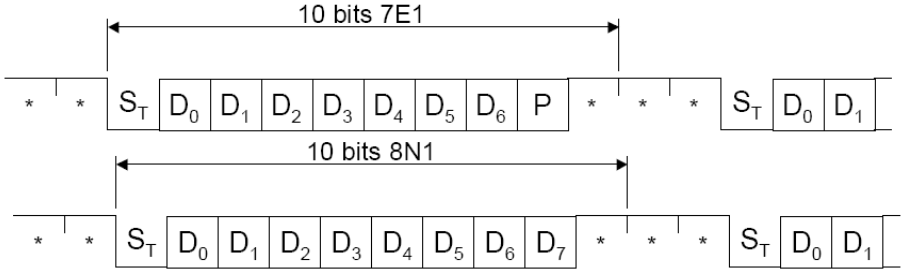
في حال كون خانة الإيجابية "Even" فإن هذه الخانة تملك القيمة "0" إذا كان عدد الواحدات في البايت المرسل زوجي وإلا فستصبح "1". الأمثلة التالية توضح ذلك.

10110010 > Parity Bit = 0 | 10110110 > Parity Bit = 1

في حال كون خانة الإيجابية "Odd" فإن هذه الخانة تملك القيمة "0" إذا كان عدد الواحدات في البايت المرسل فردي وإلا فستصبح "1". الأمثلة التالية توضح ذلك.

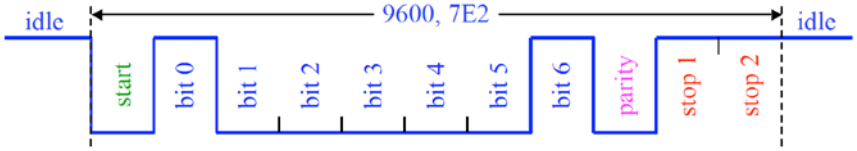
10110010 > Parity Bit = 1 | 10110110 > Parity Bit = 0

**عدد البتّات لكل محرف (N):** يتم فيها التصريح عن عدد البتات لبايت البيانات التي سيتم إرسالها، فإما أن تكون 5, 6, 7 or 8bit، ولكن يجب الانتباه مثلاً: في حال إرسال N=7bit فإن قيم العظمى ASCII=127.




الشكل4 مثال عن عدد بتات مختلف 8|7 في إشارتي بيانات لبروتوكول إرسال غير متزامن

خانة بت التوقف (Stop Bit): يعلم المرسل من خلالها المستقبل بانتهاء عملية الإرسال. 2 or 1.5 ,1 بت.




الشكل5 توضع خانة بت التوقف في نهاية إشارة البيانات لبروتوكول إرسال غير متزامن

معدل سرعة النقل (Baud Rate): وهو عدد البتات المرسلة خلال ثانية واحد على خط اتصال تسلسلي، وهناك قيم قياسية متعارف عليها لمعدلات النقل وهي: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, etc…

إن الزمن اللازم لإرسال بت واحد يعطى بالعلاقة التالية:

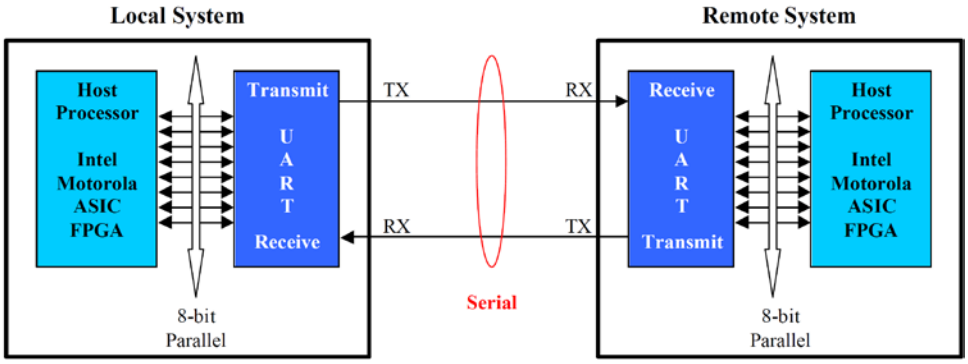
$$Bit_{Time} = \frac{1}{Baud\ Rate}$$

إن عدد البايتات التي يمكن إرسالها خلال ثانية واحدة يمكن حسابها من العلاقة التالية:

$$Bytes_{Num/1sec} = \frac{Baud\ Rate}{8}$$

## 3-8 النافذة التسلسلية UART (Universal Asynchronous Receiver and Transmitter Interface):

تعتبر هذه النافذة من أكثر نوافذ الاتصال التسلسلي استخداماً في الأنظمة الرقمية ومبدأ عملها وكذلك بروتوكولها متوافق تماماً مع البروتوكول RS232 إلا أن المستويات المنطقية فيها وفق المنطق TTL، وتتميز بسهولة وبساطة استخدامها بالإضافة إلى الكلفة المنخفضة للربط بين متحكمين (MCU-MCU)، أو الربط بين حاسب ومتحكم (MCU-PC).



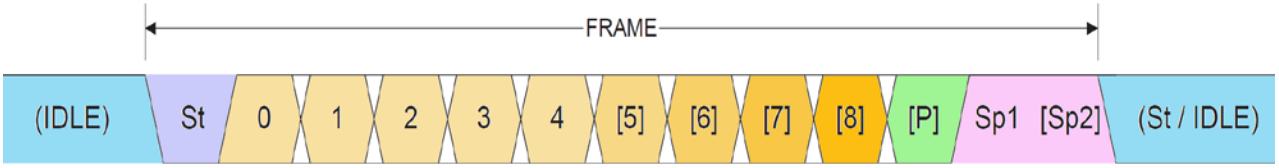

الشكل6 الربط بين متحكمين من خلال النافذة UART

تملك النافذة التسلسلية في متحكمات العائلة AVR على ميزات عديدة وهي تعمل في نمطين مستقلين:

- ✓ UART: نافذة تسلسلية عامة للإرسال والاستقبال اللامتزامن عبر القطبان TXD, RXD.
- ✓ USART: نافذة تسلسلية عامة للإرسال والاستقبال المتزامن عبر القطبان TXD, RXD بالإضافة إلى القطب XCK كقطب تزامن.

### بنية إطار البيانات (UART Frame Format):

إن تشكيل إطار البيانات المرسلة أو المستقبلة للنافذة UART مشابه تماماً لبنية إطار البروتوكول RS232 باختلاف وحيد وهو المستوى المنطقي المعكوس.




الشكل7 بنية إطار البيانات المرسلة أو المستقبلة للنافذة UART

*St*: Start bit, always low.

*Data bits*: (0 to 8).

*P*: Parity bit (Can be odd or even)

*Sp*: Stop bit, always high.

*IDLE*: No transfers on the communication line (RxD or TxD), IDLE line is high.

## حساب قيمة مسجل معدل النقل (Baud Rate Register):

من أجل تحديد معدل سرعة النقل للنافذة التسلسلية يتم شحن المسجل UBRR بقيمة تحسب وفقاً للعلاقات في الشكل8.

| Operating Mode | Equation for Calculating Baud Rate[1] | Equation for Calculating UBRR Value |
|---|---|---|
| Asynchronous Normal Mode (U2X = 0) | $BAUD = \frac{f_{OSC}}{16(UBRR+1)}$ | $UBRR = \frac{f_{OSC}}{16BAUD} - 1$ |
| Asynchronous Double Speed Mode (U2X = 1) | $BAUD = \frac{f_{OSC}}{8(UBRR+1)}$ | $UBRR = \frac{f_{OSC}}{8BAUD} - 1$ |
| Synchronous Master Mode | $BAUD = \frac{f_{OSC}}{2(UBRR+1)}$ | $UBRR = \frac{f_{OSC}}{2BAUD} - 1$ |

الشكل8 معادلات حساب قيمة المسجل UBRR الموافقة لمعدل النقل

حيث أنّ: UBRR هي محتوى المسجل UBRRH and UBRR وتتراوح 4095 – 0.

**مثال**: أحسب قيمة المسجل UBRR من أجل تردد هزاز كريستالي 1Mhz ومعدل نقل 9600bps ونمط عمل عام غير متواقت.

$$UBRR_{H,L} = \frac{f_{osc}}{16 \times Baud} - 1 = \frac{1000000}{16 \times 9600} - 1 = 5.510416 \approx 6$$

كما هو ملاحظ فإن القيمة غير دقيقة أي أن هناك خطأ في قيمة معدل النقل ولن تكون القيمة تماماً 9600، وبالتالي إذا كانت دارة المستقبل تعتمد تردد عمل مختلف وكان الخطأ مختلف فإنه ربما يحصل تشوه في البيانات بسبب عدم التزامن الدقيق في معدل النقل.

لذلك يوصى بمعدلات نقل قياسية وترددات هزازات كريستالية قياسية لتفادي الأخطاء الكبيرة في حساب معدلات النقل، بحيث أن الخطأ يجب أن لا يتجاوز 0.5% من أجل الحصول على وثوقية عمل عالية؛ لكن يمكن أن يعمل النظام بدون مشاكل حتى خطأ 5%.

يمكن حساب الخطأ من العلاقة التالية:

$$ERROR_{[\%]} = \left(\frac{BaudRate_{CloseMatch}}{BaudRate_{Calculated}} - 1\right) \times 100\%$$

مثال: من أجل نفس المثال السابق، نعوض في العلاقة السابقة:

$$ERROR_{[\%]} = \left(\frac{9600}{8928.571} - 1\right) \times 100\% = 7.52\%$$

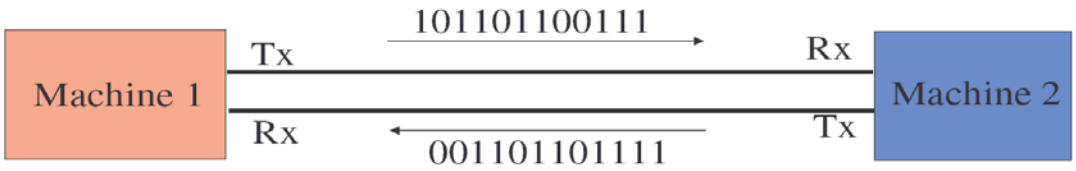
ملاحظة: من أجل تفادي مشكلة أخطاء معدل النقل قم باختيار تردد الهزاز الكريستالي بحيث يكون من مضاعفات معدل النقل.

## 8-4 تحقيق اتصال بين طرفيتين في برتوكول UART:

هناك بارامترات يجب تحديدها بين المرسل والمستقبل قبل إرسال البيانات في الاتصالات غير المتواقتة وهي:

- ✓ تحديد نمط الإرسال: أحادي الاتجاه (Half-Duplex) أو ثنائي الاتجاه (Full-Duplex).
- ✓ تحديد عدد البتات لكل محرف: 6, 7 or 8 bit.
- ✓ تحديد معدل سرعة الإرسال (Baud Rate).
- ✓ تحديد استخدام أو عدم استخدام خانة فحص الإيجابية (Parity Bit)، وفي حال الاستخدام يجب تحديد نمط فحص خانة الإيجابية (Even or Odd).
- ✓ تحديد عدد بتات التوقف (1, 1.5 or 2).

عموماً، فإنه من أجل تحقيق اتصال بين طرفيتين بدون مصافحة يكفي توصيل قطب الإرسال "**TxD**" والاستقبال "**RxD**" على التوازي المتعاكس كما في الشكل التالي:




الشكل9 تحقيق اتصال بين طرفيتين من خلال النافذة UART

## 8-5 هناك نمطين للبيانات في الاتصالات التسلسلية وهما:

1) **نمط الآسكي** (Ascii Mode): يتم تمثيل كل خانة على أنها محرف مستقل ويتم إرسال قيمة الآسكي لهذا المحرف. **مثال**: التعليمة "**Print 123**" ستقوم بإرسال الأرقام (1,2,3) على أنها محارف، وبالتالي سترسل الآسكي لكل منها [51][50][49] – بالنتيجة سترسل ثلاث بايتات.

2) **النمط الثنائي** (BIN Mode): يتم تمثيل البيانات على أنها قيمة عديدة وليس محرفية ويتم إرسال القيمة الثنائية لهذا العدد. **مثال**: التعليمة "**Printbin 123**" ستقوم بإرسال القيمة (123) على أنها بايت واحد، وبالتالي سترسل [1111011] – بالنتيجة سترسل بايت واحد فقط.

## 8-6 التعامل مع النافذة UART في Bascom-AVR:

1) تعليمات التهيئة (Configuration).
2) تعليمات الإرسال (Sending over TXD).
3) تعليمات الاستقبال (Receiving over RXD).

| التعليمة البرمجية | شرح التعليمة |
|---|---|
| `$baud = Var` | تحديد معدل النقل العام للنافذة التسلسلية UART0. |
| `Print Var ; "const"` | إرسال البيانات عبر النافذة التسلسلية UART0. |
| `Printbin Var [; Varn]` | إرسال البيانات بصيغة ثنائية عبر النافذة التسلسلية UARTx.<br>[; Varn]: خيار من أجل تحديد عدد البايتات المراد إرسالها (مصفوفة). |
| `Input ["prompt"] , Var [,Varn]` | قراءة البيانات الواردة على النافذة التسلسلية UART0.<br>["prompt"]: خيار يقوم بإرسال رسالة نصية قبل قراءة محتوى النافذة.<br>Var: المتحول الذي سيتم إدخاله (رقمي، محرفي).<br>[,Varn]: خيار من أجل إدخال أكثر من متحول بنفس التعليمة (n=1,2…). |
| `Inputbin Var1 [,Var2]` | قراءة البيانات الواردة على النافذة UART0 بصيغة ثنائية.<br>[,Var2]: خيار من أجل تحديد عدد البايتات المراد إدخالها (مصفوفة). |
| `Inputhex ["prompt"],Var[,Varn]` | قراءة البيانات الواردة على النافذة UART0 بالصيغة HEX. |
| `var = INKEY()` | تعود بقيمة الـ Ascii لأول محرف في مسجل buffer النافذة UART0. |
| `var = WAITKEY()` | ينتظر وصول أول محرف إلى مسجل buffer النافذة التسلسلية UART0 ويعود بقيمة الـ Ascii له. |
| `Var = Iscarwaiting()` | يفحص محتوى buffer مسجل النافذة UART0 ويعود بالقيمة "1" إذا كان هناك أي محرف، وإلا فسوف يعود بالقيمة "0"، مع العلم أن هذه التعليمة تفحص محتوى المسجل ولا تؤثر على محتواه! |

ملاحظة: من أجل إرسال أكثر من متحول على نفس السطر يمكن استخدام (;) للفصل بين المتحولات (Print A ; B ; C).

ملاحظة: إن التعليمة **Printbin** مكافئة تماماً للتعليمة **Print Chr (var);**.

ملاحظة: يمكن استخدام التعليمة **Printbin** من أجل إرسال عدة متحولات مخزنة في مصفوفة؛ كما في المثال التالي سوف يتم إرسال عشر بايتات موجودة في المتحول (مصفوفة) Arr.

```
Printbin Arr(1) ; 10
```

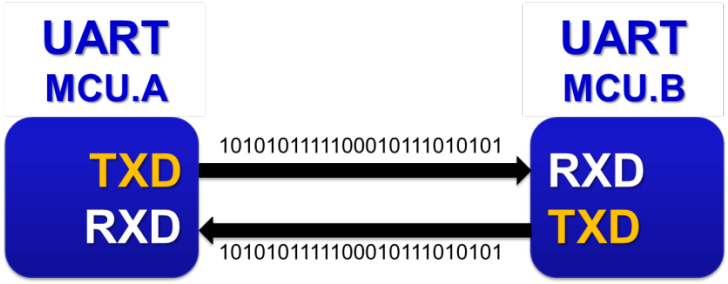
ملاحظة: يمكن استخدام التعليمة **Inputbin** من أجل إدخال عدة متحولات وإسنادها إلى مصفوفة؛ كما في المثال التالي سوف يتم استلام عشر بايتات ووضعها في المصفوفة Arr.

```
Inputbin Arr(1) , 10
```

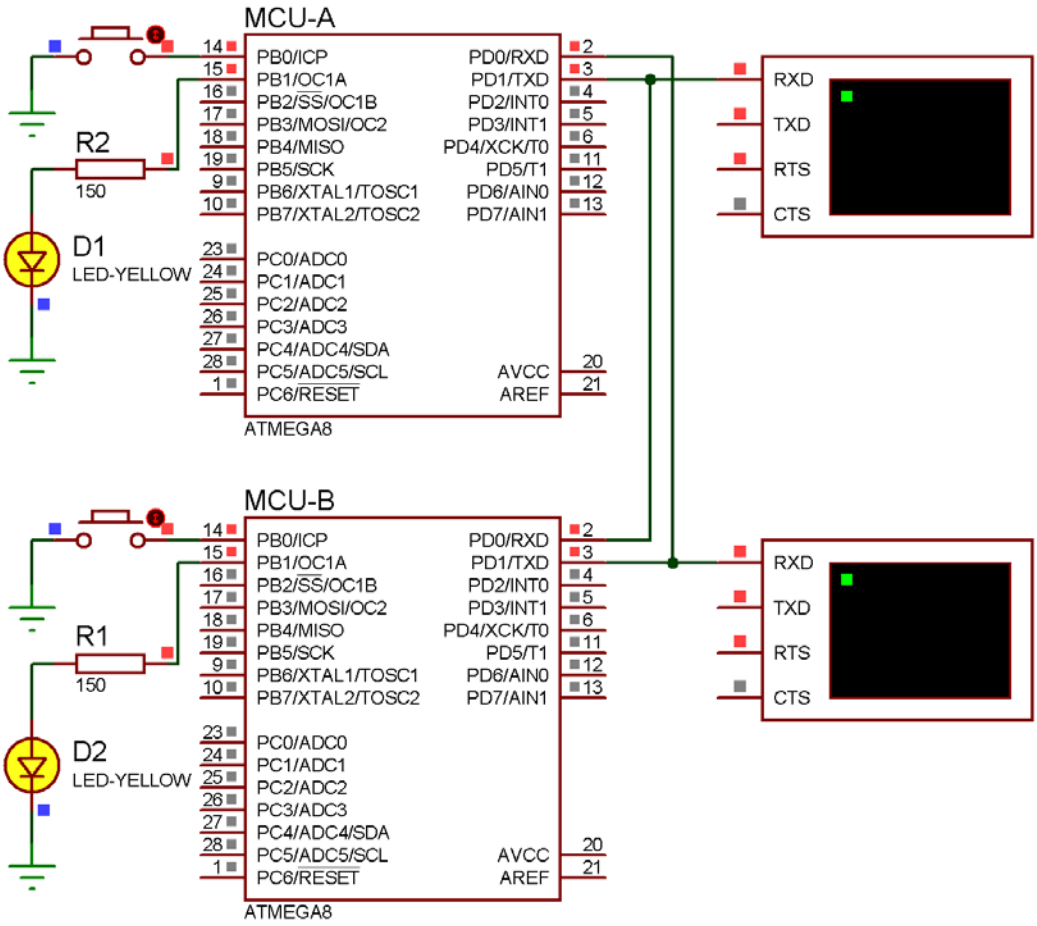
ملاحظة: إن التعليمة **Inputbin** سوف تنتظر حتى تستلم جميع البايتات المحددة في متحولاتها!

## 7-8 تطبيق: ربط متحكمي AVR من خلال النافذة التسلسلية UART . . .

المطلوب وصل متحكمي AVR من خلال النافذة التسلسلية UART بحيث يتم إرسال أوامر تحكم بينهما على الشكل التالي: عند الضغط على المفتاح الموصول مع المتحكم MCU-A سيتم إرسال المحرف "A" من MCU-A إلى MCU-B، وعندما يستلم المتحكم MCU-B المحرف "A" سيقوم بتغيير حالة الثنائي D2. وبالمثل تماماً: عند الضغط على المفتاح الموصول مع المتحكم MCU-B سيتم إرسال المحرف "B" من MCU-B إلى MCU-A، وعندما يستلم المتحكم MCU-A المحرف "B" سيقوم بتغيير حالة الثنائي D1.




الشكل10 المخطط التمثيلي لربط المتحكمين من خلال النافذة UART




الشكل11 يبين طريقة الوصل للنافذة التسلسلية بين المتحكمين

البرنامج "Exp.18-A.bas" للمتحكم MCU-A في بيئة BASCOM-AVR:

```
'-----------------------[Definitions]
$regfile = "m8def.dat"
$crystal = 8000000
$baud = 9600

'-----------------------[GPIO Configuration]
Config Pinb.0 = Input : Switch Alias Pinb.0 : Portb.0 = 1
Config Pinb.1 = Output : Led Alias Portb.1
'
'-----------------------[Variables]
Dim Var As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

'--->[Main Program]
Do
   If Ischarwaiting() = 1 Then
      Var = Inkey()
      If Var = "B" Then Toggle Led
   End If

   If Switch = 0 Then
      Print "A" : Waitms 200
   End If
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

البرنامج "Exp.18-B.bas" للمتحكم MCU-B في بيئة BASCOM-AVR:

```
'-----------------------[Definitions]
$regfile = "m8def.dat"
$crystal = 8000000
$baud = 9600

'-----------------------[GPIO Configuration]
Config Pinb.0 = Input : Switch Alias Pinb.0 : Portb.0 = 1
Config Pinb.1 = Output : Led Alias Portb.1

'-----------------------[Variables]
Dim Var As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

'--->[Main Program]
Do
   If Ischarwaiting() = 1 Then
      Var = Inkey()
      If Var = "A" Then Toggle Led
   End If

   If Switch = 0 Then
      Print "B" : Waitms 200
   End If
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

## 8-8 حلول التطوير باستخدام منفذ الاتصالات التسلسلي USB:

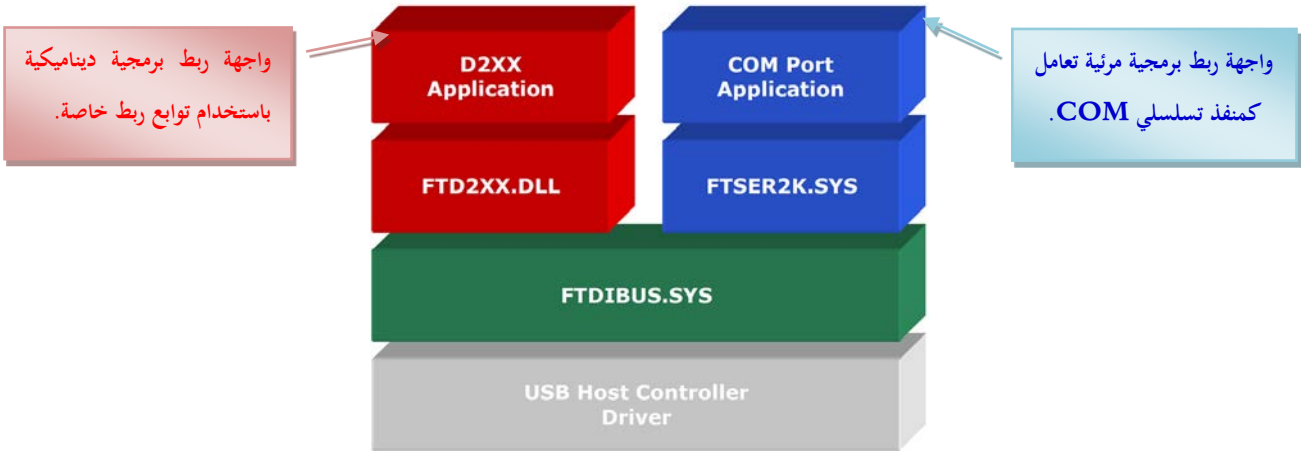
تعتبر تقنية USB في الوقت الحالي من التقنيات المعقدة حيث أن تضمين منفذ USB في النظام الإلكتروني وكتابة برنامج القيادة الخاص به على الحاسب أمر شديدة التعقيد، وذلك لأنه يتوجب على المصمم تحقيق أمرين:

1. تصميم عتاد الكتروني (Hardware) يحقق معايير البروتوكول USB.
2. كتابة برنامج التعريف الخاص بقيادة هذا العتاد.

لذلك وبسبب الطلب المتزايد على هذه التقنية واقتحامها للسوق العالمية فإن هنالك الكثير من الشركات التي وفرت على المصممين عناء تصميم العتاد الالكتروني لينصّب اهتمامهم على كتابة برامج القيادة، لذلك كل ما يتوجب على المصمم هو الاطلاع على معايير USB بغرض فهم كيفية التعامل مع هذا العتاد الالكتروني.

تقدم بعض الشركات حلولاً للتعامل مع المنفذ USB باستخدام شرائح متكاملة تقوم على تحويل البروتوكول USB إلى نافذة تسلسلية UART تمكن المستخدم من توصيل المتحكم المصغر بشكل مباشرة مع هذه النافذة، بالإضافة إلى ذلك توفر هذه الشرائح حلولاً برمجية من خلال مكتبات ربط ديناميكية من أجل ربط نظام مع الحاسب عن طريق البروتوكول USB ومعالجة بارامترات النظام أو إرسال أوامر التحكم إلى النظام. من أشهر وأكثر الشرائح انتشاراً واستخداماً هي الدارة المتكاملة FT232 التي هي عبارة عن دارة تحويل USB<>UART التي تنتجها شركة FTDI. حيث أن عملية تحويل البروتوكول USB تم بنائها في داخل هذه الشريحة ككيان صلب (Hardware) دون الحاجة إلى برمجة الشريحة، حيث تؤمن هذه الشريحة واجهتي ربط ديناميكي للتعامل برمجياً مع المنفذ باستخدام توابع خاصة وجاهزة موجودة في مكتبات الربط الديناميكي للشريحة دون الحاجة إلى بناء البروتوكول USB بشكل برمجي من البداية أو حتى فهم مبدأ عمله.

إن واجتهي الربط (D2XX driver & VCP driver) التي تؤمنها هذه الشريحة هي على الشكل التالي:



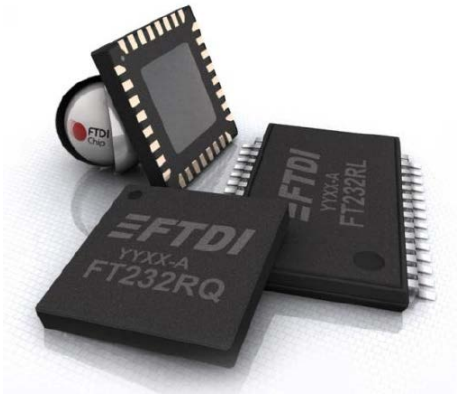

الشكل12 واجهتي الربط (نمطي العمل) للشريحة FT232R المخصصة للتحويل UART<>USB

فيما يلي جدول مقارنة بين واجتهي الربط (D2XX driver & VCP driver) للشريحة FT232R:

| | VCP Driver | D2XX.DLL Driver |
|---|---|---|
| بساطة البرنامج | برنامج بسيط | برنامج معقد |
| السرعة | سرعة ثابتة لا يمكن تغيرها 300 KB/s | سرعة قابلة للتغيير تصل إلى 3MB |
| التحكم بالشريحة | لا يمكن التحكم بالشريحة | تحكم كامل ومباشر بالشريحة |

- VCP (Virtual Com Port): يعرف منفذ USB كمنفذ COM تسلسلي إضافي، مما يسمح لنا بالتخاطب مع منفذ USB كمنفذ Com معياري.
- D2XX.DLL: يسمح هذا التعريف بالوصول المباشر إلى كامل مميزات هذه الشريحة عن طريق أوامر موجودة ضمن مكتبة ربط ديناميكية DLL.

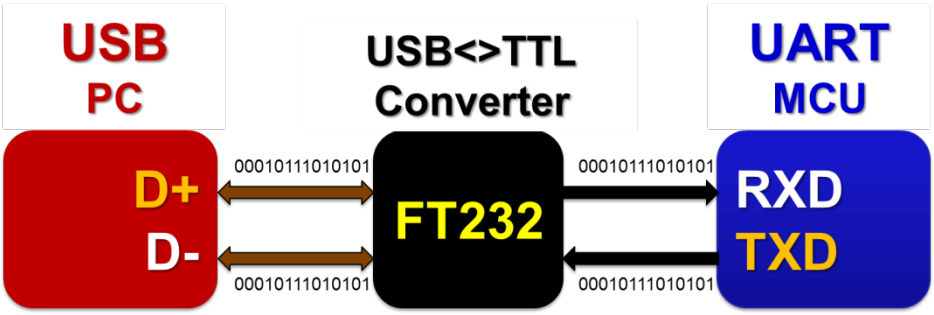
## 9-8 الشريحة FT232R – دارة متكاملة مخصصة للتحويل UART<>USB:



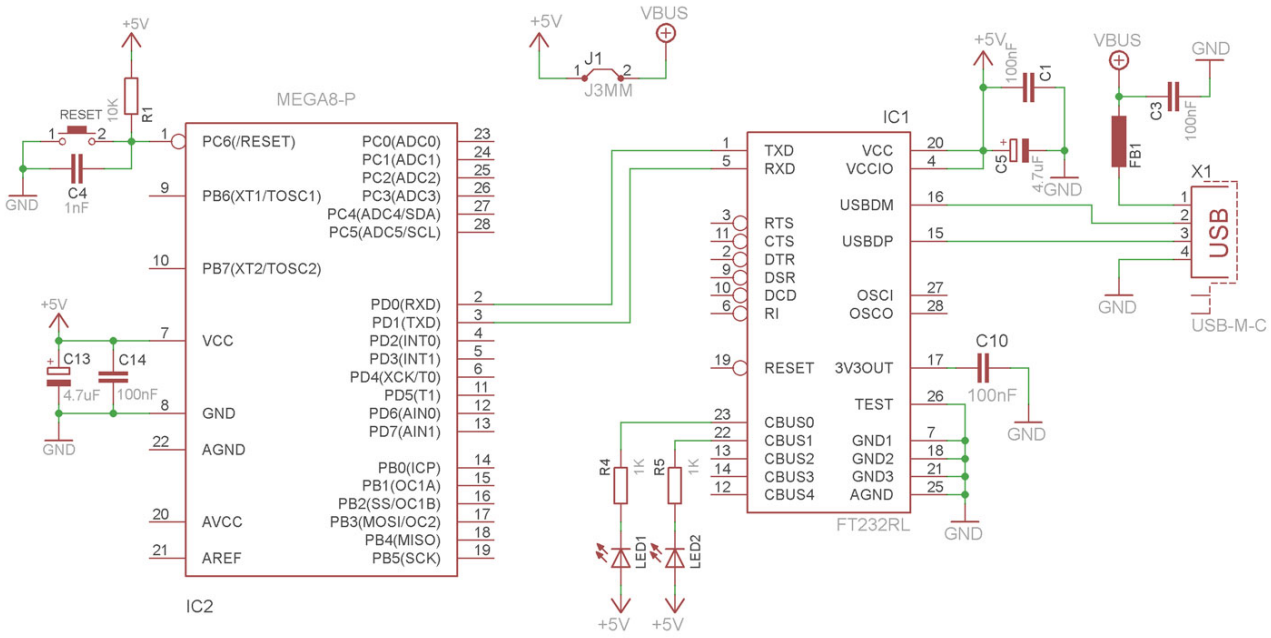

- ✓ توفر الشركة الصانعة برنامج القيادة لهذه الشريحة بشكل مجاني متوافق مع معظم أنظمة التشغيل.
- ✓ تقدم شركة FTDI برنامجي قيادة لشرائحها (VCP & D2XX.DLL).
- ✓ متوافقة مع المعيارين USB1.1, USB2.0.
- ✓ تدعم هذه الشريحة ملائمة كاملة لنظم الاتصالات التسلسلية.
- ✓ سرعة اتصال 300kb~3Mb بحسب نوع برنامج القيادة.
- ✓ ذاكرة استقبال وسيطية من نوع FIFO بطول 256 بايت.
- ✓ ذاكرة إرسال وسيطية من نوع FIFO بطول 128 بايت.
- ✓ رقمي VID, PID ورقم تسلسلي للمنتج ووصف لهذا الجهاز.
- ✓ توفر العديد من المقالات التقنية من الشركة المصنعة تقدم معلومات مفصلة عن طرق استخدام هذه الشريحة.

تلعب هذه الشريحة دور الملائم بين منفذ USB وبين النظام حيث تقوم باستقبال بيانات منفذ USB وتستخلص منها البيانات المطلوبة، كما تقوم بإرسال البيانات من المتحكم بشكلها التسلسلي إلى منفذ USB بعد إضافة الحقول اللازمة لتحقيق بروتوكول USB.

## 10-8 ربط متحكم AVR من خلال النافذة UART (TTL) مع منفذ USB (Differential) .

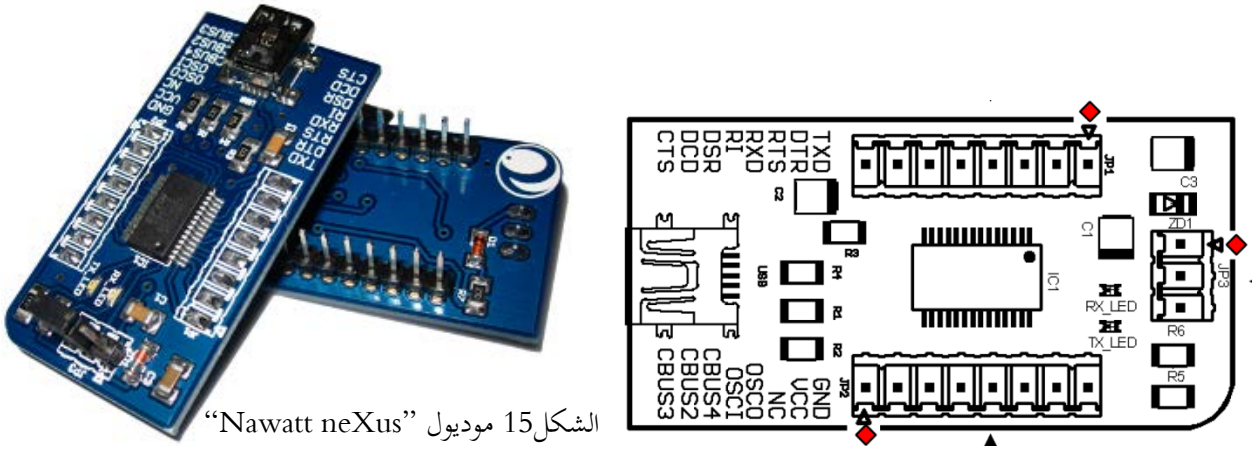



الشكل13 المخطط التمثيلي لربط متحكم AVR مع منفذ USB من خلال الشريحة FT232R




الشكل14 مخطط التوصيل (Schematic) لربط متحكم AVR مع منفذ USB من خلال الشريحة FT232R

إن التعامل فيزيائياً مع الشريحة FT232R يعتبر أمراً صعباً لعدم توفرها في غلاف فيزيائي من النوع DIP وهي فقط متوفرة كعنصر SMD، لذلك يمكن استخدام موديول التحويل UART<>USB الجاهز "Nawatt neXus" أو أي موديول آخر مشابه.



الشكل15 موديول "Nawatt neXus"

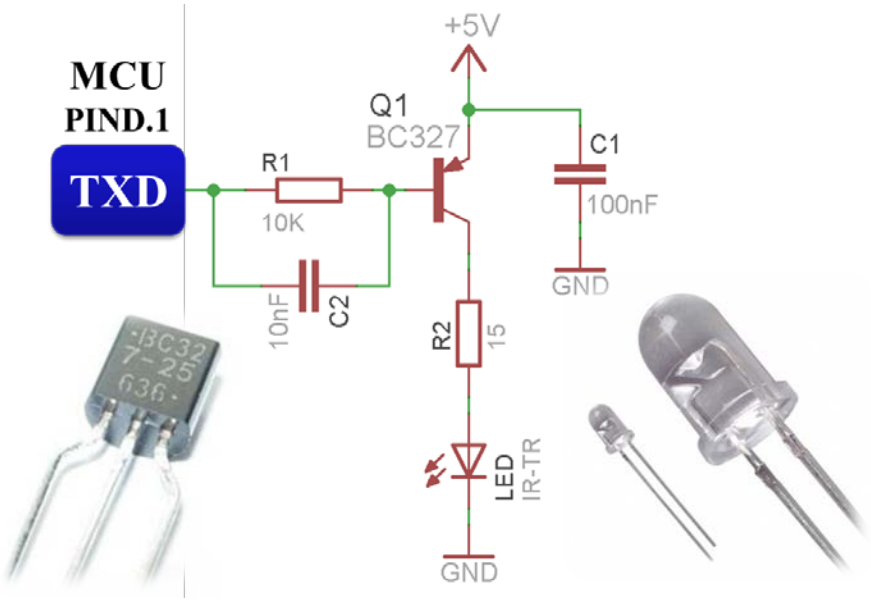
يمتلك الموديول العديد من الأقطاب، ولكن يلزمنا فقط الأقطاب التالية:

- ✓ TXD: قطب الإرسال من الحاسب.
- ✓ RXD: قطب الاستقبال من الحاسب.
- ✓ GND: قطب الأرضي 0V من الحاسب.
- ✓ +5V: قطب التغذية 5V+ من الحاسب (يستخدم فقط عندما يراد الحصول على تغذية من USB من أجل تغذية المتحكم).

## 11-8 تطبيق: إرسال البيانات بين متحكم AVR والحاسب باستخدام الأشعة تحت الحمراء (IR Data Link):

إن عملية إرسال البيانات بين متحكم AVR من خلال النافذة UART (TTL) والحاسب عبر منفذ USB (Differential) ستتم في هذا التطبيق باستخدام الأشعة تحت الحمراء، وبالتالي سيتضمن التصميم دارتين:

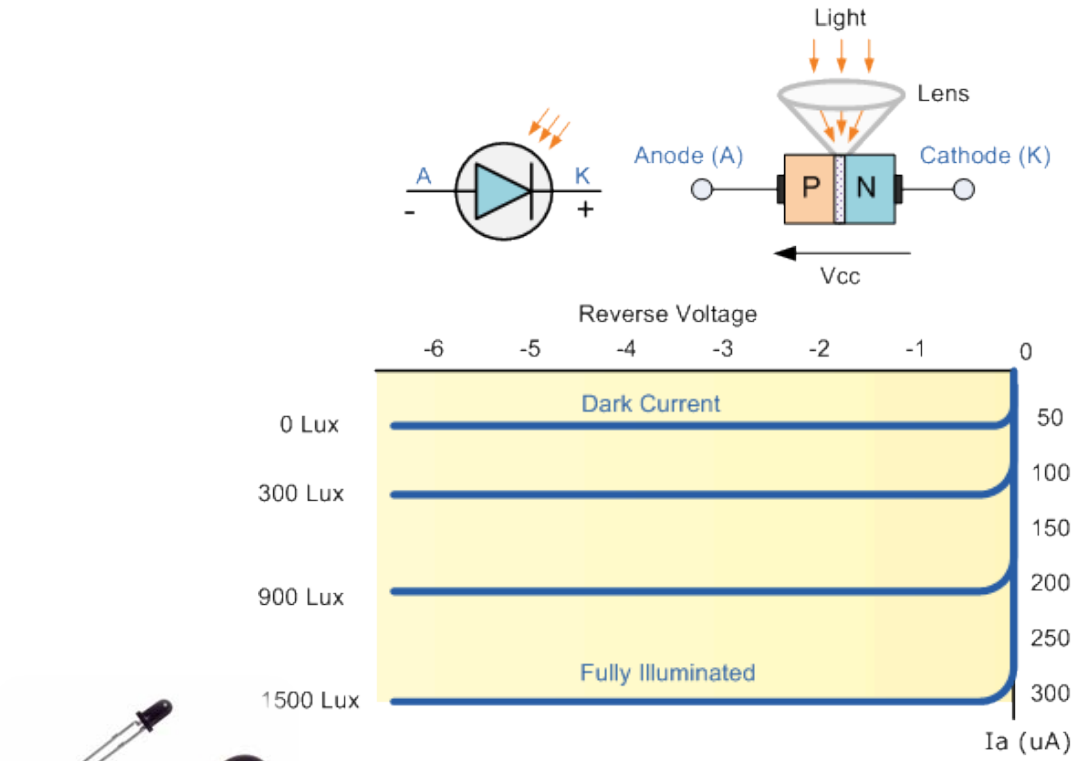
1) **دارة الإرسال للأشعة تحت الحمراء (IR Data Sender)**: وهي عبارة عن مرسل أشعة تحت الحمراء (IR LED) متحكم به عن طريق مفتاح إلكتروني ترانزستوري (Q1). إن التيار الاسمي للثنائي LED يتراوح بين 25~100mA وكلما ازدادت قيمة التيار ازدادت استطاعة الإرسال وجهد العمل للثنائي 2V (R2 = 30Ω). تم توصيل مدخل دارة الإرسال إلى قطب الإرسال للنافذة التسلسلية UART للمتحكم. وبالتالي فإن جميع البيانات الصادرة من النافذة على القطب TXD سوف ترسل على شكل ثنائي (0,1) على المرسل IR-LED.



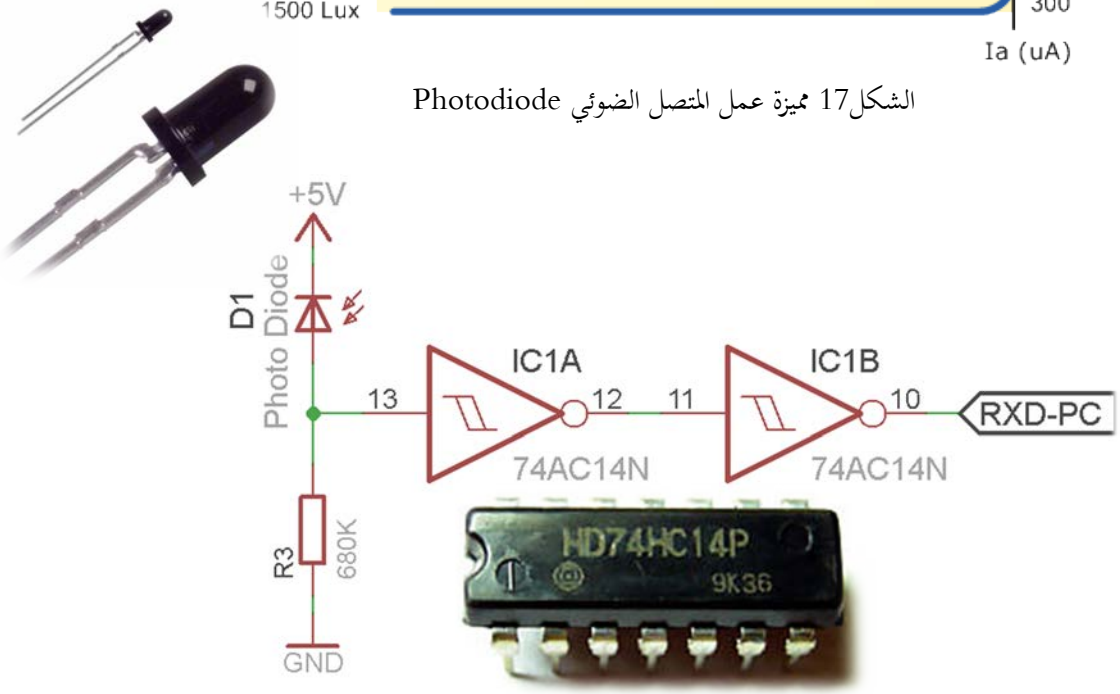

الشكل16 مخطط التوصيل (Schematic) لدارة الإرسال بالأشعة تحت الحمراء ووصلها مع القطب TXD للمتحكم

2) دارة الاستقبال للأشعة تحت الحمراء (IR Data Receiver): وهي عبارة عن متصل ضوئي (Photodiode) محيز عكسياً بحيث أنه عندما يتم تسليط ضوء على نافذة الثنائي التي تمثل المنطقة الفاصلة بين المتصل P/N يقوم على تمرير كمية أكبر من

التيار كما هو مبين على مميزة العمل في الشكل17. عندما يكون الثنائي في الظلام فإن مقاومة الثنائي تكون كبيرة جداً (بالميغا أوم)، وعندما يتم تسليط الضوء على الثنائي تصبح مقاومته بضع كيلو أوم، كما أن تغير شدة الضوء الساقط على الثنائي سيؤدي إلى تغير مطال الخرج على طرفي المقاومة R3، وبالتالي سنستخدم قادح شميث (74HC14) لتثبيت المطال بحيث تتأرجح إشارة الخرج بين القيمة "0" (عندما يرسل المرسل القيمة المنطقية "0") والقيمة "1" (عندما يرسل المرسل القيمة المنطقية "1"). الشكل18 يبين مخطط دارة الاستقبال وتوصيلها مع القطب RXD لموديول الوصل مع الحاسب neXus.




الشكل17 مميزة عمل المتصل الضوئي Photodiode

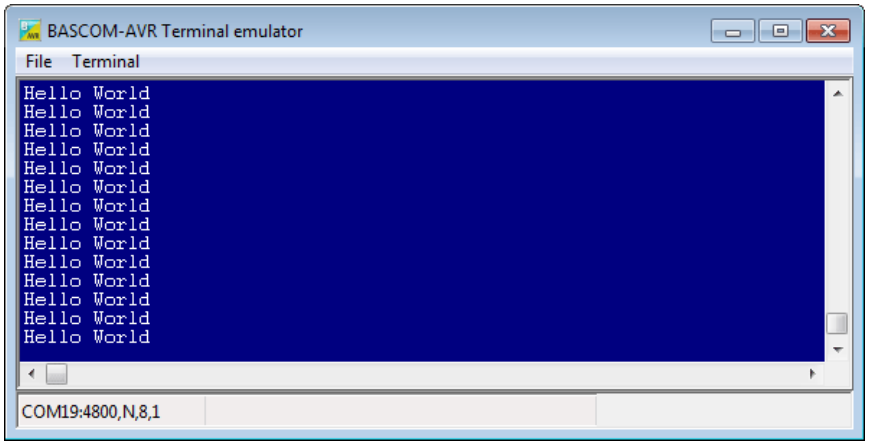



الشكل18 مخطط التوصيل (Schematic) لدارة الاستقبال بالأشعة تحت الحمراء ووصلها مع القطب RXD للموديول neXus

البرنامج "Exp.19.bas" في بيئة BASCOM-AVR:

```
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 4800
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   Print "Hello World" : Waitms 50
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

سيقوم البرنامج بإرسال (TXD) العبارة "Hello World" كل 50 ميلي ثانية على النافذة التسلسلية (UART) بشكل مستمر. على الطرف الآخر سيكون المستقبل (Photodiode) موصل مع منفذ USB من خلال الموديول neXus وبالتالي يمكن عرض القيم المستقبلة من خلال النافذة Terminal – الشكل19.




الشكل19 خرج دارة الاستقبال في نافذة Terminal في الحاسب

في حال طلب استقبال البيانات من خلال متحكم آخر بدل الحاسب وعرضها على شاشة إظهار LCD فيتم ذلك بتوصيل خرج دارة الاستقبال إلى القطب RXD للمتحكم الآخر وسيكون البرنامج على الشكل التالي:

البرنامج "Exp.20.bas" في بيئة BASCOM-AVR:

```
'-----------------------[Definitions]
$regfile = "m8def.dat"
$crystal = 8000000
$baud = 4800

'-----------------------[LCD Configuration]
Config Lcdpin = Pin , Db4 = Portc.2 , Db5 = Portc.3 , Db6 = Portc.4 , Db7 =
Portc.5 , E = Portd.3 , Rs = Portd.4
Config Lcd = 16 * 2
```

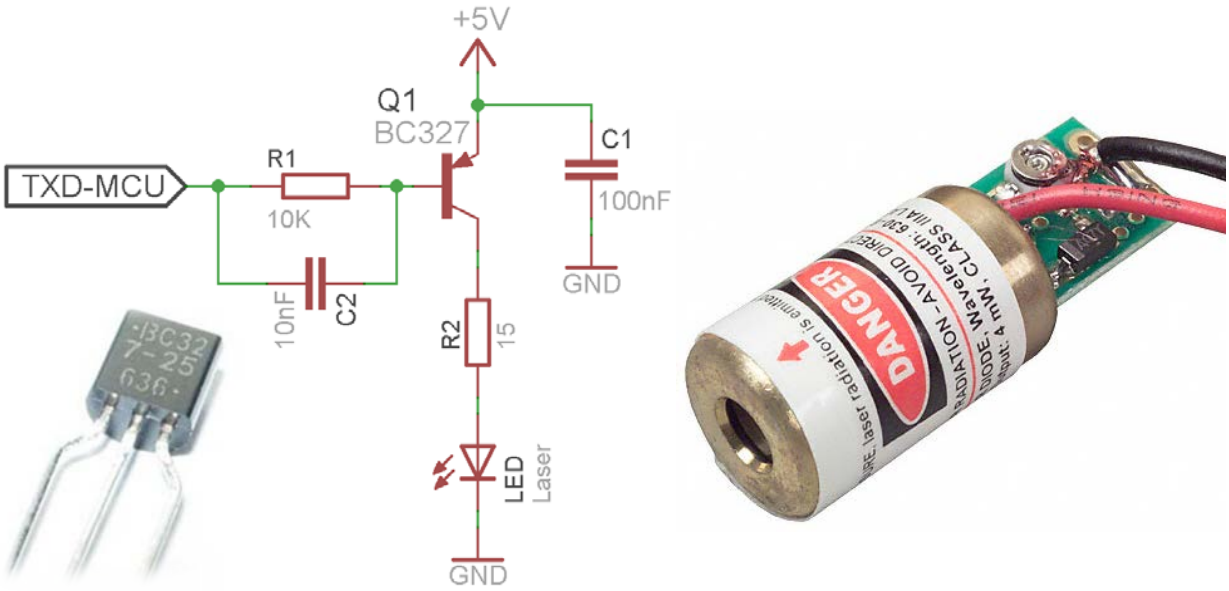
```
'-----------------------[Variables]
Dim Var As Byte
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   If Ischarwaiting() = 1 Then
      Var = Inkey() : LCD Var
   End If
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

سوف يقوم التابع "**Ischarwaiting()**" بفحص محتوى مسجل الدخل للنافذة التسلسلية (UART) وفي حال ورود بيانات سيتحقق الشرط (**Ischarwaiting() = 1**) ويتم قراءة البيانات الواردة (**Var = Inkey()**) وعرضها على شاشة LCD.

## 12-8 تطبيق: إرسال البيانات بين متحكم AVR والحاسب باستخدام الليزر (Laser Data Link):

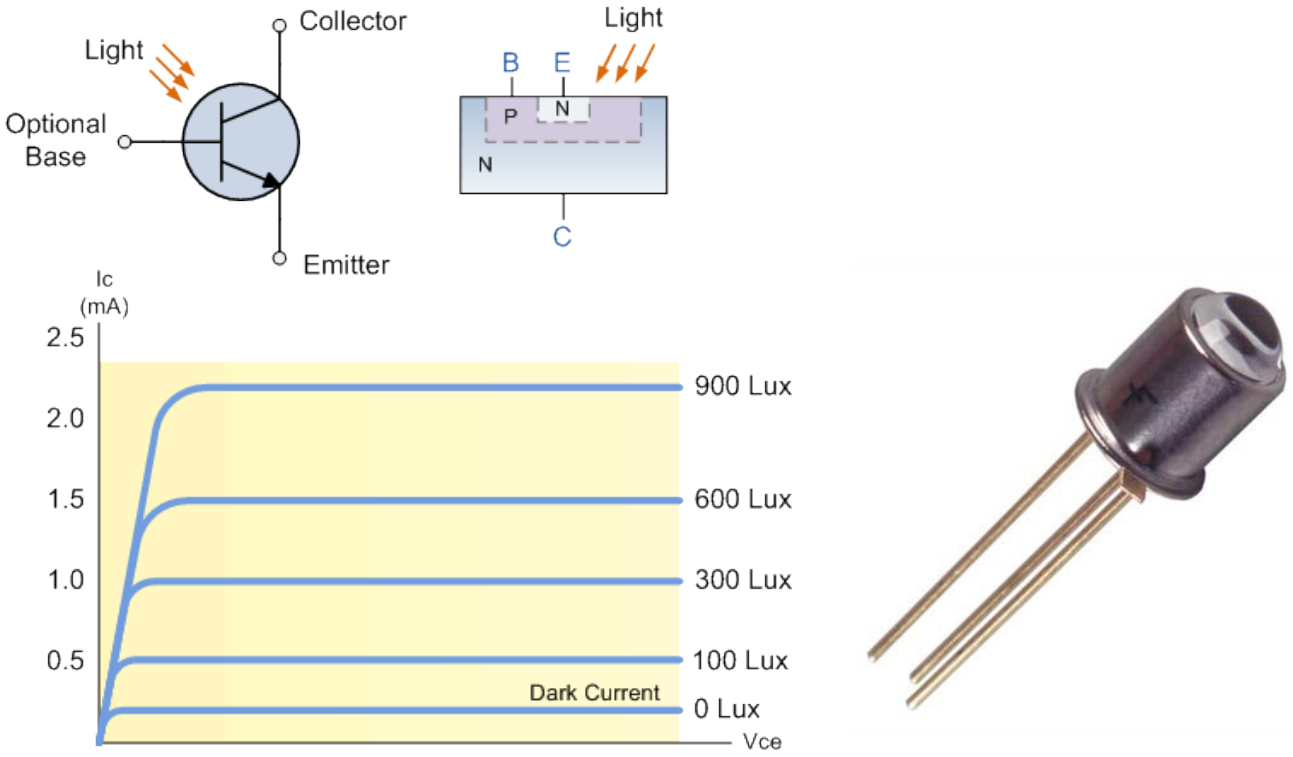
إن عملية إرسال البيانات بين متحكم AVR من خلال النافذة UART (TTL) والحاسب عبر منفذ USB (Differential) ستتم في هذا التطبيق باستخدام أشعة الليزر، وبالتالي سيتضمن التصميم دارتين:

1) **دارة الإرسال لأشعة الليزر (Laser Data Sender)**: وهي عبارة عن مرسل ليزري (Laser LED) متحكم به عن طريق مفتاح إلكتروني ترانزستوري (Q1). إن التيار الاسمي للثنائي LED يتراوح بين 25~100mA وكلما ازدادت قيمة التيار ازدادت استطاعة الإرسال وجهد العمل للثنائي 2V (R2 = 30Ω). تم توصيل مدخل دارة الإرسال إلى قطب الإرسال للنافذة التسلسلية UART للمتحكم. وبالتالي فإن جميع البيانات الصادرة من النافذة على القطب TXD سوف ترسل على شكل ثنائي (0,1) على المرسل Laser-LED.

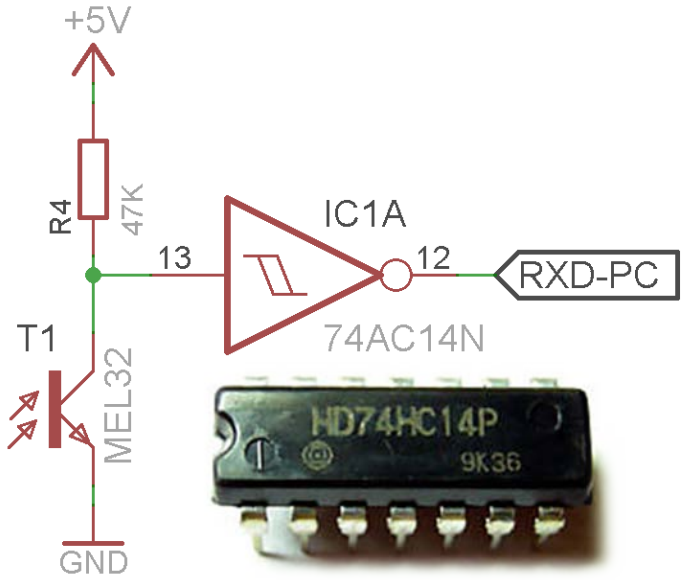



الشكل20 مخطط التوصيل (Schematic) لدارة الإرسال بأشعة الليزر ووصلها مع القطب TXD للمتحكم

2) **دارة الاستقبال لأشعة الليزر (Laser Data Receiver)**: وهي عبارة عن ترانزستور ضوئي (Phototransistor) محيز أمامياً بحيث أنه عندما يتم تسليط ضوء على نافذة الترانزستور التي تمثل القاعدة فسوف يقوم الترانزستور بتمرير كمية أكبر من التيار كما هو مبين على مميزة العمل في الشكل21. عندما يكون الترانزستور في الظلام فإن مقاومة الترانزستور تكون كبيرة جداً (بالميغا أوم) وسيكون في حالة القطع، وعندما يتم تسليط الضوء سوف يفتح الترانزستور، كما أن تغير شدة الضوء الساقط على الترانزستور سيؤدي إلى تغير مطال الخرج على طرفي الترانزستور، وبالتالي سنستخدم قادح شميث (74HC14) لتثبيت المطال بحيث تتأرجح إشارة الخرج بين القيمة "0" (عندما يرسل المرسل القيمة المنطقية "0") والقيمة "1" (عندما يرسل المرسل القيمة المنطقية "1"). الشكل22 يبين مخطط دارة الاستقبال وتوصيلها مع القطب RXD لموديول الوصل مع الحاسب neXus.




الشكل21 مميزة عمل الترانزستور الضوئي Phototransistor



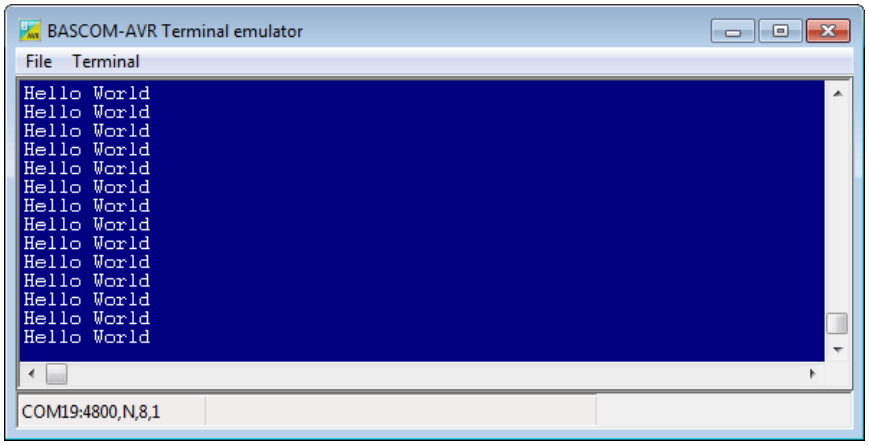

الشكل22 مخطط التوصيل (Schematic) لدارة الاستقبال بالأشعة تحت الحمراء ووصلها مع القطب RXD للموديول neXus

البرنامج "Exp.19.bas" في بيئة BASCOM-AVR:

```
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 4800
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
'--->[Main Program]
Do
   Print "Hello World" : Waitms 50
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

سيقوم البرنامج بإرسال (TXD) العبارة "Hello World" كل 50 ميلي ثانية على النافذة التسلسلية (UART) بشكل مستمر. على الطرف الآخر سيكون المستقبل (Phototransistor) موصل مع منفذ USB من خلال الموديول neXus وبالتالي يمكن عرض القيم المستقبلة من خلال النافذة Terminal – الشكل23.




الشكل23 خرج دارة الاستقبال في نافذة Terminal في الحاسب

في حال طلب استقبال البيانات من خلال متحكم آخر بدل الحاسب وعرضها على شاشة إظهار LCD فيتم ذلك بتوصيل خرج دارة الاستقبال إلى القطب RXD للمتحكم الآخر وسيكون البرنامج هو نفسه البرنامج "Exp.20.bas".

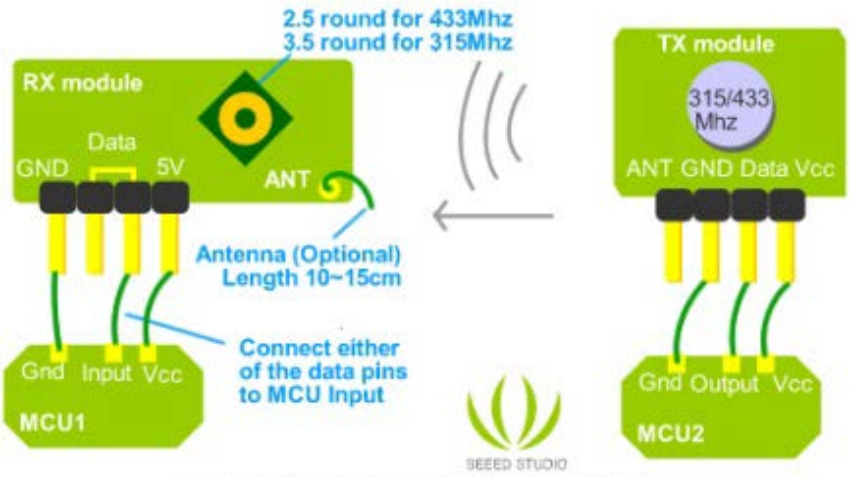
بالنتيجة فإن مشروع إرسال واستقبال البيانات باستخدام الأشعة تحت الحمراء مشابه تماماً لمشروع إرسال واستقبال البيانات باستخدام الليزر والاختلاف الوحيد هو باستبدال مرسل الأشعة تحت الحمراء بمرسل ليزري واستبدال الثنائي الضوئي بترانزستور ضوئي.

## 13-8 تطبيق: إرسال البيانات بين متحكم AVR والحاسب باستخدام الأمواج الراديوية (RF Data Link):

إن عملية إرسال البيانات بين متحكم AVR من خلال النافذة UART (TTL) والحاسب عبر منفذ USB (Differential) ستتم في هذا التطبيق باستخدام الأمواج الراديوية RF، وبالتالي سيتضمن التطبيق دارتين:

1) **دارة الإرسال للأمواج الراديوية (RF Data Sender)**: وهي عبارة عن مرسل راديوي (RF Transmitter) على شكل موديول جاهز يعمل بجهد 5V ويملك أربعة اقطاب (+5V, GND, ANT, DI). تم توصيل مدخل البيانات لموديول الإرسال (DI) إلى قطب الإرسال للنافذة التسلسلية UART للمتحكم. وبالتالي فإن جميع البيانات الصادرة من النافذة على القطب TXD سوف ترسل على شكل ثنائي (0,1) على قطب إرسال البيانات للموديول RF.

2) **دارة الاستقبال للأمواج الراديوية (RF Data Receiver)**: وهي عبارة عن مستقبل راديوي (RF Receiver) على شكل موديول جاهز يعمل بجهد 5V ويملك أربعة اقطاب (+5V, GND, ANT, DO).. تم توصيل مخرج البيانات المستقبلة (DO) لموديول الاستقبال إلى قطب الاستقبال للنافذة التسلسلية UART للموديول neXus .الشكل22 يبين مخطط دارة الاستقبال وتوصيلها مع القطب RXD لموديول الوصل مع الحاسب neXus.




الشكل24 مخطط التوصيل (Schematic) لدارة الإرسال والاستقبال بالأمواج الراديوية – الDatasheet مرفقة في مجلد المشروع

البرنامج هو نفسه البرنامج "Exp19.bas"، وبالنتيجة فإن مشروع إرسال واستقبال البيانات باستخدام الأمواج الراديوية مشابه تماماً لمشروع إرسال واستقبال البيانات باستخدام الأشعة تحت الحمراء والاختلاف الوحيد هو باستبدال مرسل الأشعة تحت الحمراء بمرسل راديوي واستبدال الثنائي الضوئي بمستقبل راديوي. وفي حال طلب استقبال البيانات من خلال متحكم آخر بدل الحاسب وعرضها على شاشة إظهار LCD فيتم ذلك بتوصيل خرج دارة الاستقبال إلى القطب RXD للمتحكم الآخر وسيكون البرنامج هو نفسه البرنامج "Exp.20.bas".

## 14-8 تطبيق: ربط موديول GPS مع متحكم AVR من خلال النافذة UART:

يستخدم هذا الجهاز لتحديد موقع أي نقطة على الأرض من حلال مجموعة من الحسابات على البيانات المستقبلة من الأقمار الصناعية وقد تقدم أثناء عرض المحاضرة الثامنة (Session_08_CE_2012.wmv) مبدأ عمل نظام الGPS وكذلك حزم البيانات التي يتم بثها من الأقمار الصناعية المخصصة لنظام الملاحة العالمي وكيف يتم استقبالها من خلال موديول استقبال "GPS Receiver Module".

```
$GPGGA,092750.000,5321.6802,N,00630.3372,W,1,8,1.03,61.7,M,55.2,M,,*76
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,19,13,28,070,17,26,23,252,,04,14,186,14*79
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092750.000,A,5321.6802,N,00630.3372,W,0.02,31.66,280511,,,A*43
$GPGGA,092751.000,5321.6802,N,00630.3371,W,1,8,1.03,61.7,M,55.3,M,,*75
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,16,13,28,070,17,26,23,252,,04,14,186,15*77
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092751.000,A,5321.6802,N,00630.3371,W,0.06,31.66,280511,,,A*45
```

(reference: http://www.gpsinformation.org/dale/nmea.htm)

الشكل25 مثال عن الحزم المستقبلة على خرج موديول GPS.

في هذا التطبيق سوف نتعامل مع إحدى حزم البيانات المستقبلة من خلال الموديول وهي الحزمة $GPZDA وهي تحوي على الوقت والتاريخ فقط؛ وبالتالي من أجل استخلاص البيانات من الحزمة فإنه يجب معرفة نوع وشكل البيانات التي يستقبلها موديول الـG.P.S والتي تعتمد البروتوكول NMEA (National Marine Electronics Association) الذي يعد أشهر برتوكولات هذا النظام.

### الحزمة ZDA - Time and Date:

شكل الحزمة هو: **$GPZDA,hhmmss.ss,DD,MM,YYYY,ltzh,ltzn*cs<CR><LF>**

| Name | ASCII String | | Units | Description | |
|---|---|---|---|---|---|
| | Format | Example | | | |
| $GPZDA | string | $GPZDA | | Message ID | ZDA Protocol header |
| hhmmss.ss | hhmmss.ss | 082710.00 | | UTC time | hours, minutes, seconds, seconds |
| day | dd | 16 | day | UTC time: day | 01 ... 31 |
| month | mm | 09 | month | UTC time: month | 01 ... 12 |
| year | yyyy | 2002 | year | UTC time: year | 4 digit year |
| ltzh | xx or -xx | 00 | | Local zone hours | Not supported (fixed to 00) |
| ltzn | zz | 00 | | Local zone minutes | Not supported (fixed to 00) |
| cs | hexadecimal | *64 | | Checksum | |
| <CR> <LF> | | | | | End of message |

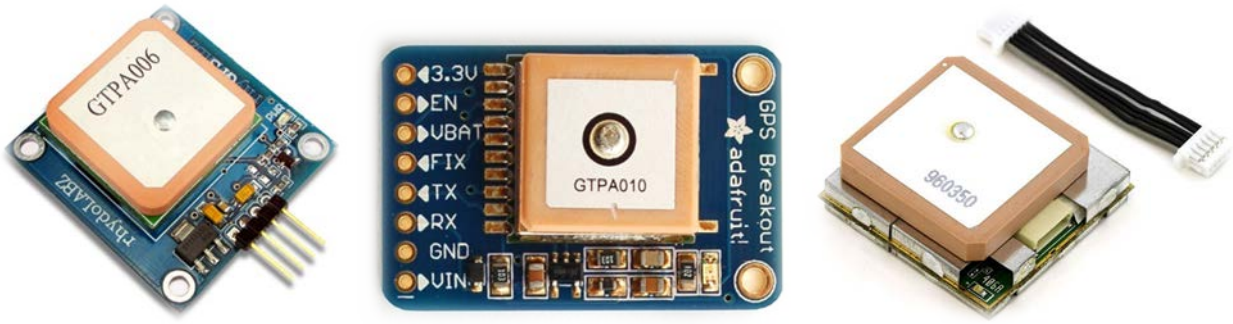
مثال: **$GPZDA,071802.00,29,10,2008,00,00*6A**

الحزمة في المثال تشير إلى أن التاريخ هو: 29/10/2008 والوقت هو: 07:18:02 بتوقيت غرينتش.

## ربط موديول GPS مع المتحكم المصغر:

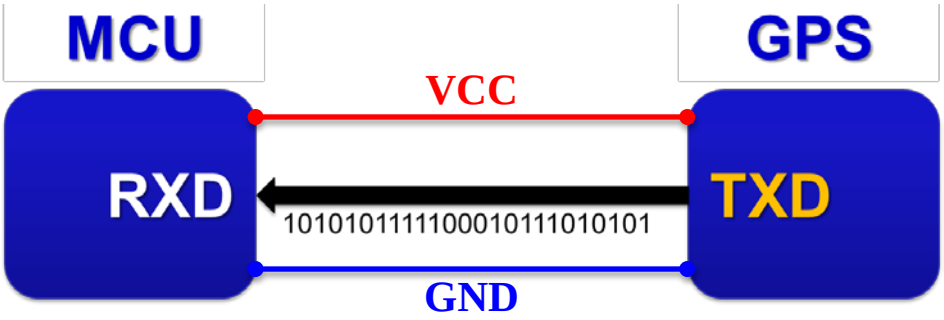
يتوفر تجارياً العديد من موديولات GPS (\$20 ~ \$50) وجميع موديولات الGPS تملك نافذة تسلسلية UART. بشكل عام يمتلك الموديول من 4~6 أقطاب لها الوظائف التالية:

- القطب VCC: قطب التغذية الرقمية للموديول (+3V ~ +5V).
- القطب GND: قطب النقطة الأرضية للتغذية.
- القطب TXD: قطب خرج البيانات المستقبلة من قبل الموديول ( يجب أن يوصل مع القطب RXD للمتحكم المصغر).
- القطب RXD: قطب دخل من أجل ضبط بارامترات الموديول.
- القطب PPS: قطب توليد نبضة تزامن بدور 1sec.




الشكل26 بعض موديولات الGPS التجارية

من أجل هذا التطبيق فإنه يكفي تغذية الموديول (VCC; GND) ووصل القطب TXD من الموديول مع القطب RXD للنافذة التسلسلية UART للمتحكم المصغر كما في الشكل27.




البرنامج "GPS_ZDA.bas" في بيئة BASCOM-AVR:

```
' ********************************************************************************
' * Title          : GPS_ZDA.bas                                                 *
' * Target Board   : Mini-Phoenix - REV 1.00                                     *
' * Target MCU     : ATMega32A                                                   *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : LCD - GPS - LED - Buzzer                                    *
' * Description    : Acquiring Time/Date/Coordinates from GPS Module             *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 4800
'-----------------------
'-----------------------[LCD Configurations]
Config Lcdpin = Pin , Db4 = Portc.2 , Db5 = Portc.3 , Db6 = Portc.4 , Db7 =
Portc.5 , E = Portd.3 , Rs = Portd.4
Config Lcd = 16 * 2
'-----------------------
'-----------------------[Variables]
Dim Uart_var As Byte , Pps_f As Bit
Dim Temp_str As String * 2, Identifier As String * 6, Data_stream As String * 27
Dim Hour_val As String * 2 , Min_val As String * 2 , Sec_val As String * 2
Dim Day_val As String * 2 , Month_val As String * 2 , Year_val As String * 2
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   If Ischarwaiting() = 1 Then Gosub Gps_isr

   If Pps_f = 1 Then
      Reset Pps_f : Cls
      Locate 1 , 1 : Lcd "Time: " ; Hour_val ; ":" ; Min_val ; ":" ; Sec_val
      Locate 2 , 1 : Lcd "Date: " ; Day_val ; "/" ; Month_val ; "/20" ; Year_val
   End If
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[UART]
Gps_isr:
   Uart_var = Inkey()
   If Uart_var = "$" Then
      $timeout = 100000 : Input Data_stream
      Identifier = Mid(data_stream , 1 , 6)
      If Identifier = "GPZDA," Then
         '->[Time]
         Hour_val = Mid(data_stream , 07 , 2)
         Min_val = Mid(data_stream , 09 , 2)
         Sec_val = Mid(data_stream , 11 , 2)
         '->[Date]
         Day_val = Mid(data_stream , 17 , 2)
         Month_val = Mid(data_stream , 20 , 2)
         Year_val = Mid(data_stream , 25 , 2)
         Set Pps_f
      End If
   End If
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

إن جميع حزم البيانات الواردة على خرج موديول GPS تبدأ بالمحرف "$" وبالتالي فإن البرنامج "GPS_ZDA.bas" سيقوم بمايلي:

1) أولاً بانتظار ورود بيانات على النافذة التسلسلية UART حتى يتحقق الشرط (**If Iscarwaiting()=1**).
2) سوف يقوم بقراءة المحرف الوارد على النافذة (Uart_var = **Inkey**()) والتأكد فيما إذا كان المحرف هو "$" (**If** Uart_var = "$" **Then**) إشارة إلى بداية حزمة بيانات.
3) في حال كانت بداية حزمة بيانات ("$") فسيتم قراءة كامل الحزمة (**Input** Data_stream) إلى سلسلة محرفية "Data_stream" والذي تم تعريفه "Data_stream **As String** * 27" بـ27 محرف ممثلة بالمتحول "Data_stream".
4) سنحتاج الآن إلى التأكد من أن الحزمة التي تم وضعها في المتحول "Data_stream" هي حزمة البيانات "GPZDA" المطلوبة. لذلك سيتم استخدام تعليمة الاقتطاع من سلسلة محرفية (**Mid**) من أجل اقتطاع المحارف الستة الأولى وفحصها للتأكد فيما إذا كانت هي للحزمة "GPZDA".

شكل التعليمة هو: `String_var = Mid(String , Start , Num_of_char)`

حيث أن المتحول "String_var" هو الذي سيتم فيه وضع المحارف المقتطعة من السلسلة ويجب أن يكون حجمه معرفاً بحيث يتسع للمحارف المطلوب اقتطاعها. المتحول "String" هو السلسة المحرفية الأصلية المطلوب أن يتم الاقتطاع منها. المتحول "Start" هو نقطة بداية الاقتطاع. المتحول "Num_of_char" هو عدد المحارف المطلوب اقتطاعها.

5) في حال كانت المحارف الستة الأولى المقتطعة من السلسلة المحرفية هي للحزمة المطلوبة (GPZDA,)، فعندها يتم إكمال عملية تجزيئ السلسلة المحرفية من أجل الحصول على البيانات المطلوبة وهي الوقت والتاريخ حيث أن لكل قيمة موضع محدد في السلسلة المحرفية كما هو مبين أدناه بين الأقواس...

```
'$GPZDA,hhmmss.ss,DD,MM,YYYY,00,00*cs<CR><LF>
'hh(7,8) : mm(9,10) : ss(11,12) : DD(17,18) : MM(20,21) : YYYY(xx,xx,25,26)
```

6) المتحول "Pps_f" يستخدم كعلم من أجل عرض القيم الجديدة كلما توفرت على شاشة الإظهار LCD.

**ملاحظة هامة**: تمتلك الحزمة GPZDA مواضع ثابتة للمحارف ضمن السلسلة، أي: قيمة الثواني تتوضع دائماً في السلسلة عند المحرفين 11,12 (SS) وقيمة الشهر تتوضع عند المحرفين 20,21 (MM) وهكذا... إلا أن بعض الحزم الأخرى وأهمها الحزمة GPRMC لا تمتلك مواضع ثابتة للمحارف إذا يمكن أن تتغير تبعاً لعدد القيم بعد الفاصلة العشرية لبعض متحولات خطوط الطول والعرض. من أجل ذلك سنضع هنا فكرة برمجية من أجل استخلاص قيم الإحداثيات والوقت والتاريخ والارتفاع والسرعة من الحزمة GPRMS فيما يلي.

الحزمة RMC (RMC - Recommended Minimum Data):

شكل الحزمة هو:

**$GPRMC,hhmmss.000,status,latitude,N,longitude,E,spd,cog,ddmmyy,mv,mve,mode*cs<CR><LF>**

| Name | ASCII String | | Description | |
|---|---|---|---|---|
| | Format | Example | | |
| $GPRMC | string | $GPRMC | Message ID | RMC protocol header |
| hhmmss | hhmmss.sss | 083559.00 | UTC Time | Time of position fix |
| status | character | A | Status | V = Navigation receiver warning<br>A = Data valid. |
| latitude | ddmm.mmmm | 4717.11437 | Latitude | User datum latitude degrees, minutes, minutes |
| N | | N | N/S Indicator | N=north or S=south |
| longitude | ddmm.mmmm | 00833.91522 | Longitude | User datum latitude degrees, minutes, minutes |
| E | character | E | E/W indicator | E=east or W=west |
| Spd | numeric | 0.004 | Speed (knots) | Speed Over Ground |
| cog | numeric | 77.52 | COG (degrees) | Course Over Ground |
| ddmmyy | ddmmyy | 091202 | Date | Current Date in Day, Month Year |
| mv | numeric | | Magnetic variation | Not being output by receiver |
| mvE | character | | Magnetic variation E/W indicator | Not being output by receiver |
| mode | | | Mode Indicator | See Position Fix Flags in NMEA Mode |
| cs | hexadecimal | *53 | Checksum | |
| <CR> <LF> | | | | End of message |

مثال عن الحزمة:

$GPRMC,071802.00,A,4717.11437,N,00833.91522,E,0.004,77.52,14072011,,,A*57

البرنامج "GPS_RMC.bas" في بيئة BASCOM-AVR:

```
' ********************************************************************************
' * Title          : GPS_RMC.bas                                                 *
' * Target Board   : Mini-Phoenix - REV 1.00                                     *
' * Target MCU     : ATMega32A                                                   *
' * Author         : Walid Balid                                                 *
' * IDE            : BASCOM AVR 2.0.7.3                                          *
' * Peripherals    : LCD - GPS - LED - Buzzer                                    *
' * Description    : Acquiring Time/Date/Coordinates from GPS Module             *
' ********************************************************************************
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'-----------------------[Definitions]
$regfile = "m32def.dat"
$crystal = 8000000
$baud = 4800
'-----------------------
'-----------------------[LCD Configurations]
Config Lcdpin = Pin , Db4 = Portc.2 , Db5 = Portc.3 , Db6 = Portc.4 , Db7 =
Portc.5 , E = Portd.3 , Rs = Portd.4
Config Lcd = 16 * 2
'-----------------------
'-----------------------[Variables]
Dim Pps_f As Bit , Uart_byte As Byte , Pos(9) As Byte , J As Byte , I As Byte
Dim Hour_val As String * 2 , Min_val As String * 2 , Sec_val As String * 2
```

```
Dim Day_val As String * 2 , Month_val As String * 2 , Year_val As String * 2
Dim Identifier As String * 6 , Data_stream As String * 66 , Pos_i As Byte
Dim Latitude As String * 10 , Longitude As String * 11
Dim N_s As String * 1 , E_w As String * 1
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[Main Program]
Do
   If Ischarwaiting() = 1 Then Gosub Gps_isr

   If Pps_f = 1 Then
      Reset Pps_f : Cls
      Locate 1 , 1 : Lcd "Time: " ; Hour_val ; ":" ; Min_val ; ":" ; Sec_val
      Locate 2 , 1 : Lcd "Date: " ; Day_val ; "/" ; Month_val ; "/20" ; Year_val
      Locate 3 , 1 : Lcd Latitude ; "  - " ; N_s
      Locate 4 , 1 : Lcd Longitude ; " - " ; E_w
   End If
Loop
End
'---<[End Main]
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'--->[UART]
Gps_isr:
   Uart_byte = Inkey()
   If Uart_byte = "$" Then
      $timeout = 100000 : Input Data_stream
      Identifier = Mid(data_stream , 1 , 6)
      If Identifier = "GPRMC," Then
         '->[Looking for ',' Positions]
         J = 1
         For I = 1 To 9
            Pos(i) = Charpos(data_stream , "," , J)
            J = Pos(i)
         Next I
         '->[Time]
         Pos_i = Pos(1) + 1 : Hour_val = Mid(data_stream , Pos_i , 2)
         Pos_i = Pos(1) + 3 : Min_val = Mid(data_stream , Pos_i , 2)
         Pos_i = Pos(1) + 5 : Sec_val = Mid(data_stream , Pos_i , 2)
         '->[Date]
         Pos_i = Pos(9) + 1 : Day_val = Mid(data_stream , Pos_i , 2)
         Pos_i = Pos(9) + 3 : Month_val = Mid(data_stream , Pos_i , 2)
         Pos_i = Pos(9) + 5 : Year_val = Mid(data_stream , Pos_i , 2)
         '->[Location]
         Pos_i = Pos(3) + 1 : Latitude = Mid(data_stream , Pos_i , 9)
         Pos_i = Pos(4) + 1 : N_s = Mid(data_stream , Pos_i , 1)
         Pos_i = Pos(5) + 1 : Longitude = Mid(data_stream , Pos_i , 10)
         Pos_i = Pos(6) + 1 : E_w = Mid(data_stream , Pos_i , 1)
         Set Pps_f
      End If
   End If
Return
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

البرنامج "GPS_RMC.bas" يعتمد نفس المبدأ في البرنامج "GPS_ZDA.bas"، إلا أننا هنا لا نعتبر موقع المحارف ثابت وإنما نوجد مواقع الفاصلة "," التي تفصل بين البيانات ونوضح هذا فيما يلي:

**$GPRMC,071802.00,A,4717.11437,N,00833.91522,E,0.004,77.52,14072011,,,A*57**

بالنظر إلى الحزمة السابقة فإننا سنجد أن مواقع الفواصل "," هي: [7, 17, 19, 30, 32, 44, 46, 52, 58, 67, ...]. ومن الواضح تماماً أنه بعد الفاصلة الأولى يأتي قيمة الوقت (071802) وبعد الفاصلة الثالثة يأتي قيمة خط الطول (4717.11437) وبعد الفاصلة الرابعة تأتي قيمة محدد الاتجاه (N) وبعد الفاصلة الخامسة يأتي قيمة خط العرض (00833.91522) وبعد الفاصلة السادسة تأتي قيمة محدد الاتجاه (E) وبعد الفاصلة السابعة تأتي قيمة السرعة (0.004) وبعد الفاصلة التاسعة تأتي قيمة التاريخ (14072011)...

وبالتالي تمكنا من معرفة بدايات توضع كل صنف من البيانات والآن يمكننا اقتطاعها ابتداءً من هذا العنوان وانتهاءً بعنوان الفاصلة التالية. ويتم تحديد مواقع الفواصل من خلال تعليمة البحث عن موضع محرف ضمن سلسلة محرفية المتمثلة بالتعليمة "**Charpos**". يمكن الاطلاع على بارامترات التعليمة في برنامج BASCOM-AVR/Help.

... ﴿انتهت الجلسة العملية الثامنة والأخيرة﴾ ...

- تمنياتي لكم بمستقبل مشرق وحياة طيبة كريمة هانئة -

وليد بليد