دورة برمجة متحكمات شركة أتمبل

(ATMEL AVR)

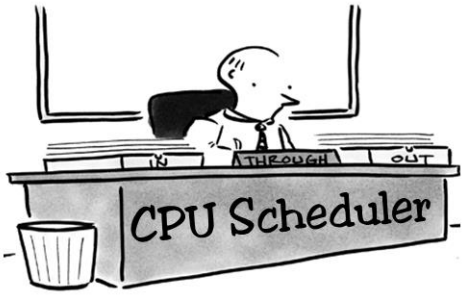
ATMEL University

Micromir

https://www.facebook.com/Micromir.Electronics

إعداد المهندس

تميم محمد ناهل الأمير



الجلسة الأولى

مقدمة :

أضحت المتحكمات الصغرية (Microcontrollers) في الوقت الراهن القلب النابض لمعظم النظم الالكترونية المضمَّنة (Electronic Embedded Systems) والتي أصبحت جزءاً لا يتجزأ من حياتنا اليومية ابتداءاً من الهواتف النقالة وانتهاءاً بالسيارات والأجهزة الكهربائية المنزلية , ومع ازدياد تعقيد هذه النظم الالكترونية ازدادت الحاجة إلى متحكمات حديثة بوظائف فائقة وسرعة معالجة عالية مما أوجب على شركات التصنيع السباق فيما بينها لإنتاج المتحكم الأفضل بالسعر الأنسب وهذا ما سبب طفرة هائلة في الانتاج رافق الثورة التكنولوجيا التي تتجدد كل يوم لتأتي بمنتجات ونظم لم تكن لتخطر ببال بشر قبل عقد أو عقدين من الزمن .

بشكلٍ عام يمكن اعتبار المتحكم الصغري كجهاز حاسب كامل موضوع ضمن شريحة الكترونية في دارة متكاملة (IC) وهو قادر على تنفيذ كافة الأعمال المطلوبة منه بحسب تعليمات البرنامج المخزن داخل ذاكرته وذلك على مستوى الأنظمة الالكترونية الصغيرة والمتوسطة , وبالتالي فإن البرنامج المكتوب داخل المتحكم يحدد إلى حد كبير مستوى تعقيد النظام المقاد من خلاله فهو أساس عمل النظام بشكل سليم ومتكامل. وحتى نستطيع كتابة برامج متقدمة للمتحكمات الصغرية علينا أن نلم بشكل كامل بمحتويات المتحكم من وحدات طرفية وذواكر وموقتات من جهة وأن نقوم بدراسة تعليمات لغة البرمجة المراد برمجة المتحكم بها من جهة أخرى وهذا ما نحن بصدده في هذا الكورس التعليمي بمشيئة الله تعالى .

المتحكمات الصغرية (Microcontrollers) :

المتحكم الصغري عبارة عن شريحة الكترونية صغيرة جداً مصنوعة بتقنية تصنيع عالية جداً وموضوعة ضمن دارة متكاملة (IC) تُستخدم لقيادة أنظمة التحكم وتعمل وفقاً للبرنامج المخزن داخلها .

هناك العديد من شركات التصنيع المنتجة للمتحكمات الصغرية في العالم من أشهر هذا الشركات شركة (ATMEL) وشركة (Microship) صاحبة متحكمات (PIC) وشركة (Intel) صاحبة متحكمات (8051) وشركة (PHILIPS) صاحبة متحكمات (NXP) وشركة (TOSHIBA) والعديد من الشركات الأخرى حول العالم التي تختص بمجال تصنيع الدارات المتكاملة والمعالجات والمتحكمات الصغرية .

وفيما يلي جدول يبين بعض أوجه المقارنة الرئيسية لمتحكمات (AVR) و (PIC) و (8051) على اعتبار أن هذه المتحكمات من أكثر الماركات انتشاراً في السوق السورية :

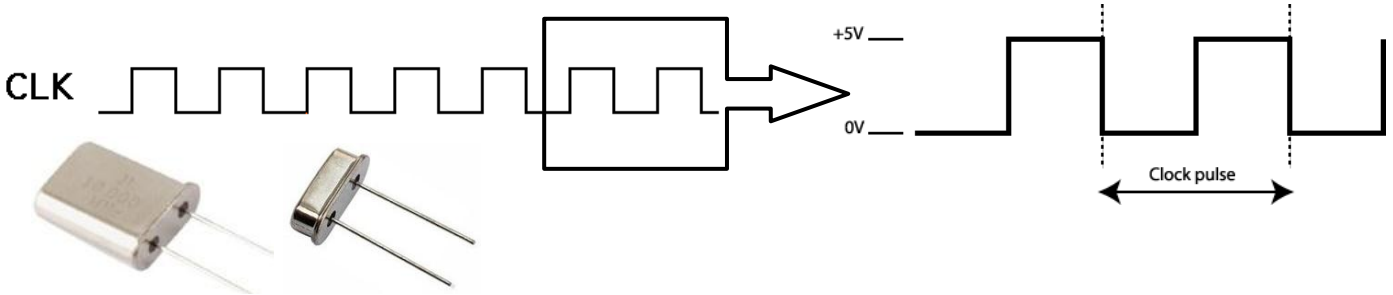
وجه المقارنة	ATMEL (AVR)	Microchip (PIC)	Intel (8051)
التردد الأعظمي لأعلى معالج	16 MHz	20 MHz	24 MHz
عدد النبضات اللازمة لكل تعليمة	1	4	12
عدد تعليمات لغة التجميع	132	35	215
بنية المعالج التصميمية	RISC	RISC	CISC

مبدأ عمل المتحكم الصغري :

بشكل مجرد وبسيط .... يعمل المتحكم الصغري وفق ما يلي :

يقوم المبرمج ببرمجة المتحكم بالبرنامج المطلوب حيث يتم تخزين تعليمات هذا البرنامج في ذاكرة البرنامج داخل المتحكم (وهي ذاكرة دائمة الحفظ للمعلومات من النوع (Flash Memory)) .

الآن لدى وصل تغذية مناسبة للمتحكم يقوم هزاز كريستالي (داخلي أو خارجي) بتوفير نبضات ساعة (CLK) لمعالج المتحكم حيث يقوم هذا المعالج بتنظيم جميع عملياته وفق هذه النبضات المنتظمة فمثلاً يقوم بالنبضة الأولى بتنفيذ التعليمة الأولى من البرنامج ومن ثم في النبضة الثانية يقوم بتنفيذ التعليمة الثانية وهكذا ..... (طبعاً هذا في متحكمات (AVR)) .

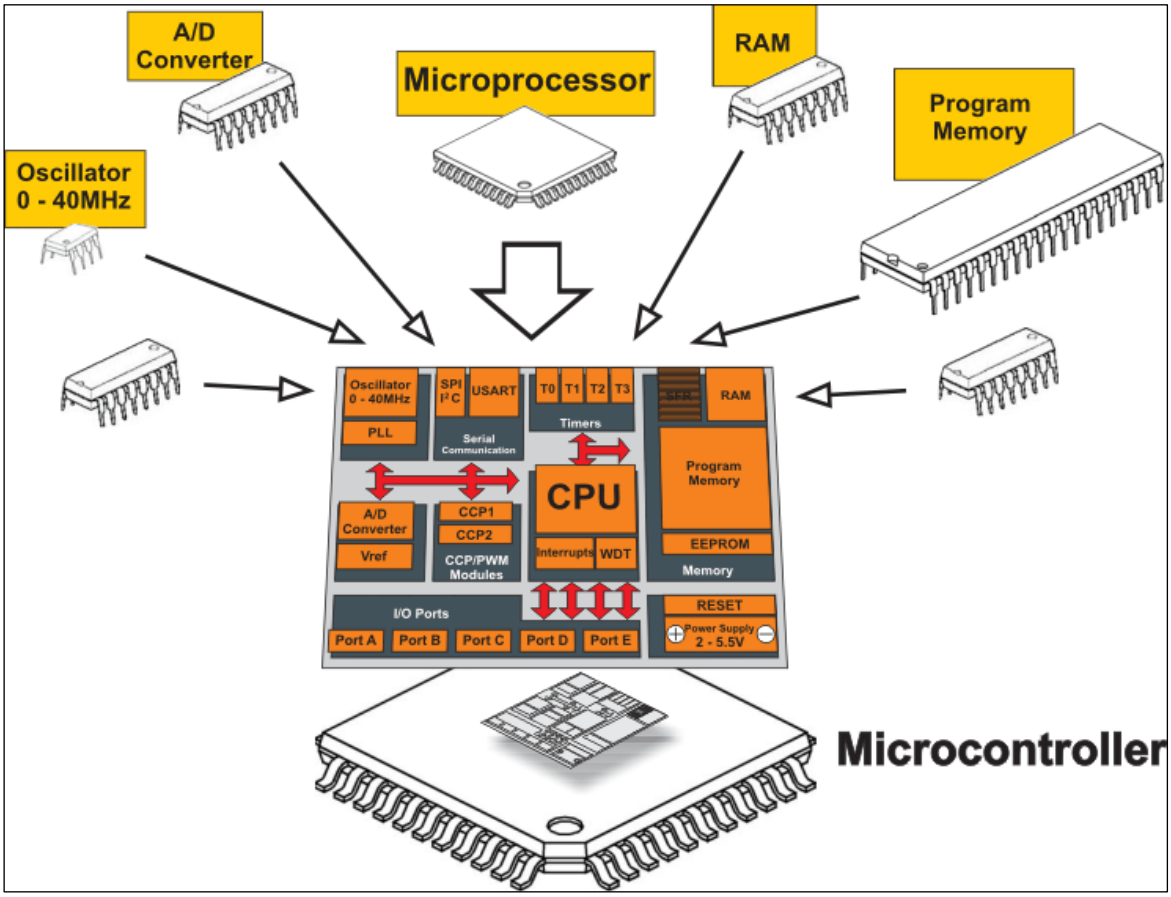


يكتب المبرمج برنامجه على أحد المترجمات (Compilers) الخاصة بالمتحكم المطلوب , ويقوم بعدها عن طريق هذا المترجم بتحويل البرنامج من لغة البرمجة عالية المستوى التي كُتب بها (مثلاً : Pascal ....etc , C++ , QBASIC) إلى ملف مشفّر بالشيفرة الست عشرية (Hexadecimal) وهو الملف الذي يتم نقله إلى ذاكرة البرنامج في المتحكم وذلك عن طريق مبرمجة مناسبة .

المتحكمات والمعالجات الصغرية (Microcontrollers & Microprocessors) :

سبق المعالج الصغري المتحكم بالظهور عقود عديدة وعلى الرغم من التشابه بالشكل الخارجي بين الاثنين إلا أن المتحكم يختلف بالمضمون اختلافاً كلياً , حيث أن كل متحكم صغري يحتوي بداخله معالج صغري بينما العكس غير صحيح , فالمتحكم قادر على العمل في دارات التحكم بشكل عام دون وحدات إضافية أخرى معه بعكس المعالج الصغري الذي لا يمكنه العمل دون وصلِه مع وحدات طرفية (وحدات دخل/خرج , وحدات توقيت , وحدات اتصال تسلسلي ..... الخ) ليتمكن من التفاعل مع عناصر دارة التحكم من حوله.

يُرمز عادة للمتحكم الصغري في المخططات الالكترونية بالرمز (MCU or μC or uC) وللمعالج الصغري بالرمز (MPU or μP or uP) .



متحكمات شركة (ATMEL) من النوع (AVR 8-bit)

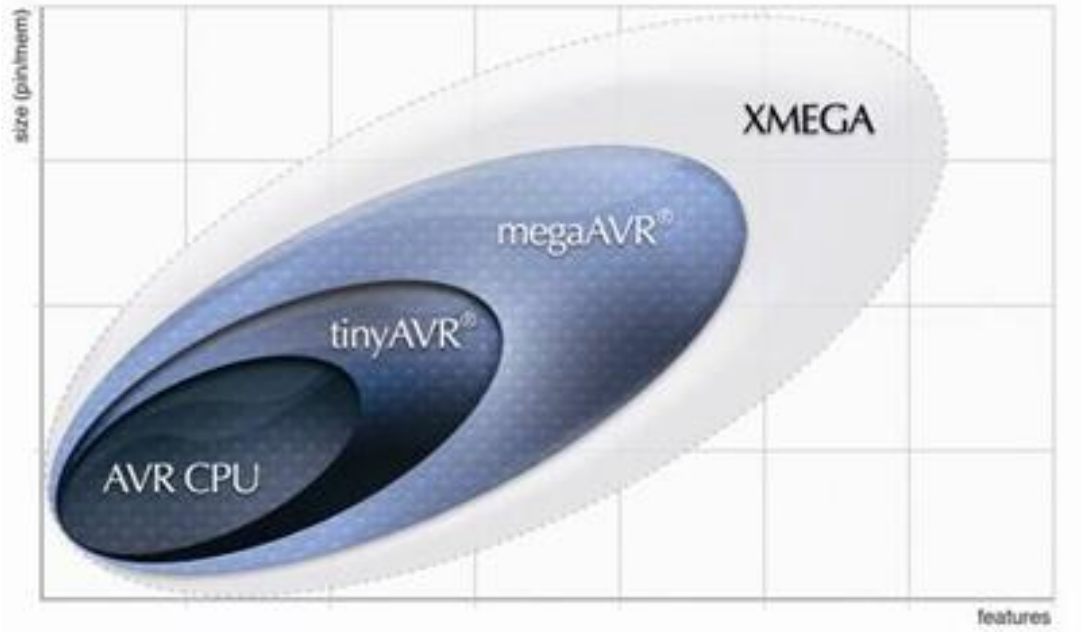
تعد شركة (ATMEL) الأمريكية واحدة من أعرق شركات تصنيع أنصاف النواقل في العالم , إذ أنها بدأت إنتاجها عام (1984) حينما كانت تصنع ذواكر (EPROMs) و الأجهزة المنطقية القابلة للبرمجة (PLDs) . ومن ثم تابعت تطورها في ظل ثورة الصناعة السيليكونية في العالم لتصل إلى ما وصلت إليه من مكانة مرموقة اليوم في هذه الصناعة . فهي تنتج المتحكمات الصغرية (Microcontrollers) وذواكر الـ(Flash) و الدارات المتكاملة (Integrated Circuits) و شاشات اللمس (Touch Screen) وطيف واسع من أنواع الحساسات الرقمية والتشابهية مما جعلها موضع ثقة كبرى شركات تصنيع الالكترونيات في العالم كشركة (Sony) وشركة (LG) وشركة (Samsung) ..... وغيرها .

تنتج شركة (ATMEL) أنواع عديدة من المتحكمات الصغرية (8-bit) بمختلف الخصائص والميزات نذكر منها ما يلي :

❖ *متحكمات (AVR) العامة :*

وهي المتحكمات التي لم تُنتَج حصرياً لغرض معين وإنما يمكن استخدامها في عمليات التحكم بشكل عام وتُنتج منها (ATMEL) أربعة أصناف :

1) AT**90S** XXXX : وهي أولى متحكمات شركة (ATMEL) من النوع (AVR) وقد تم إيقاف إنتاجها حالياً .
2) AVR **tiny** : تُنتج هذه المتحكمات بميزات منخفضة بشكل عام وبالتالي هي مناسبة للمشاريع الصغيرة .
3) AVR **MEGA** : تُنتج هذه المتحكمات بميزات متوسطة بشكل عام وبالتالي هي مناسبة للمشاريع المتوسطة .
4) AVR **XMEGA** : تُنتج هذه المتحكمات بميزات عالية وتتوفر منها متحكمات (8 or 16 bit) وبالتالي هي مناسبة للمشاريع الضخمة .



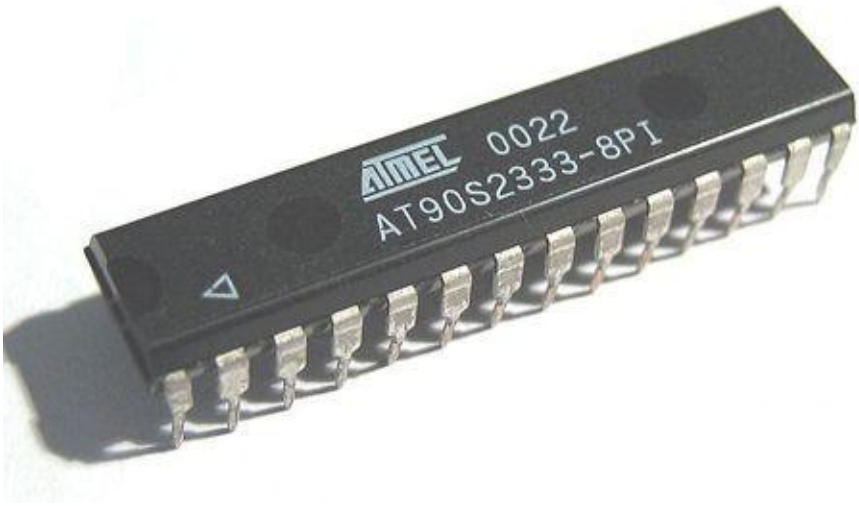
❖ *متحكمات (AVR) الخاصة :*

وهي المتحكمات التي صُنعت لتقوم بوظيفة معينة أو لتعمل في بيئة معينة بالإضافة إلى وظيفتها الأساسية كمتحكمات صغرية . وتُنتج منها (ATMEL) أصناف عديدة نذكر منها :

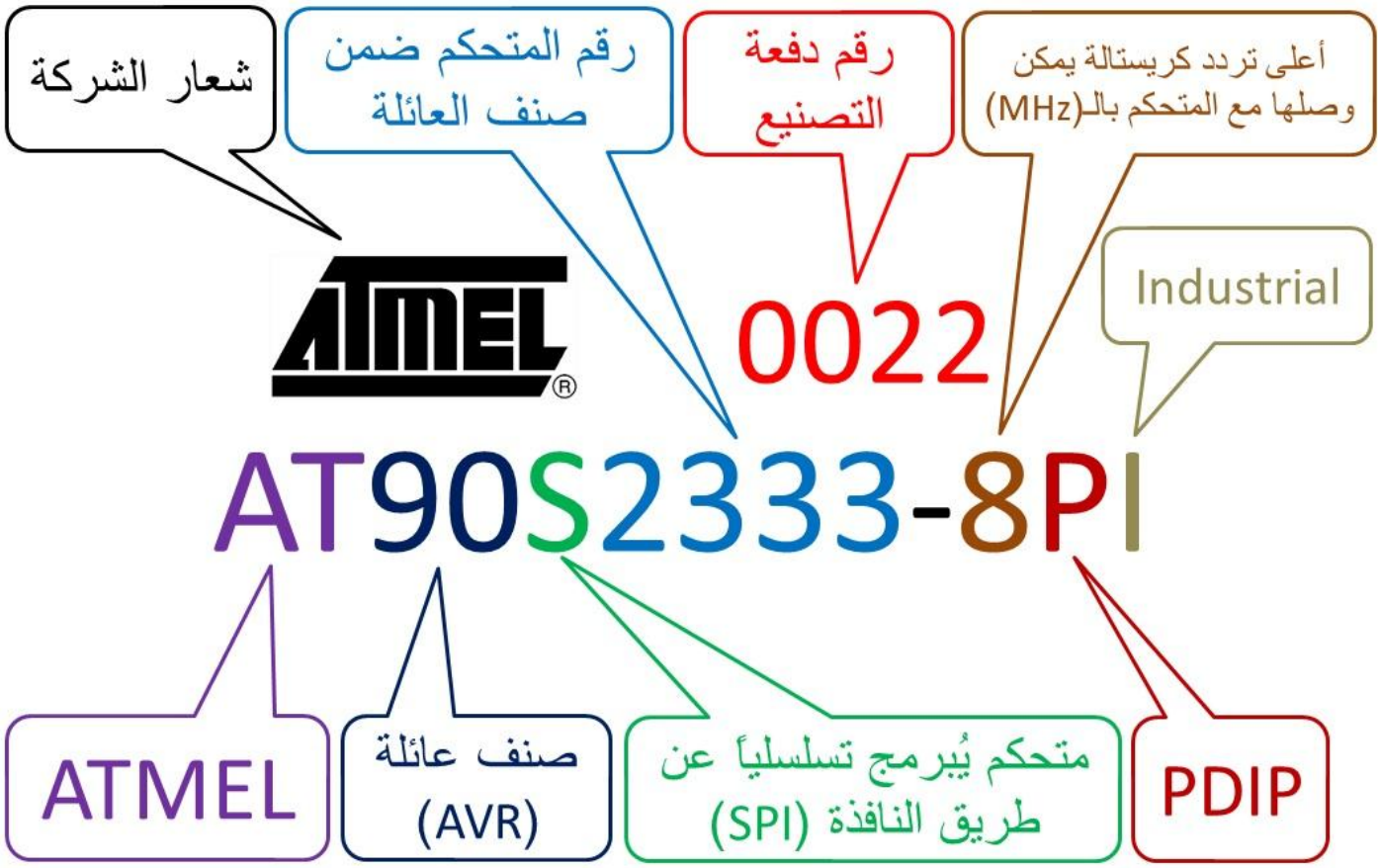
صنف المتحكم	عمله
Automotive AVR	تُستخدم في أنظمة التحكم بالسيارات
Battery Management AVR	تُستخدم في أنظمة شحن البطاريات ومراقبة جهودها
LCD AVR	تُستخدم بشكل خاص لقيادة شاشات الإظهار الكريستالية (LCD)
Lighting AVR	تُستخدم بشكل خاص للتحكم بشدة الإضاءة
USB AVR	تُستخدم بشكل خاص في التخاطب مع منفذ الـ(USB) في الحاسب

قراءة الشيفرة المكتوبة على متحكمات (ATMEL AVR) :

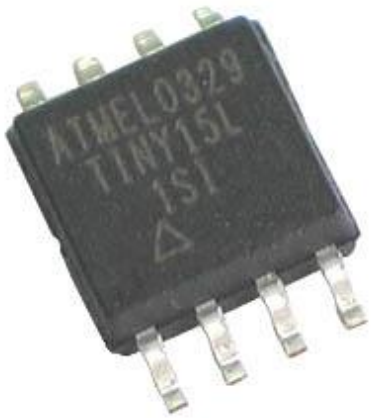
يُكتب عادة على الوجه العلوي للمتحكم مجموعة من الأرقام و الرموز تمتلك دلالات معينة عن هذا المتحكم.
وكمثال على ذلك لنأخذ المتحكم التالي ونفسر دلالات الرموز المكتوبة عليه :



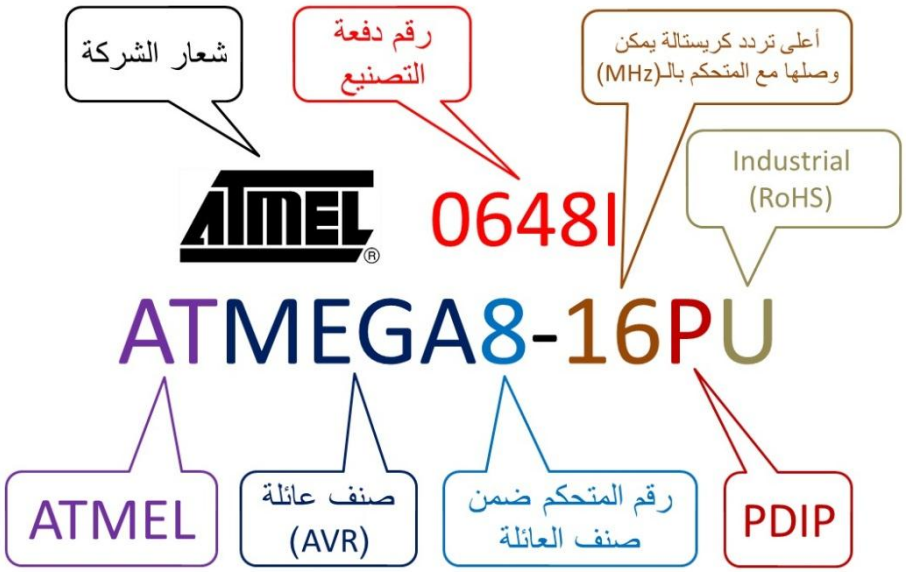
وتُفسر هذه الرموز والكتابات على الشكل التالي :



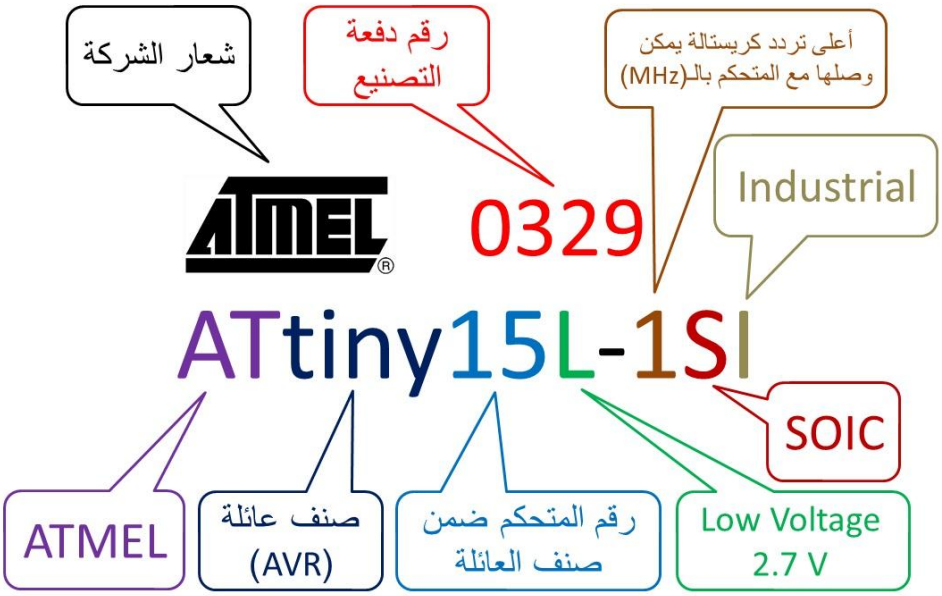
ولنأخذ أيضاً هنا المثالين التاليين :



قراءة المتحكم اليميني :



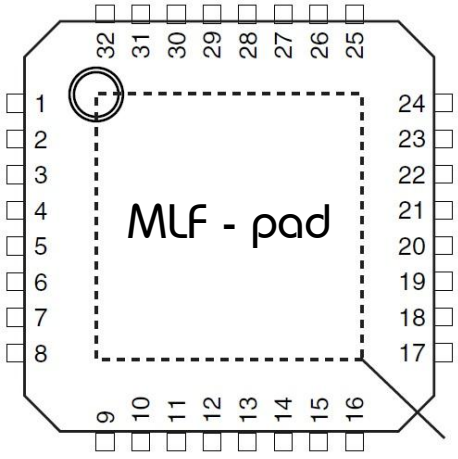
قراءة المتحكم اليساري :

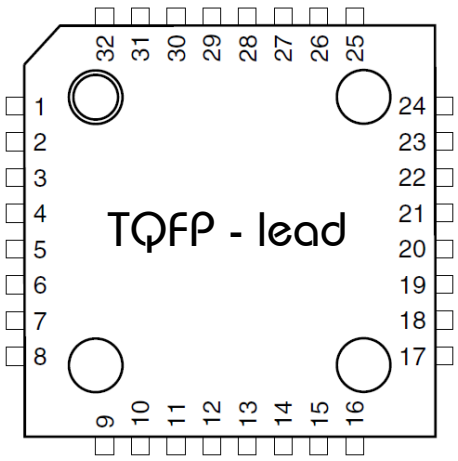


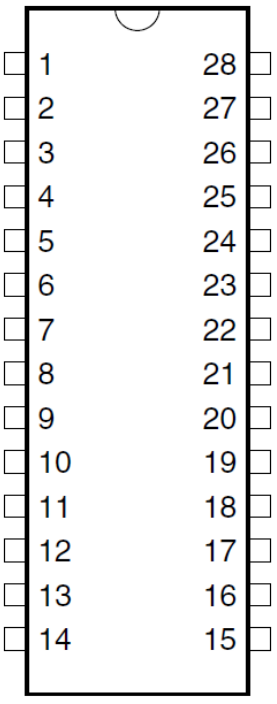
تتم قراءة الشيفرة المكتوبة على متحكمات (ATMEL AVR) كما يلي :

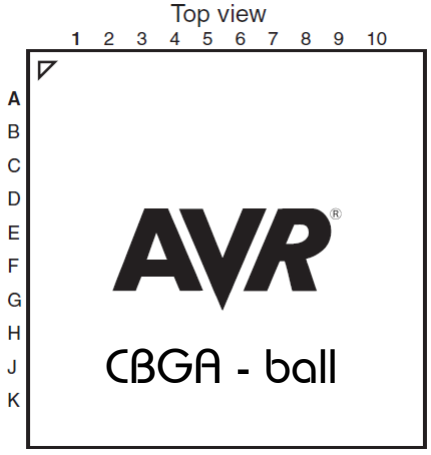
- ❖ يُشير الحرفان (AT) في البداية إلى اسم شركة (ATMEL) حيث تبدأ جميع منتجاتها من الدارات المتكاملة والمتحكمات بهذين الحرفين .
- ❖ يأتي بعده صنف عائلة (AVR) التي ينتمي إليها هذا المتحكم وهي إما أن تكون (90 or tiny or MEGA or XMEGA) كما ذكرنا ذلك سابقاً .
- ❖ يأتي بعده رقم المتحكم ضمن صنف عائلة (AVR) حيث يمتلك كل صنف عدد كبير من أرقام المتحكمات سنستعرضها لاحقاً .
- ❖ يأتي بعده رقم يُعبر عن أعلى قيمة تردد كريستالة (بالميغاهيرتز) يمكن للمتحكم أن يعمل معها .
- ❖ بعد قيمة تردد الكريستالة الأعظمي يأتي حرف يُعبر عن نوع الحافظة الخارجية للمتحكم (Package Type) ونوع أقطابه أيضاً . وهنا لدينا احتمالات كثيرة نأخذ منها الاحتمالات الأكثر شيوعاً في السوق :

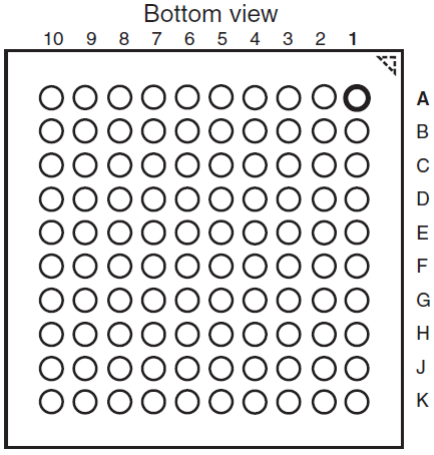
1. الحرف (P) : وهو يدل على أن الحافظة من النوع (PDIP) والأقطاب من النوع (Pin) .
2. الحرف (A) : يدل على أن الحافظة الخارجية من النوع (TQFP) والأقطاب من النوع (lead) .
3. الحرف (M) : يدل على أن الحافظة الخارجية من النوع (MLF) والأقطاب من النوع (pad) .
4. الحرف (S) : يدل على أن الحافظة الخارجية من النوع (SOIC) والأقطاب من النوع (lead) .
5. الحرف (C) : يدل على أن الحافظة الخارجية من النوع (CBGA) والأقطاب من النوع (ball) .











❖ يأتي بعده حرف يُعبر عن مجال العمل الذي خُصص المتحكم للعمل فيه , فهناك المتحكم التجاري والمتحكم الصناعي والمتحكم العسكري , وتتطابق هذه الأنواع فيما بينها بالبنية الداخلية والمواصفات إلا أنها تختلف مع بعضها البعض بدرجات حرارة الوسط الخارجي القادرة على العمل فيها ويُبيّن الجدول التالي أهم رموز مجال عمل متحكمات (AVR) :

الرمز	نوع المتحكم	درجات الحرارة التي يعمل فيها
C	متحكم مخصص للأغراض التجارية	(0 → 70) °C
I	متحكم مخصص للأغراض الصناعية	(-40 → 85) °C
M	متحكم مخصص للأغراض العسكرية	(-50 → 125) °C
U	متحكم مخصص للأغراض الصناعية مُطابق لمعيار الاتحاد الأوروبي (RoSH)	(-40 → 85) °C

❖ أخيراً يتوضع بجانب شعار شركة (ATMEL) رقم يُدعى برقم دفعة التصنيع (أو دفعة المُنتج) وهو رقم خاص بالمصنع وليس له أي دور بالنسبة للمستهلك .

بقي أن نذكر أنه قد يتم إنتاج أكثر من نوع (بالنسبة للحافظة الخارجية) للمتحكم الواحد ولمعرفة الأنواع المُنتجة من المتحكم المُراد التعامل معه نرجع إلى فقرة (Ordering Information) من الـ(Datasheet) الخاصة بالمتحكم .

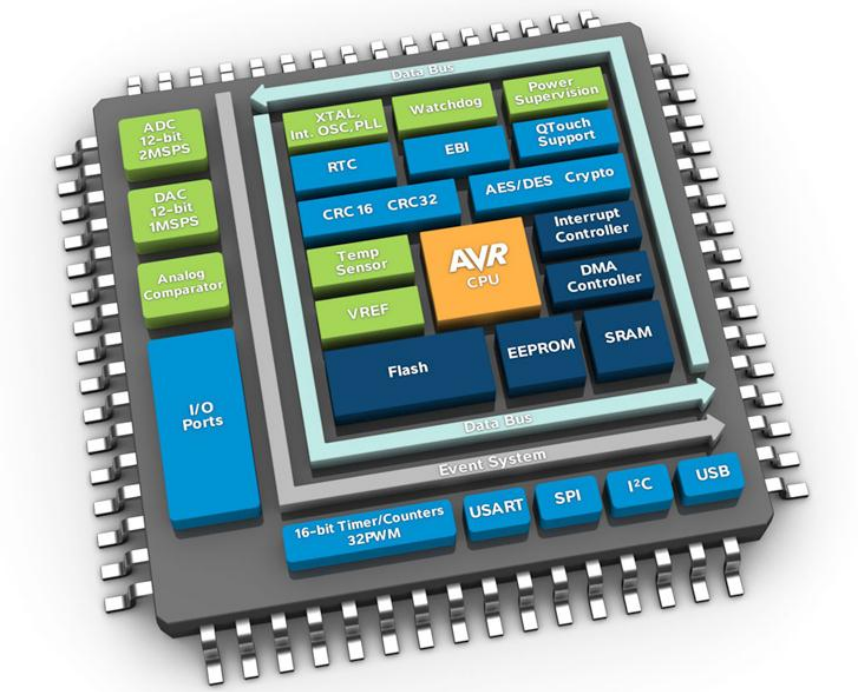
Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
8	2.7 - 5.5V	ATmega16L-8AC ATmega16L-8PC ATmega16L-8MC	44A 40P6 44M1	Commercial (0°C to 70°C)
		ATmega16L-8AI ATmega16L-8AU(1) ATmega16L-8PI ATmega16L-8PU(1) ATmega16L-8MI ATmega16L-8MU(1)	44A 44A 40P6 40P6 44M1 44M1	Industrial (-40°C to 85°C)
16	4.5 - 5.5V	ATmega16-16AC ATmega16-16PC ATmega16-16MC	44A 40P6 44M1	Commercial (0°C to 70°C)
		ATmega16-16AI ATmega16-16AU(1) ATmega16-16PI ATmega16-16PU(1) ATmega16-16MI ATmega16-16MU(1)	44A 44A 40P6 40P6 44M1 44M1	Industrial (-40°C to 85°C)

Note: 1. Pb-free packaging alternative, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

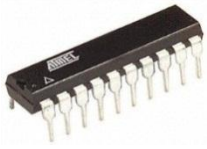
بنية متحكمات (ATMEL AVR 8-bit) :

زودت شركة (ATMEL) متحكماتها (AVR) بعدد كبير من الوحدات الداخلية نذكر منها على سبيل المثال لا الحصر : (مبدل رقمي تشابهي , مبدل تشابهي رقمي , مقارن تشابهي , وحدات دخل / خرج (I/O) , وحدات التوقيت والعد , وحدات الاتصال التسلسلي , مؤقت المراقبة , ذواكر (EEPROM) , ذواكر (Flash) , وحدة المقاطعات ..... الخ) .



تختلف الوحدات الداخلية من متحكم إلى آخر وذلك بحسب درجة تطور المتحكم . إلا أن هناك عدد من الوحدات الداخلية الأساسية المتواجدة في جميع المتحكمات بشكل عام وعدد آخر يكون موجوداً بحسب الوظيفة التي صُنع المتحكم لأجلها وذلك تبعاً لدرجة تطوره و تعقيده .

و يُبيّن الجدول التالي أشهر الأرقام التجارية لمتحكمات شركة (ATMEL) من النوع (AVR 8-bit PDIP) :

8 Pins	14 Pins	20 Pins	28 Pins	40 Pins
				
AT90S2323 AT90S2343 ATtiny11/12 ATtiny13 ATtiny15 ATtiny22 ATtiny25/45/85	ATtiny24 ATtiny44 ATtiny84	AT90S1200 AT90S2313 ATtiny2313 ATtiny26 ATtiny46 ATtiny86	AT90S2333 AT90S4433 ATtiny28 ATmega8 ATmega48 ATmega88 ATmega168	AT90S4414 AT90S4434 AT90S8515 AT90S8535 ATmega8515 ATmega8535 ATmega16 ATmega32

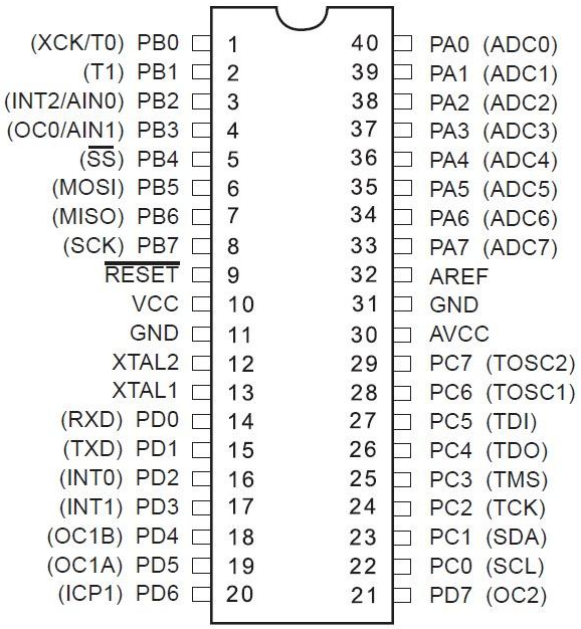
المتحكم الصغري (ATmega16)

سنقوم في هذه الفقرة بدراسة شاملة عن المتحكم الصغري (Atmega16) وما ينطبق عليه يسري على جميع متحكمات (ATMEL AVR 8-bit) على اعتبار أن البنية العامة واحدة للجميع والاختلاف بالمواصفات وبأحجام الذواكر فقط .

مواصفات المتحكم (ATmega16) :

- مُتحكم ذو أداء عالي باستهلاك طاقة منخفض مُصنّع وفق بنية (AVR 8-bit) .
- شريحة ذات (40) قطب خارجي مبنية وفق معيارية (RISC) المتقدمة في صناعة المعالجات .
- (32) قطب دخل / خرج (I/O) قابلة للتهيئة بشكل مستقل لتعمل مع المتحكم كدخل أو كخرج .
- (131) تعليمة تُنفّذ معظمها في دورة ساعة واحدة (تعليمات لغة الاسمبلي) .
- (32) مسجل من مسجلات الأغراض العامة بعرض (8-bit) للمسجل الواحد .
- قادر على تنفيذ (16) مليون تعليمة في الثانية الواحدة لدى وصل كريستالة قيمتها (16 MHz) معه.
- ذاكرة برنامج بحجم (16 Kbyte) من النوع (flash) غير قابلة للمحي وبديمومة تصل حتى (10,000) دورة مسح / كتابة .
- ذاكرة دائمة (EEPROM) بحجم (512 Byte) وبديمومة تصل حتى (100,000) دورة مسح / كتابة .
- ذاكرة مؤقتة (SRAM) بحجم (1 Kbyte) .
- أقفال برمجية لحماية البرنامج المكتوب داخل المتحكم (Firmware) بعد البرمجة .
- يمتلك (2) (مؤقت / عداد) بعرض (8-bit) و (مؤقت / عداد) واحد بعرض (16-bit) .
- يمتلك (مؤقت / عداد) واحد بعرض (8-bit) يعمل بنمط الزمن الحقيقي مع كريستالة خارجية مستقلة .
- مبدل (تشابهي / رقمي) (A/D Converter) بدقة (10-bit) وبثمانية مداخل قادر على التعامل مع ثماني إشارات تشابهية مختلفة بنفس الوقت .
- مؤقت مراقبة (Watchdog Timer) بهزاز كريستالي منفصل للتأكد من سير البرنامج داخل المتحكم بشكل سليم .
- نافذة اتصال تسلسلية (USART) قابلة للبرمجة يمكن أن تعمل بالنمط المتزامن أو غير المتزامن .
- نافذة اتصال تسلسلية (I^2C) بواجهة (Tow-Wire Serial Interface) .
- نافذة تسلسلية (SPI) تُتيح برمجة المتحكم دون فصله عن الدارة الالكترونية التي يعمل فيها (On-Board Programming) .
- مصادر مقاطعة داخلية وخارجية .
- مقارن تشابهي مبني ضمن الشريحة .
- أنماط نوم مختلفة ومتعددة تُتيح تشغيل المتحكم بأقل قدر ممكن لاستهلاك الطاقة وذلك لادخارها بأكبر قدر ممكن (Idle Mode, Power-Save Mode, Power-Down Mode, Standby Mode) .
- جهد التشغيل للمتحكم (ATmega16**L**) (2.7 to 5.5 V) .
  و للمتحكم (ATmega16) (4.5 to 5.5 V) .

أقطاب المتحكم (ATmega16) :



يمكن تصنيف أقطاب متحكمات (AVR) بشكل عام ضمن المجموعات التالية :

- **قطبي التغذية وقطب التصفير :**
  - ❖ قطب التغذية الموجبة (Vcc) : وهو القطب ذو الرقم (10) ويُوصل إلى منبع التغذية الموجبة (+5V) في دارة التحكم .
  - ❖ قطب الأرضي (GND) : وهو القطب ذو الرقم (11) ويُوصل إلى أرضي دارة التحكم .
  - ❖ قطب تصفير المتحكم الصغرى (RESET) : وهو القطب ذو الرقم (9) وهو قطب هام جداً للمتحكم حيث أنه لدى تطبيق (0) منطقي عليه يقوم بإعادة المتحكم إلى أول تعليمة من البرنامج المخزن في ذاكرته (تصفير المتحكم) .
- **قطبي وصل الهزاز الكريستالي الخارجي :**
  - ❖ قطب وصل الكريستالة الأول (XTAL1) : وهو القطب ذو الرقم (13) .
  - ❖ قطب وصل الكريستالة الثاني (XTAL2) : وهو القطب ذو الرقم (12) .
- **أقطاب تغذية المبدل التشابهي الرقمي :**
  - ❖ قطب التغذية الموجبة للمبدل (AVcc) : وهو القطب ذو الرقم (30) .
  - ❖ قطب الأرضي الخاص بالمبدل (GND) : وهو القطب ذو الرقم (31) .
  - ❖ قطب الجهد المرجعي للمبدل (AREF) : وهو القطب ذو الرقم (32) .
- **أقطاب الدخل / الخرج (I/O) :** وتتضمن ما تبقى من أقطاب المتحكم وعددها (32) قطب مؤلفة من اجتماع أربع نوافذ كل نافذة تمتلك ثمانية أقطاب :
  - ❖ النافذة(A) (Port A): وتمتد أقطابها من القطب رقم (40) وحتى القطب رقم (33) (PA0 → PA7).
  - ❖ النافذة(B) (Port B): وتمتد أقطابها من القطب رقم (1) وحتى القطب رقم (8) (PB0 → PB7).
  - ❖ النافذة(C) (Port C): وتمتد أقطابها من القطب رقم (22) وحتى القطب رقم (29) (PC0 → PC7).
  - ❖ النافذة(D) (Port D): وتمتد أقطابها من القطب رقم (14) وحتى القطب رقم (21) (PD0 → PD7).

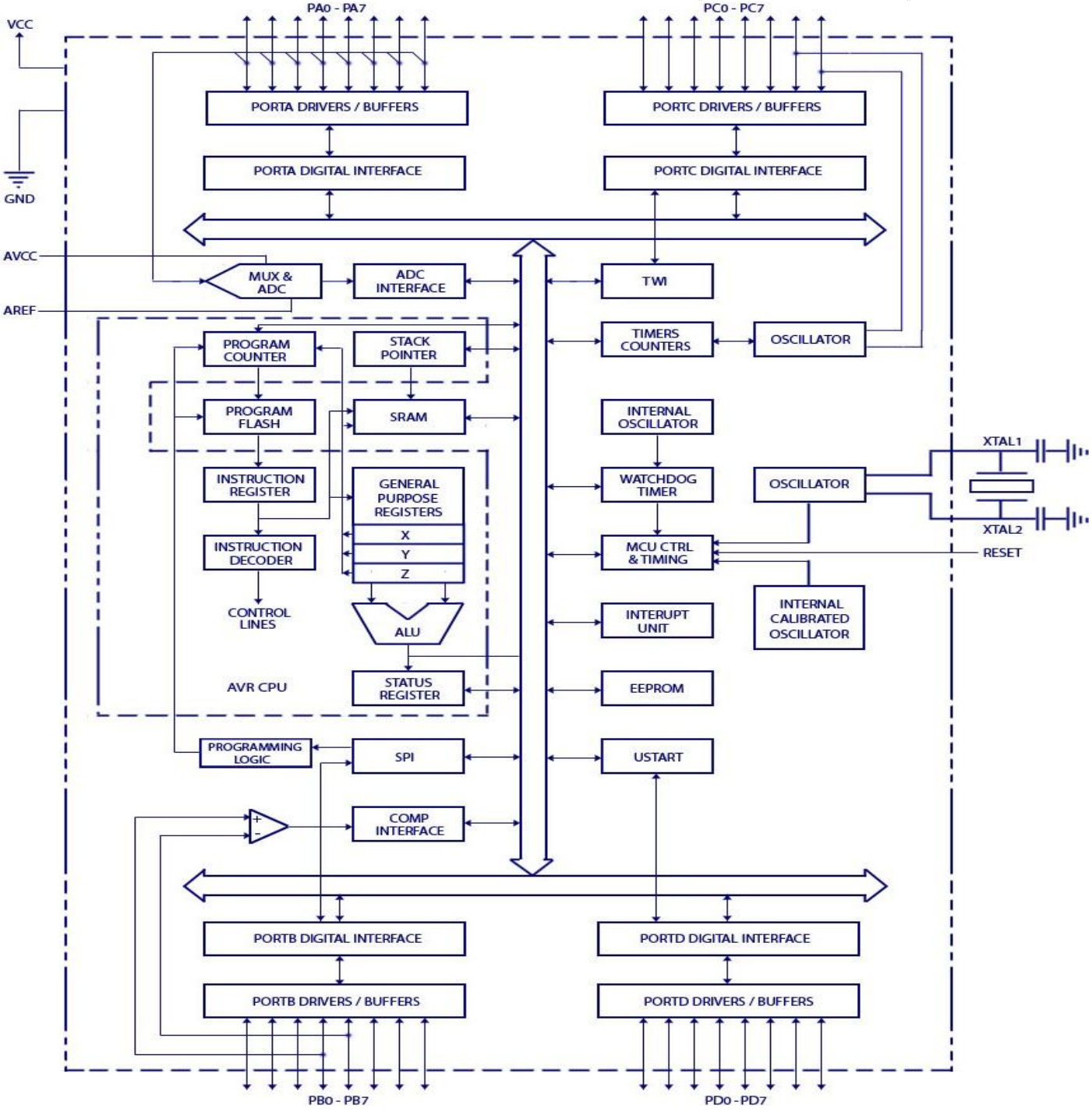
يمتلك كل قطب من هذه الأقطاب وظيفتان : وظيفة عامة و وظيفة خاصة.

أما *الوظيفة العامة* فهي المقدرة على تهيئته للعمل كقطب دخل **أو** قطب خرج للمتحكم .

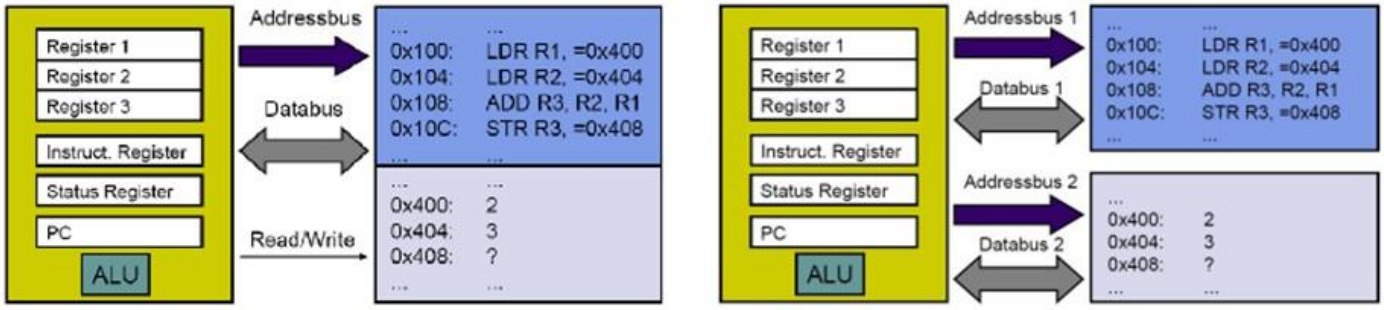
وأما *الوظيفة الخاصة* فتحدد بحسب عمل الوحدة المحيطيّة المرتبطة مع هذا القطب وتُكتب بجانب اسم القطب بين قوسين . ويمكن للقطب أن يعمل بإحدى الوظيفتين ولا يمكنه الجمع بينهما . فمثلاً القطب رقم (14) هو القطب الأول من النافذة (D) (PD0) (وظيفة عامة) , كما أنه يمثل قطب الاستقبال (RXD) *في حال* استخدام نافذة الاتصال التسلسلية (USART) (وظيفة خاصة) .

بنية المتحكم (ATmega16) :

يُبيّن الشكل التالي المخطط الصندوقي للمتحكم (ATmega16) :



نلاحظ من المخطط الصندوقي للمتحكم عدد الوحدات المحيطية الكبير المزروعة ضمن المتحكم الصغري .
كما نلاحظ مخطط الوصل التمثيلي بينها وبين مر المعطيات ضمن المتحكم .
بُنيت متحكمات (AVR) وفق مفهوم معمارية هارفارد (Harvard) حيث تُعنون ذاكرة البرنامج بخطوط منفصلة عن ذاكرة المعطيات (بخلاف معمارية فون نيومان (Von Neumann)) وهذا ما أعطى سرعة عالية في عمل المعالج . كما تم اعتماد تقنية (RISC) في تصميمه مما ادى إلى زيادة فاعليته و انخفاض كلفة تصنيعه .



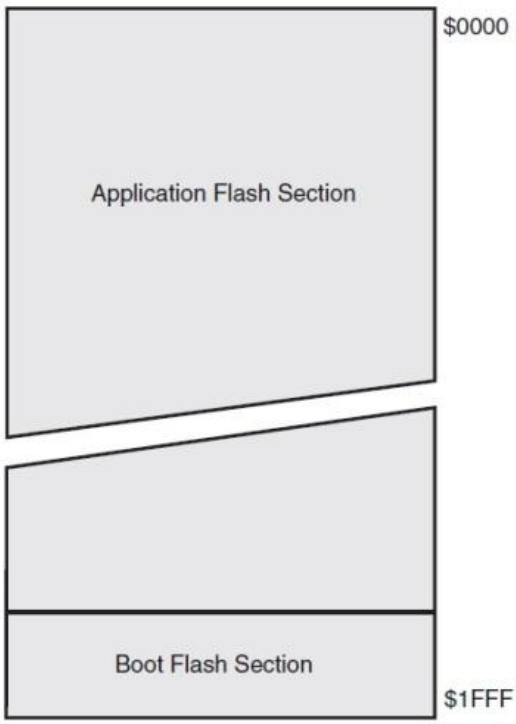
Von Neumann Architecture — Harvard Architecture

ذواكر المتحكم (ATmega16) :

يمتلك المتحكم (ATmega16) ثلاثة أنواع من الذواكر :

- ذاكرة البرنامج الوميضية (Flash Program Memory) .
- ذاكرة المعطيات (SRAM Data Memory) .
- الذاكرة (EEPROM) .

أولاً : ذاكرة البرنامج الوميضية (Flash Program Memory) :

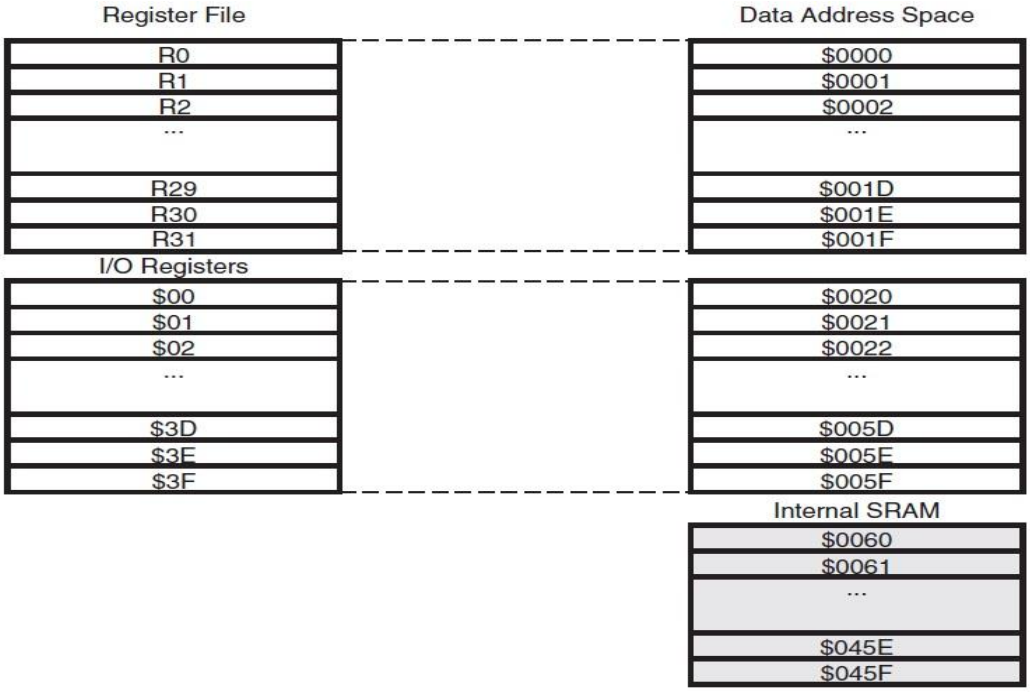


وهي ذاكرة دائمة من النوع (Flash) يبلغ حجمها (16 Kbyte) لا تفقد معلوماتها بزوال التغذية الكهربائية عنها . يُخزّن فيها تعليمات البرنامج الذي يقود المتحكم الصغري حيث تُشفّر كل تعليمة من تعليمات متحكمات (AVR) بـ(10 bit) وبالتالي فإن تنظيم حجرات هذه الذاكرة (8K × 16 bits) و تبلغ ديمومتها حوالي (10,000) دورة كتابة / مسح على الأقل .
تُتيح بنية الذاكرة إمكانية تخصيص قسم منها يتراوح حجمه بين (256 Byte → 4Kbyte) ليعمل كـ(Boot Flash Section) وذلك في حال تم استخدام ميزة (Self-Programming) في المتحكم إذ يمكن من خلال هذه الميزة كتابة و قراءة معطيات على ذاكرة البرنامج وبالتالي تغيير تعليمات البرنامج دون الحاجة إلى برمجة المتحكم بواسطة مبرمجة موصولة على جهاز الحاسب (المتحكم يُبرمج نفسه بنفسه) .

ثانياً : ذاكرة المعطيات (SRAM Data Memory) :

وهي ذاكرة مؤقتة يبلغ حجمها (1Kbyte) تفقد معلوماتها بمجرد انقطاع التغذية الكهربائية عنها , وهي مُنظّمة بشكل (1K×8 bits) وتُقسم إلى ثلاثة أقسام رئيسية :

- مسجلات الأغراض العامة (General Purpose Registers) : وعددها (32) مسجل .
- مسجلات التحكم أو مسجلات (I/O) (I/O Registers) : وعددها (64) مسجل .
- الذاكرة (SRAM) : وتُشكل ما تبقى من ذاكرة المعطيات وتعمل كذاكرة مؤقتة (Buffer) يستخدمها المتحكم لتنفيذ العمليات الحسابية وما يلزمه من تعليمات أخرى .



مسجلات الأغراض العامة :

وهي عبارة عن (32) مسجل تتواجد ضمن وحدة المعالجة المركزية للمتحكم (AVR CPU) وتُستخدم بشكل عام كمكان حفظ مؤقت (Buffer) ضمن تعليمات لغة التجميع للمتحكم . ويُبيّن الشكل المجاور تركيبة مسجلات الأغراض العامة الاثنين وثلاثين .

ضمن التعليمات يتم التعامل مع مسجلات الأغراض العامة إما عن طريق اسم المسجل (مثل R0,R2,R17) أو عن طريق عنوان المسجل (مثل $00,$02,$11) فمثلاً لتحميل القيمة الفورية ($5F) إلى المسجل (R16) نكتب التعليمة التالية :

```
LDI   R16,$5F
```

او نكتب التعليمة التالية :

```
LDI   $10,$5F
```

حيث أن ($10) هو عنوان المسجل (R16) .

R0	$00
R1	$01
R2	$02
...	
R13	$0D
R14	$0E
R15	$0F
R16	$10
R17	$11
...	
R26	$1A
R27	$1B
R28	$1C
R29	$1D
R30	$1E
R31	$1F

تُقسم مسجلات الأغراض العامة إلى قسمين :

القسم الأول يضم المسجلات التي تمتد من المسجل (R0) إلى المسجل (R15) (R0 → R15) .

القسم الثاني يضم المسجلات التي تمتد من المسجل (R16) إلى المسجل (R31) (R16 → R31) .

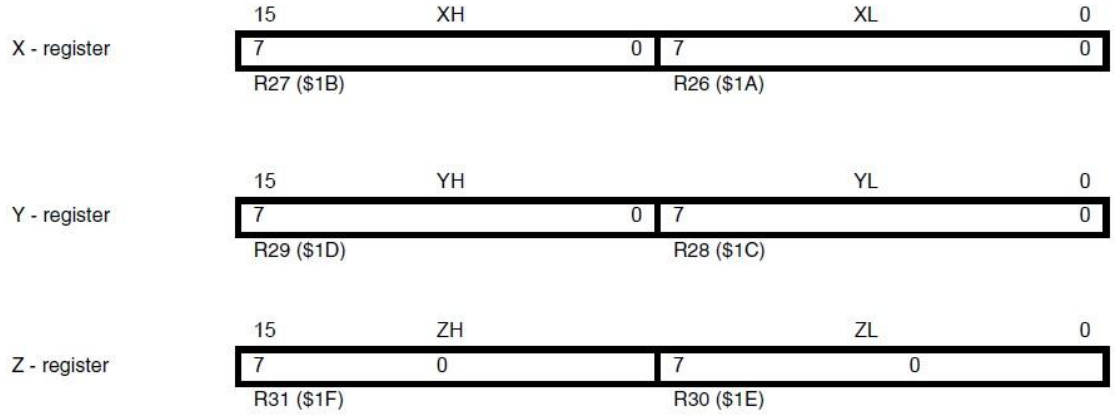
لا يختلف القسمان الأول والثاني عن بعضهما البعض سوى أن بعض تعليمات الاسمبلي الخاصة بمتحكمات (AVR) لا تُنفّذ إلا على القسم الثاني من مسجلات الأغراض العامة . هذه التعليمات هي :

(CBR , CPI , LDI , LDS , IN , OUT , SBR , STS , SUBI)

بينما تُنفّذ باقي تعليمات لغة الاسمبلي الخاصة بمتحكمات (AVR) على كامل مسجلات الأغراض العامة .

ملاحظة :

تُستخدم المسجلات (R26 → R31) لنفس الوظائف التي تقوم بها مسجلات الأغراض العامة الأخرى كما ويمكن استخدامها كثلاثة مسجلات بطول (16 bit) هي على الترتيب (X,Y,Z) وهي تُستخدم عادةً كمؤشر عنونة وذلك في حالة العنونة غير المباشرة . ويُبيّن الشكل التالي تركيبة هذه المسجلات :



مسجلات التحكم أو مسجلات (I/O) :

وهي عبارة عن (64) مسجل كل مسجل منها بطول (8 bit) تُكتب فيها بايتات تحكم تُحدد آلية عمل الوحدات الداخلية المتواجدة ضمن المتحكم الصغري حيث أن كل وحدة داخلية منها تمتلك مسجل تحكم واحد أو أكثر . فمثلاً النافذة (D) تمتلك ثلاثة مسجلات تحكم و (المؤقت / العداد 1) يمتلك عشرة مسجلات تحكم والمقارن التشابهي يمتلك مسجل تحكم واحد فقط ..... وهكذا .

يتم التعامل مع مسجلات التحكم إما عن طريق اسم المسجل (مثل DDRA , TCNT0 , SREG) أو عن طريق عنوان المسجل (مثل $22 , $50 , $5F) .

تتم القراءة والكتابة من وإلى مسجلات التحكم عن طريق التعليمتين (IN,OUT) حصراً ولا يمكن استخدام تعليمة التحميل الفوري (LDI) مع مسجلات التحكم . حيث أن التعليمة (IN) تقرأ قيمة مسجل تحكم وتضعها في مسجل أغراض عامة والتعليمة (OUT) تكتب قيمة من مسجل أغراض عامة إلى مسجل تحكم . فمثلاً إذا أردنا قراءة القيمة الموجودة في مسجل التحكم (PINB) نقوم بنسخها إلى أحد مسجلات الأغراض العامة (القسم الثاني حصراً) عن طريق التعليمة التالية :

```
IN   R21,PINB
```

أما إذا أردنا كتابة القيمة الست عشرية ($4E) داخل مسجل التحكم (DDRB) عندها نقوم بكتابة هذه القيمة إلى أحد مسجلات الأغراض العامة (القسم الثاني حصراً) ومن ثم ننسخ محتوى مسجل الأغراض العامة إلى مسجل التحكم .

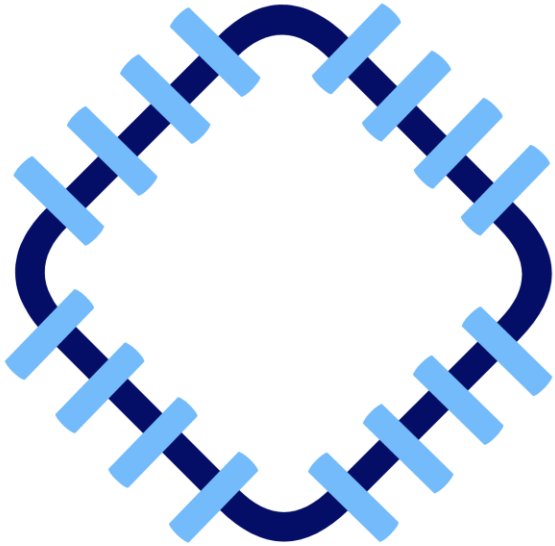
ولفعل ذلك يلزمنا التعليمتين التاليتين :

كتابة القيمة ($4E) إلى المسجل (R18) LDI R18,$4E

نسخ محتوى المسجل (R18) إلى مسجل التحكم (DDRB) OUT DDRB,R18

ثالثاً : الذاكرة (EEPROM) :

يحتوي المتحكم (ATmega16) على ذاكرة من النوع (EEPROM) بحجم يبلغ (512 Byte) وهي منظمة كحيز معطيات منفصل (512 × 8 bits) ويمكن القراءة والكتابة على أي حجرة من حجراتها بشكل مستقل , وتبلغ ديمومة هذه الذاكرة ما يقارب (100,000) دورة كتابة / مسح .



Micromir

Work Intelligently

SalahAldeen St.-Hama-SYRIA

Tel.:+963332539446 - Mob.:+963991045658

الجلسة الثانية

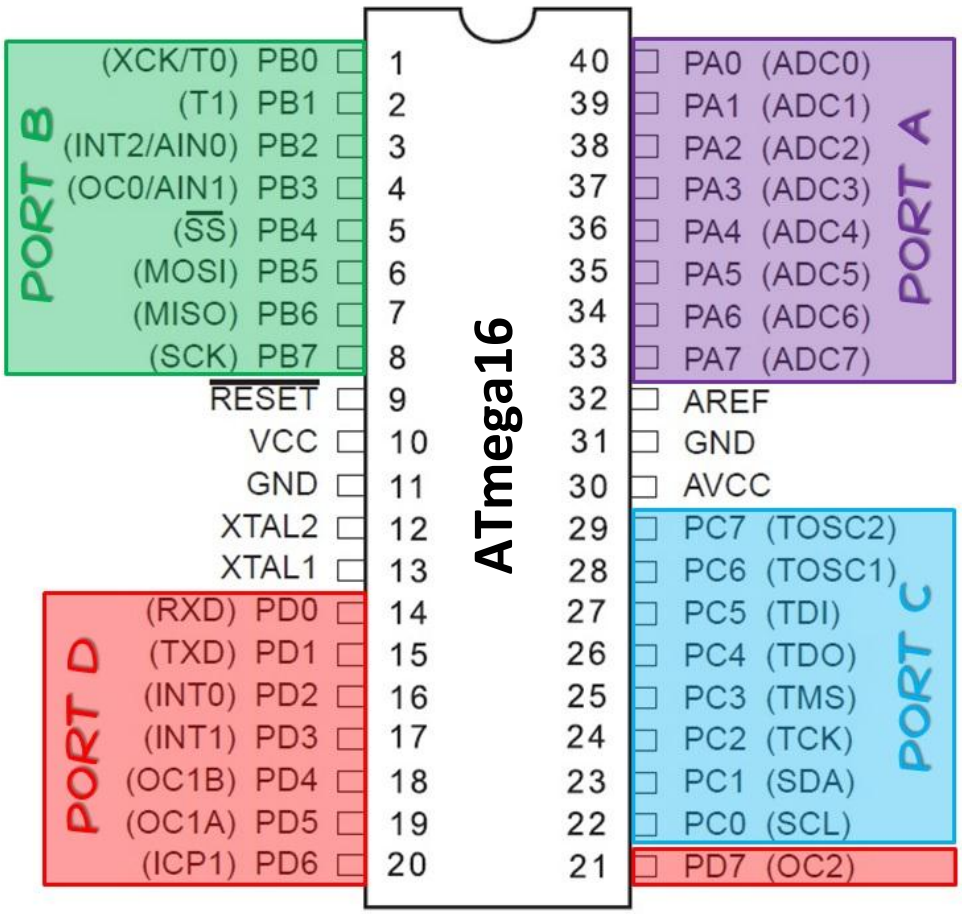
نوافذ الدخل / الخرج في متحكمات (AVR)

تمتلك متحكمات (AVR) نوافذ دخل / خرج (I/O) تسمح لها بإرسال / استقبال إشارات رقمية (1 or 0) على أقطابها الخارجية . وتضم النافذة الواحدة بشكل عام ثمانية أقطاب دخل / خرج (I/O) يُمكن أن تُهيئ بشكل مستقل كأقطاب خرج أو دخل . وسندرس فيما يلي أقطاب الدخل / الخرج عند المتحكم (ATmega16) حيث تُطبّق هذه الدراسة على جميع متحكمات شركة (ATMEL) من النوع (AVR) .

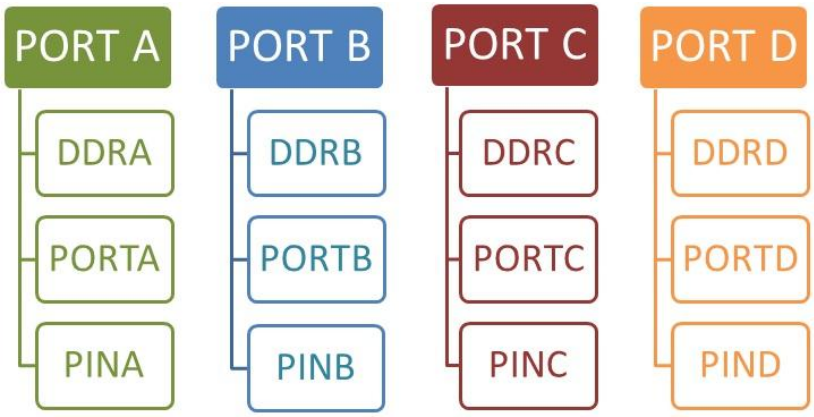
أقطاب دخل / خرج المتحكم (ATmega16) :

يمتلك المتحكم (ATmega16) أربع نوافذ (I/O) كما ذكرنا ذلك سابقاً (Port A , Port B , Port C , Port D) وتتألف كل نافذة من ثمانية أقطاب تعمل أقطابها إما كأقطاب خرج (Output) أو كأقطاب دخل (Input) . ويمكن تهيئة بعض أقطاب النافذة الواحدة كدخل وباقي أقطابها كخرج (كل قطب مستقل عن الآخر في العمل) .

لدى تهيئة قطب المتحكم كقطب خرج يستطيع عندها قيادة حمل بما لا يتجاوز (20 mA) وإلا فإن قطب المتحكم سوف يُعطب . كما أن التيار الكلي المار بالمتحكم الصغري يجب أن لا يتجاوز (200 mA) .



تمتلك كل نافذة (I/O) في المتحكم ثلاثة مسجلات تحكم هي (PINx , PORTx , DDRx) حيث (x) هي اسم النافذة و يمكن عن طريق هذه المسجلات التعامل مع أقطاب النوافذ بشكل كامل كأقطاب دخل أو خرج. وسندرس فيما يلي مسجلات التحكم الثلاثة الخاصة بالنافذة (B) وما يسري عليها ينطبق على جميع نوافذ المتحكم مع استبدل الحرف (B) باسم النافذة المطلوبة :



مسجل اتجاه معطيات النافذة (B) (DDRB) :

*Port **B** **D**ata **D**irection **R**egister (DDRB)*

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

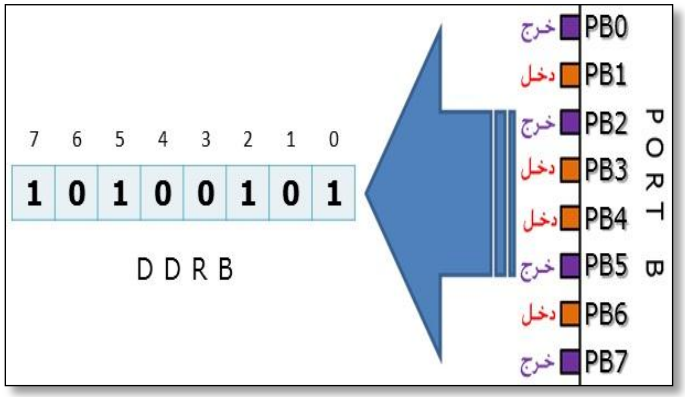
يتم عن طريق هذا المسجل تحديد أياً من أقطاب النافذة (B) ستعمل كأقطاب خرج وأياً منها ستعمل كأقطاب دخل ويتم تحديد ذلك في بداية البرنامج *ولمرة واحدة فقط* إلا إذا أردنا تغيير اتجاه سير المعطيات على أقطاب النافذة مرة أخرى من داخل البرنامج الرئيسي .

يتحكم كل بت من بتات المسجل (DDRB) بالقطب المقابل له من أقطاب النافذة (B) فمثلاً البت (DDB3) من المسجل يتحكم باتجاه معطيات القطب (PB3) من النافذة و البت (DDB7) من المسجل يتحكم باتجاه معطيات القطب (PB7) من النافذة وهكذا ..... ويتم ذلك وفق القاعدتين التاليتين :

القطب المُراد تهيئته كقطب **دخل** من أقطاب النافذة (B) نكتب **(0)** في البت المقابل له من المسجل (DDRB)

القطب المُراد تهيئته كقطب **خرج** من أقطاب النافذة (B) نكتب **(1)** في البت المقابل له من المسجل (DDRB)

فمثلاً إذا أردنا تهيئة النافذة (B) كنافذة خرج بالكامل فإننا نكتب القيمة الست عشرية ($FF) في المسجل (DDRB) . وإذا أردنا تهيئة النافذة (B) كنافذة دخل بالكامل فإننا نكتب القيمة الست عشرية ($00) في المسجل (DDRB) . أما إذا أردنا تهيئة الأقطاب (PB0,PB2,PB5,PB7) من النافذة (B) كأقطاب خرج والأقطاب المتبقية منها كأقطاب دخل فإننا نكتب القيمة ($A5) في المسجل (DDRB) . (لاحظ الشكل جانباً واستنتج السبب)



مسجل معطيات النافذة (ß) (PORTB) :

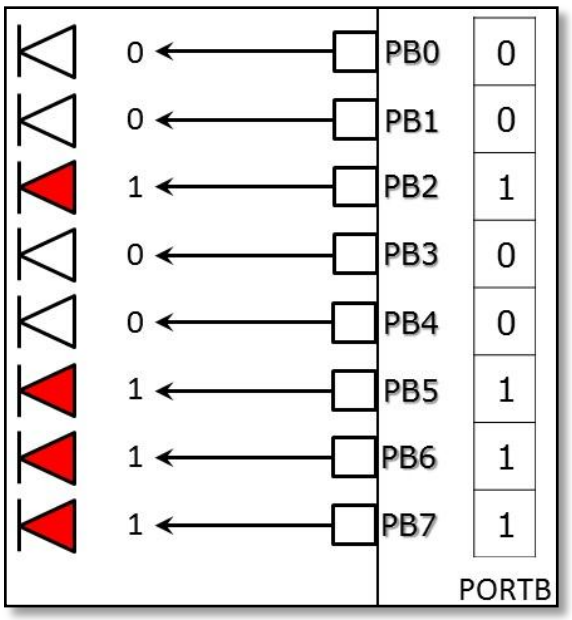
***Port B** Data Register (PORTB)*

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

تختلف وظيفة مسجل التحكم (PORTB) بحسب نوع الأقطاب التي سيتعامل معها (أقطاب دخل أو أقطاب خرج) وسنفصّل هاتين الوظيفتين فيما يلي :

- *في حالة أقطاب الخرج :*

يقوم هذا المسجل بتخريج القيم المنطقية (0,1) على أقطاب النافذة فيزيائياً في حال كون هذه الأقطاب مهيئة كأقطاب خرج , فمثلاً إذا تم تهيئة النافذة (ß) كنافذة خرج ومن ثم كُتب على المسجل (PORTB) القيمة الست عشرية ($E4) مثلاً فعندها سيتم تخريج القيم المنطقية (1110 0100) على أقطاب النافذة (ß) بالترتيب (ابتداءاً من القطب PB0 وانتهاءاً بالقطب PB7) كما هو مبين بالشكل جانباً .



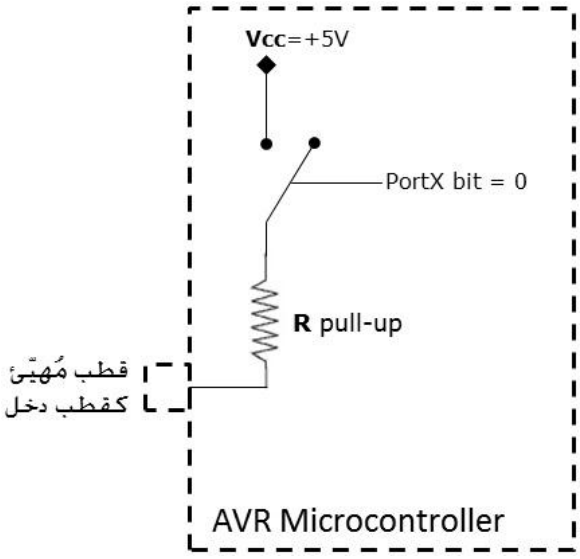
- *في حالة أقطاب الدخل :*

أما في حالة أقطاب الدخل فيصبح للمسجل (PORTB) وظيفة مغايرة تماماً للوظيفة السابقة إذ أنه يصبح مسؤولاً عن وصل وفصل مقاومات الرفع الداخلية لأقطاب الدخل , ولابد لنا هنا من الوقوف بُرهةً عند مقاومات الرفع الداخلية المُهيَّئة للوصل مع أقطاب المتحكم الصغري .

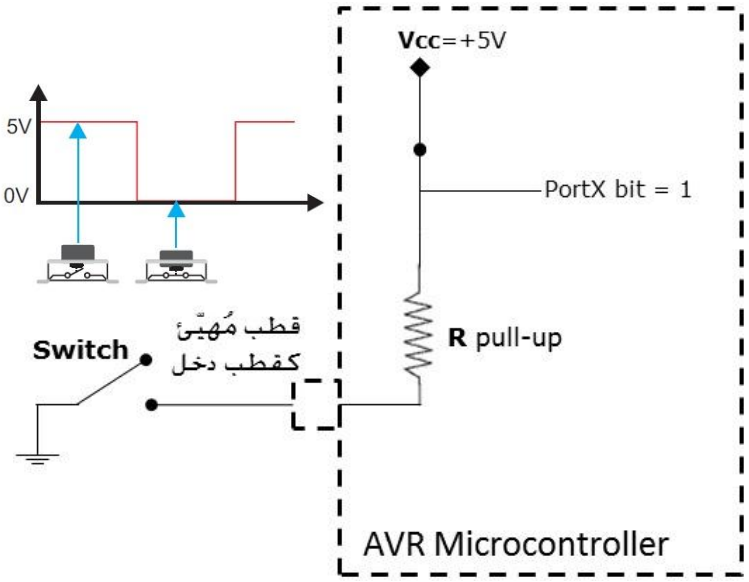
مقاومة الرفع الداخلية :

تم تزويد جميع أقطاب الدخل / الخرج الخاصة بمتحكمات (AVR) بإمكانية وصلها داخلياً مع مصدر التغذية (Vcc=+5V) عن طريق مقاومة (10 KΩ) وهي ما تسمى بمقاومة الرفع الداخلية. وتُفيد هذه الخاصية فقط لدى تهيئة الأقطاب كأقطاب دخل وليس لها أي دور بالنسبة لأقطاب الخرج .

إن الفائدة المحصّلة من وصل مقاومة الرفع الداخلية مع قطب الدخل هي تهيئة هذا القطب بقيمة افتراضية هي (1) منطقي (+5V) وبالتالي لدى تغير هذه القيمة عنده نعرف أن شيء ما طرأ على قطب الدخل المطلوب .



إن وصل مقاومة الرفع الداخلية عادةً يتم لدى وصل زر كباس (Switch) مع قطب الدخل المطلوب للمتحكم الصغري كما هو مُبين في الشكل التالي :



في الشكل المبين جانباً لدى فتح زر الكباس (Switch) الموصول مع قطب الدخل تتوضّع على هذا القطب القيمة (1) منطقي نتيجة لوصل مقاومة الرفع الداخلية لديه .

أما عند إغلاق زر الكباس (Switch) يتصل قطب الدخل عند المتحكم مع الأرضي (GND) وبالتالي تتوضّع عليه القيمة (0) منطقي .

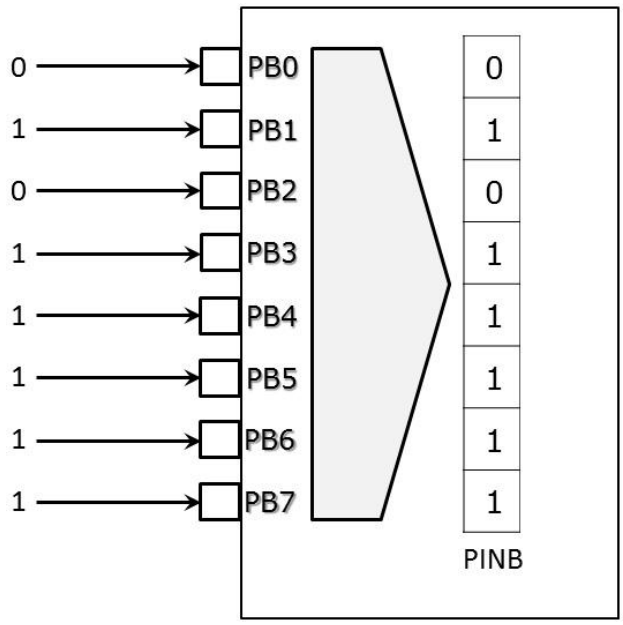
إن اختلاف القيمتين المنطقيتين عند فصل ووصل زر الكباس (Switch) يسمح لنا بمعرفة فيما إذا كان هذا الزر مضغوطاً أو لا . وهذا هو الاستخدام الأكثر شيوعاً لمقاومة الرفع الداخلية .

إن المسؤول عن وصل أو فصل مقاومة الرفع الداخلية الخاصة بأقطاب الدخل للنافذة هو مسجل التحكم (PORTB) حيث أنه لدى وضع (1) منطقي في البت المقابل لقطب الدخل المطلوب يؤدي ذلك إلى وصل مقاومة الرفع الداخلية لهذا القطب ويتم فصلها بوضع (0) منطقي في البت المقابل للقطب .

عنوان دخل أقطاب النافذة (ß) (PINB) :

*Port **B** Input **PIN**s Address (PINB)*

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
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	



وهو المسجل المسؤول عن قراءة الأقطاب المُهيّئة كأقطاب دخل من أقطاب النافذة (ß) حيث أن هذا المسجل قابل للقراءة فقط (لا يمكن الكتابة عليه عبر تعليمات البرنامج داخل المتحكم لأن من يكتب عليه هو الوسط الخارجي عبر أقطاب الدخل في النافذة كما هو مُبين بالشكل جانباً) .

بشكل عام لمعرفة القيمة المنطقية المتوضعة على قطب دخل ما في المتحكم نقوم بقراءة البت المقابل لهذا القطب من المسجل (PIN) الخاص بالنافذة المطلوبة فإذا كانت قيمة هذا البت (1) منطقي نستدل من ذلك على توضع القيمة (1) منطقي (+5V) على هذا القطب ونفس الأمر بالنسبة لـ(0) منطقي .

الخلاصة :

- نستخلص مما سبق أنه عند التعامل مع أقطاب الدخل / الخرج (I/O) في متحكمات (AVR) يجب في البداية معرفة أياً منها سيعمل كأقطاب دخل وأياً منها سيعمل كأقطاب خرج ومن ثم تهيئتها من خلال الكتابة على مسجل التحكم **(DDRX)** (حيث (X) هي اسم البوابة) وتَذكّر دوماً أن :

وضع **(1)** منطقي في بت ما من المسجل (DDRX) يُهيئ القطب المقابل له **كقطب خرج**

وضع **(0)** منطقي في بت ما من المسجل (DDRX) يُهيئ القطب المقابل له **كقطب دخل**

- ومن ثم نُحدد أياً من أقطاب الدخل نريد وصل مقاومة الرفع الداخلية معه وأياً منها لا نريد وصلها مع المقاومة ومن ثم نقوم بذلك من خلال مسجل التحكم **(PORTX)** (حيث (X) هي اسم البوابة) وتذكّر دوماً أن :

وضع **(1)** منطقي في بت ما *مقابل لقطب دخل* من المسجل (PORTX) يصل مقاومة الرفع الداخلية مع هذا القطب

وضع **(0)** منطقي في بت ما *مقابل لقطب دخل* من المسجل (PORTX) يفصل مقاومة الرفع الداخلية عن هذا القطب

- بعد ذلك يُمكننا تخريج قيم منطقية على أقطاب خرج المتحكم من خلال *الكتابة* على مسجل التحكم **(PORTX)** (حيث (X) هي اسم البوابة) . كما ويمكننا قراءة القيم المنطقية المتوضعة على أقطاب دخل المتحكم من خلال *قراءة* مسجل التحكم **(PINX)** (حيث (X) هي اسم البوابة) .

يُبيّن الجدول التالي جميع الحالات المنطقية الممكنة بالنسبة لمسجلي التحكم (DDRBn) و (PORTBn) للنافذة (B) حيث أن (n) هو رقم البت ضمن مسجل التحكم ويُقابله رقم القطب ضمن نافذة المتحكم :

DDRBn	PORTBn	نوع القطب	مقاومة الرفع الداخلية	ملاحظات
0	0	دخل	مفصولة	ممانعة عالية (Z)
0	1	دخل	موصولة	دخل مرفوع إلى (1) منطقي
1	0	خرج	-----	خرج قيمته (0) منطقي
1	1	خرج	-----	خرج قيمته (1) منطقي

تعليمات لغات البرمجة الخاصة بأقطاب الدخل / الخرج (I/O)

أولاً : لغة الإسمبلي (Assembly Language) :

في لغة الاسمبلي يمكننا الكتابة على أي مسجل تحكم من خلال التعليمة (OUT) كما يمكننا القراءة من أي مسجل تحكم من خلال التعليمة (IN) مع مراعاة وضع مسجل أغراض عامة (من القسم الثاني لملف المسجلات) كمسجل وسيط بين مسجلات التحكم عند القراءة وعند الكتابة .

أمثلة :

1. قم بتهيئة النافذة (C) كنافذة خرج ثم خرج عليها القيمة الست عشرية ($3D) .

```
LDI   R20,$FF
OUT   DDRC,R20

LDI   R20,$3D
OUT   PORTC,R20
```

2. قم بتهيئة النافذة (D) لتعمل أقطابها كما يلي :

PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
دخل	خرج	دخل	دخل	خرج	خرج	دخل	دخل

ومن ثم قم بتخريج القيمة (1) منطقي على قطب الخرج (PD6) . ومن ثم قم بقراءة حالة أقطاب الدخل منها.

أولاً نقوم بتحديد القيمة الست عشرية المراد كتابتها ضمن مسجل التحكم (DDRD) وذلك لتهيئة الأقطاب السابقة على الشكل المطلوب :

DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
دخل	خرج	دخل	دخل	خرج	خرج	دخل	دخل
0	1	0	0	1	1	0	0
4				**C**			

```
LDI   R19,$4C
OUT   DDRD,R19

LDI   R20,$40
OUT   PORTD,R20

IN    R16,PIND
```

ثانياً : لغة الباسكم (BASCOM AVR Language) :

في البداية لابد لنا من التعرف على بعض التعليمات الأساسية في لغة الـ(BASCOM AVR) :

- تعليمات التوجيه الأساسية :

شرح التعليمة	التعليمة
تعريف المترجم (BASCOM) بنوع المتحكم المُستخدم	`$regfile = "m16def.dat"`
تعريف المترجم بتردد الكريستالة المُوصولة مع المتحكم	`$crystal = 4000000`
تعريف المترجم بمعدل بود المستخدم في الاتصال التسلسلي	`$baud = 9600`

- تعليمات التعامل مع أقطاب الدخل / الخرج :

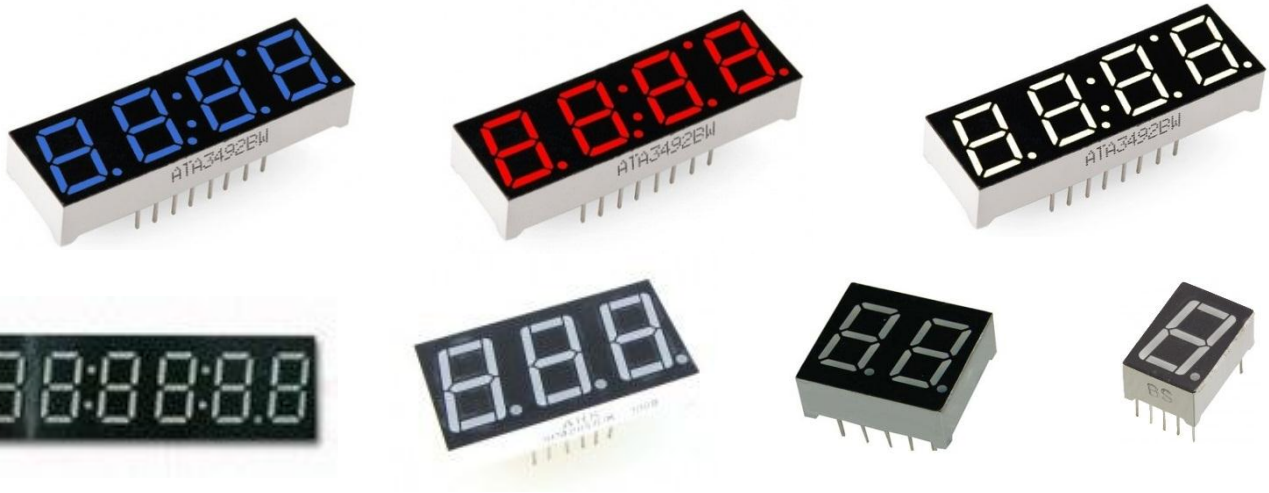
شرح التعليمة	التعليمة
تهيئة النافذة (A) كنافذة خرج	`Config PortA = Output`
تهيئة النافذة (D) كنافذة دخل	`Config PinD = Input`
تهيئة القطب (PD7) من النافذة (D) كقطب دخل	`Config PortD.7 = Input`
تهيئة القطب (PB3) من النافذة (B) كقطب خرج	`Config PortB.3 = Output`
كتابة القيمة العشرية (255) داخل المسجل (PortC)	`PortC = 255`
كتابة القيمة الثنائية (11000101) داخل المسجل (DDRA)	`DDRA = &B11000101`
كتابة القيمة الست عشرية (4A) داخل المسجل (DDRD)	`DDRD = &H4A`

الجلسة الثالثة

شاشة الإظهار ذات السبع قطع (7-Segments Display)

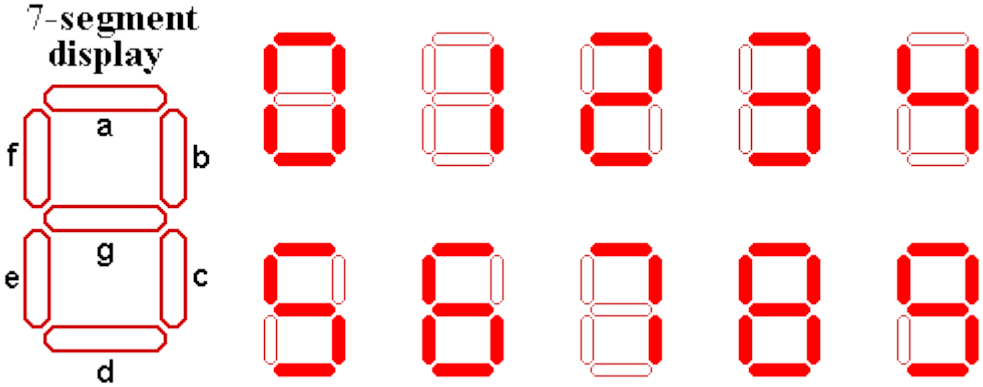
Seven Segments Display (SSD)

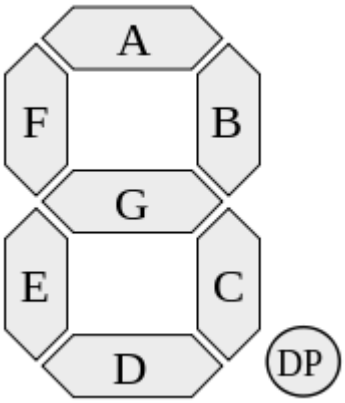
تُعتبر شاشة الإظهار ذات السبع قطع (7-Seg Display) من أشهر أدوات الإظهار المتوفرة في الدارات الالكترونية وبالرغم من محدوديّة الإظهار فيها (تُظهر الأرقام وبعض الأحرف فقط) إلا أنها منتشرة بشكل واسع وبألوان مختلفة وبخانة واحدة أو اثنتين أو ثلاث أو أربع أو ست أو ثماني خانات وبكافة القياسات . وسندرس في هذه الجلسة طرق قيادة شاشة الإظهار ذات السبع قطع (7-Seg) عبر وصلها مع المتحكم الصغري بالإضافة إلى البنية الداخلية لها .



البنية الداخلية لشاشة الإظهار ذات السبع قطع (7-Seg Display) :

لا تمتلك شاشة الإظهار ذات السبع قطع (7-Seg) بنية معقدة على خلاف أدوات الإظهار الأخرى (كشاشة الكريستال السائل (LCD) مثلاً) إذ أنها تتألف - في أبسط حالاتها - من سبعة ليدات ضوئية تم ترتيبها ضمن هيكل بلاستيكي خارجي بحيث تصطف بشكل يسمح بإظهار الأرقام (بشكل أساسي) وبعض الأحرف عليها وذلك بتشغيل ليدات معينة وإطفاء ليدات أخرى ضمن هذه القطع السبع .

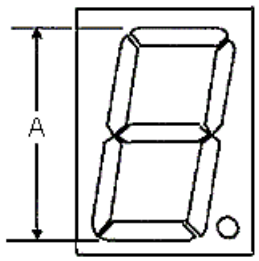




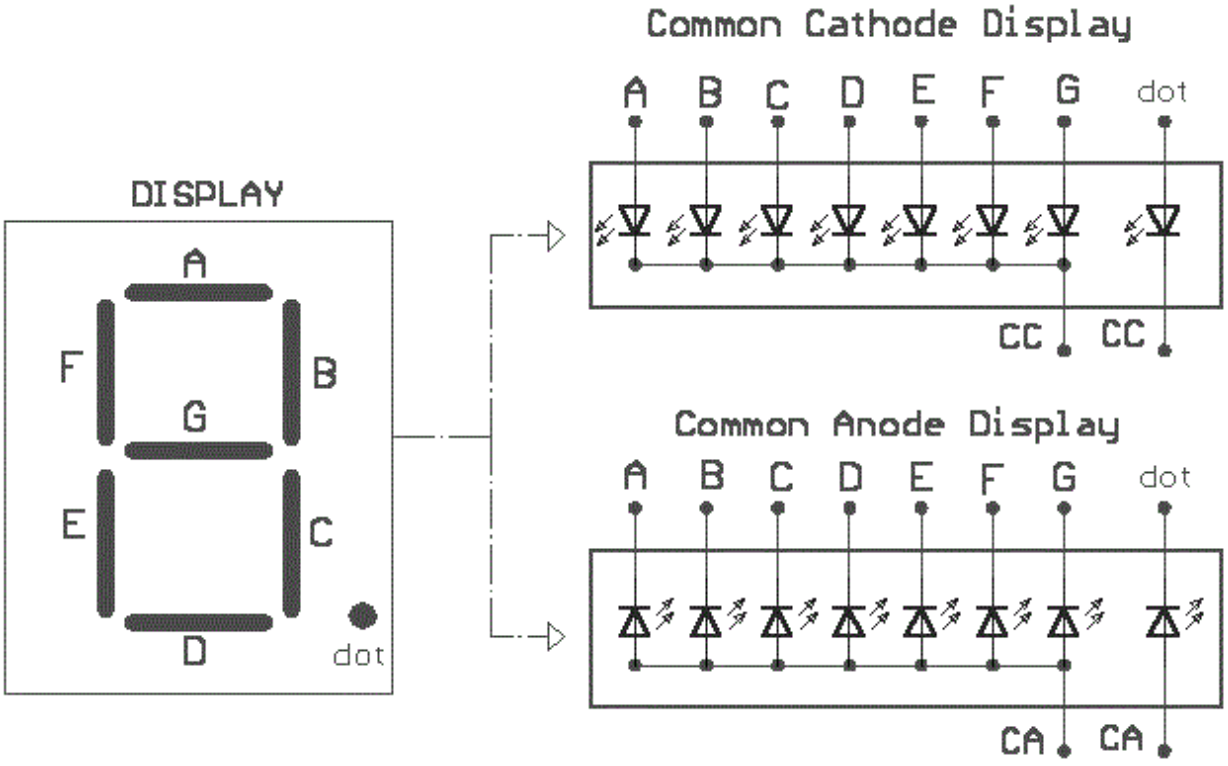
- تتألف شاشة الإظهار ذات السبع قطع (7-Seg) من سبع قطع – اسم على مُسمّى – يتوضع وراء كل قطعة ليد واحد أو أكثر يُصطلح على تسمية هذه القطع بالأحرف الانكليزية (A,B,C,D,E,F,G) بالترتيب وذلك ابتداءاً من أعلى الشاشة وبالمرور على القطع باتجاه دوران عقارب الساعة انتهاءاً بالقطعة الداخلية. كما وتُزوّد عادةً جميع الشاشات بقطعة إضافية على شكل نقطة مسؤولة عن إظهار النقطة بجانب الرقم يُرمز لها عادة (DP) كما هو مبين بالشكل المجاور :

كما ويمكن أن تُزوّد شاشات أخرى بقطع إضافية خاصة بها .

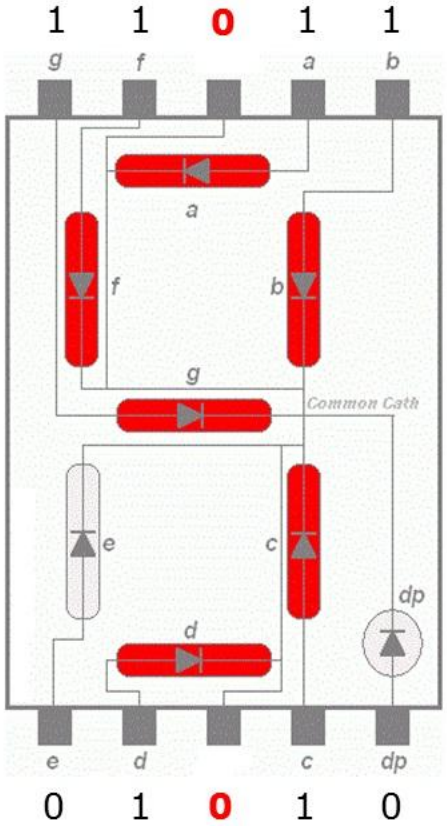
- تمتلك شاشات السبع قطع مقاسات مختلفة كما أنها مختلفة الألوان , فمنها اللون الأحمر والأخضر والأصفر والأزرق والبرتقالي وذلك بحسب لون الليدات الداخلية التي تُضيء القطع السبع ضمن الشاشة. كما ويمكن أن تتوفر هذه الشاشة بكافة الألوان إذا تمت صناعتها من ليدات (RGB) .



- أما بالنسبة للقياس فتُقاس عادةً شاشات السبع قطع بالإنش (inch) ويُعتمد معيار موحد لمقاسات هذه الشاشة وهو المسافة بين أعلى القطعة (A) من الشاشة إلى أسفل القطعة (D) . فعندما نقول مثلاً شاشة سبع قطع بمقاس (2.3") إنش فهذا يعني أن المسافة بين أعلى القطعة (A) وأسفل القطعة (D) منها هي (2.3") أي (2.3×2.54=5.8 c.m.) وهكذا ..... .
- يمكن تصنيف شاشات الإظهار ذات السبع قطع بالنسبة لطريقة وصل الليدات الداخلية إلى صنفين رئيسيين : *شاشات المهبط المشترك* و *شاشات المصعد المشترك* .



أولاً : شاشات المهبط المشترك (Common Cathode Display) :

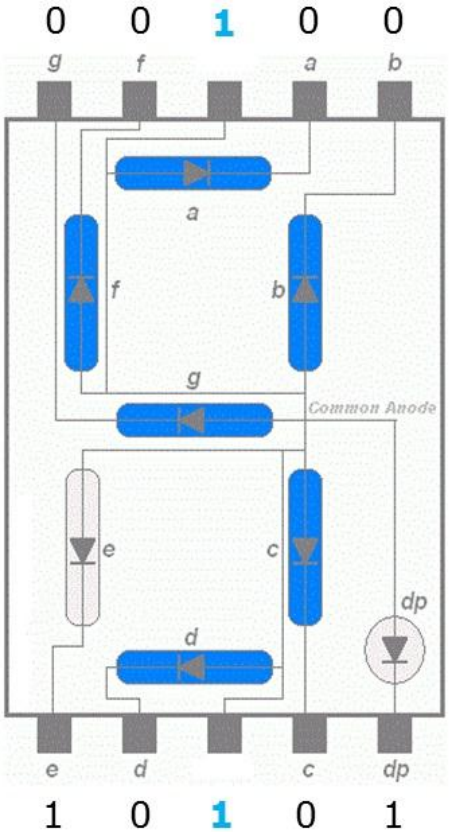


في هذا النوع من الشاشات يتم وصل جميع *مهابط* الليدات الداخلية مع بعضها البعض ومن ثم ربطها مع قطب خارجي من أقطاب الشاشة يُسمّى قطب التغذية السالبة المُشترك , بينما تبقى مصاعد الليدات محررة بحيث يرتبط كل مصعد ليد مع القطب الموافق للقطعة التي يُضيؤها هذا الليد من الشاشة .

الآن في هذا النوع إذا أردنا إضاءة أي قطعة من القطع السبع للشاشة علينا وصل قطب التغذية السالبة منها إلى أرضي الدارة (0 V) (0 منطقي) ومن ثم وصل قطب القطعة المُراد تشغيلها إلى قطب تغذية الدارة (+5 V) (1 منطقي) وبذلك نكون قد أغلقنا دارة الليد المطلوب وبالتالي يسري فيه التيار ويُنار الليد , وفي حال أردنا عدم إضاءة ليد معين نقوم بوصل قطبه إلى أرضي الدارة (0 V) (0 منطقي).

وبالتالي لإظهار الرقم (9) على شاشة ذات سبع قطع من نوع المهبط المشترك (Common Cathode Display) نطُبق على أقطابها القيم المنطقية المُبيّنة بالشكل المجاور :

ثانياً : شاشات المصعد المشترك (Common Anode Display) :



في هذا النوع من الشاشات يتم وصل جميع *مصاعد* الليدات الداخلية مع بعضها البعض ومن ثم ربطها مع قطب خارجي من أقطاب الشاشة يُسمّى قطب التغذية الموجبة المُشترك , بينما تبقى مهابط الليدات محررة بحيث يرتبط كل مهبط ليد مع القطب الموافق للقطعة التي يُضيؤها هذا الليد من الشاشة .

الآن في هذا النوع إذا أردنا إضاءة أي قطعة من القطع السبع للشاشة علينا وصل قطب التغذية الموجبة منها إلى قطب التغذية في الدارة (+5 V) (1 منطقي) ومن ثم وصل قطب القطعة المُراد تشغيلها إلى قطب أرضي الدارة (0 V) (0 منطقي) وبذلك نكون قد أغلقنا دارة الليد المطلوب وبالتالي يسري فيه التيار ويُنار الليد , وفي حال أردنا عدم إضاءة ليد معين نقوم بوصل قطبه إلى قطب التغذية في الدارة (+5 V) (1 منطقي) .

وبالتالي لإظهار الرقم (9) على شاشة ذات سبع قطع من نوع المصعد المشترك (Common Anode Display) نطُبق على أقطابها القيم المنطقية المُبيّنة بالشكل المجاور :

طُرق قيادة شاشة الإظهار (SSD) عن طريق المتحكم الصغري

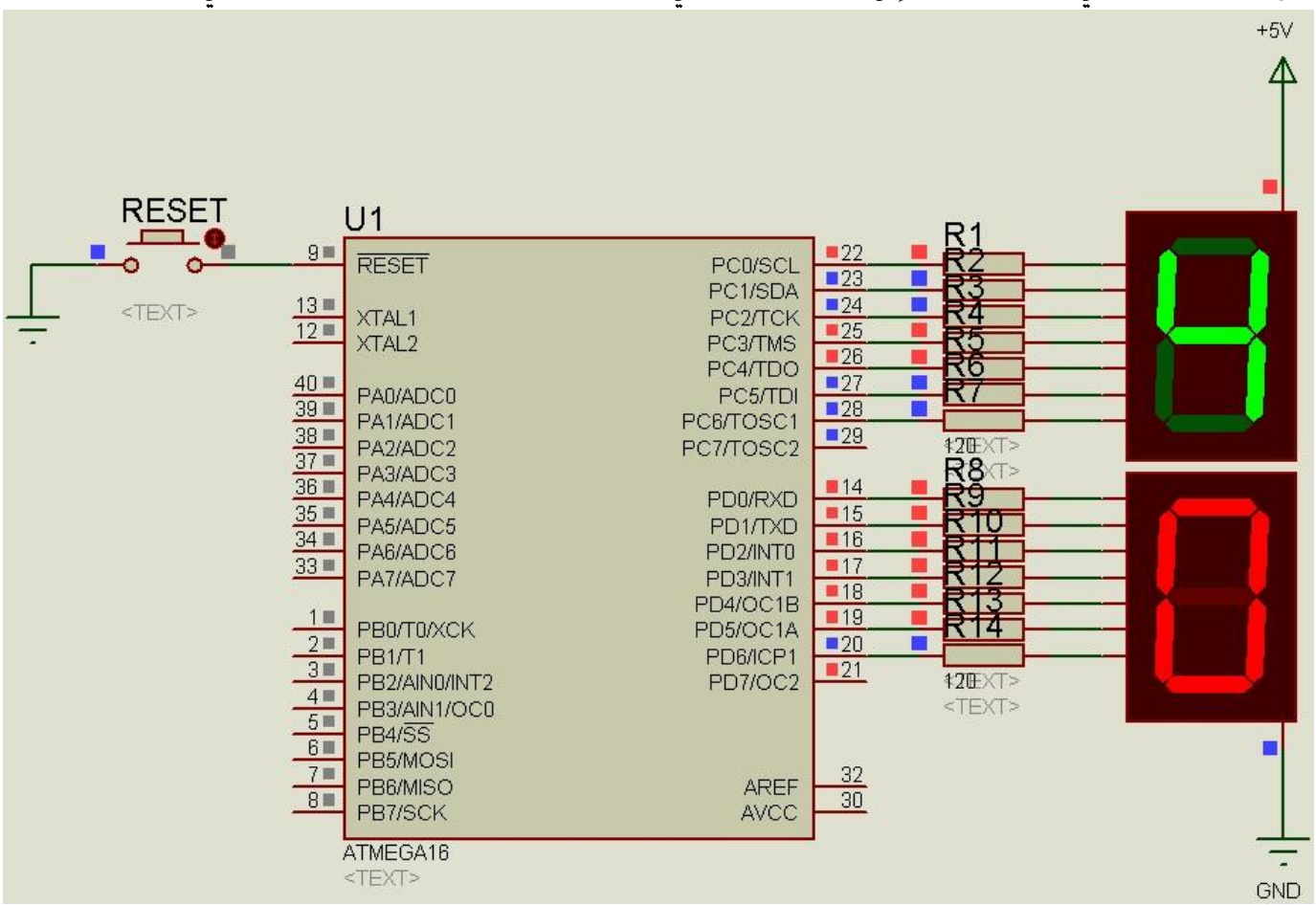
هناك عدة طُرق لقيادة شاشة الإظهار ذات السبع قطع (7-seg Display) عبر وصلها مع المتحكم الصغري سنستعرض في هذه الفقرة أهم تلك الطرق المستخدمة في الحياة العملية وسنذكر ميزات وعيوب كل طريقة حيث أن كل طريقة تمتلك مخطط وصل خاص بها . كما أن برنامج المتحكم يختلف أيضاً تبعاً لطريقة وصله مع شاشة الإظهار ذات السبع قطع (7-Seg Display) .

أولاً : طريقة الوصل المباشر :

وهي أبسط طريقة للوصل حيث يتم فيها وصل أقطاب المتحكم مباشرة (عبر مقاومة) إلى أقطاب شاشة (SSD) السبعة (A,B,C,D,E,F,G) , وبوصل القطب المشترك للشاشة للأرضي أو للتغذية (حسب نوع الشاشة) يصبح التحكم بإطفاء أو تشغيل ليدات القطع السبعة عن طريق إرسال (0) أو (1) منطقي مباشرة من أقطاب خرج المتحكم إلى أقطاب تلك الليدات .

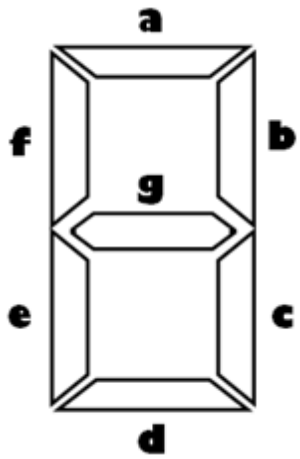
مخطط التوصيل :

بفرض أنه لدينا خانتين (SSD) الأولى (باللون الأخضر) من النوع مصعد مشترك , والثانية (باللون الأحمر) من النوع مهبط مشترك نقوم بوصل أقطاب الخانة الأولى (A,B,C,D,E,F,G) مع أقطاب النافذة (PORTC) بالترتيب , وأقطاب الخانة الثانية (A,B,C,D,E,F,G) مع أقطاب النافذة (PORTD) وذلك عبر مقاومات بقيمة (120 Ω) , كما ونقوم بوصل قطب المصاعد المشترك في الخانة الأولى إلى قطب التغذية (+5V) وقطب المهابط المشترك في الخانة الثانية إلى القطب الأرضي (GND) كما هو مُبين بالمخطط التالي :



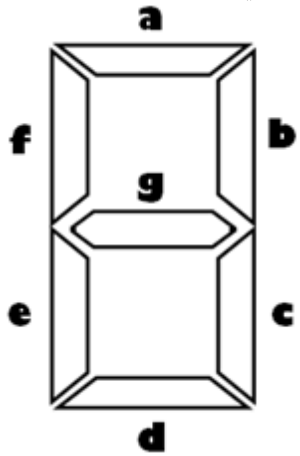
الآن لقيادة هاتين الخانتين علينا استخراج شيفرات الإظهار لكل رقم والتي من خلالها سيتم إظهار الرقم على الخانة .

في الخانة الأولى لتشغيل ليد قطعة ما نقوم بإرسال (0) منطقي على قطب المتحكم المتصل مع قطب هذه القطعة ولإطفائه نرسل (1) منطقي (خانة من النوع مصعد مشترك) , و وفقاً لذلك نكتب شيفرات إظهار الأرقام للخانة الأولى ضمن الجدول التالي :



Hex	PC7	g PC6	f PC5	e PC4	d PC3	c PC2	b PC1	a PC0	شكل الرقم على الشاشة	الرقم
40	0	1	0	0	0	0	0	0		0
79	0	1	1	1	1	0	0	1		1
24	0	0	1	0	0	1	0	0		2
30	0	0	1	1	0	0	0	0		3
19	0	0	0	1	1	0	0	1		4
12	0	0	0	1	0	0	1	0		5
02	0	0	0	0	0	0	1	0		6
78	0	1	1	1	1	0	0	0		7
00	0	0	0	0	0	0	0	0		8
10	0	0	0	1	0	0	0	0		9

أما في الخانة الثانية لتشغيل ليد قطعة ما نقوم بإرسال (1) منطقي على قطب المتحكم المتصل مع قطب هذه القطعة ولإطفائه نرسل (0) منطقي (خانة من النوع مهبط مشترك) , و وفقاً لذلك نكتب شيفرات إظهار الأرقام للخانة الثانية ضمن الجدول التالي :



Hex	PD7	g PD6	f PD5	e PD4	d PD3	c PD2	b PD1	a PD0	شكل الرقم على الشاشة	الرقم
3F	0	0	1	1	1	1	1	1		0
06	0	0	0	0	0	1	1	0		1
5B	0	1	0	1	1	0	1	1		2
4F	0	1	0	0	1	1	1	1		3
66	0	1	1	0	0	1	1	0		4
6D	0	1	1	0	1	1	0	1		5
7D	0	1	1	1	1	1	0	1		6
07	0	0	0	0	0	1	1	1		7
7F	0	1	1	1	1	1	1	1		8
6F	0	1	1	0	1	1	1	1		9

نستطيع الآن وبعد الحصول على شيفرات إظهار الأرقام على هاتين الخانتين كتابة برنامج لإرسال هذه الشيفرات عبر أقطاب المتحكم الصغري لإظهار الأرقام المختلفة على الخانتين السابقتين .

كتابة البرنامج :

نقوم بكتابة برنامج بلغة (BASCOM AVR) للمتحكم (ATmega16) المتوضع في مخطط الدارة السابقة بحيث يقوم هذا البرنامج بإظهار الأرقام المختلفة (من 0 إلى 9) بالترتيب وبشكل متوازي على هاتين الخانتين وبفاصل زمني مقداره (1 Sec) بين عملية الإظهار والأخرى :

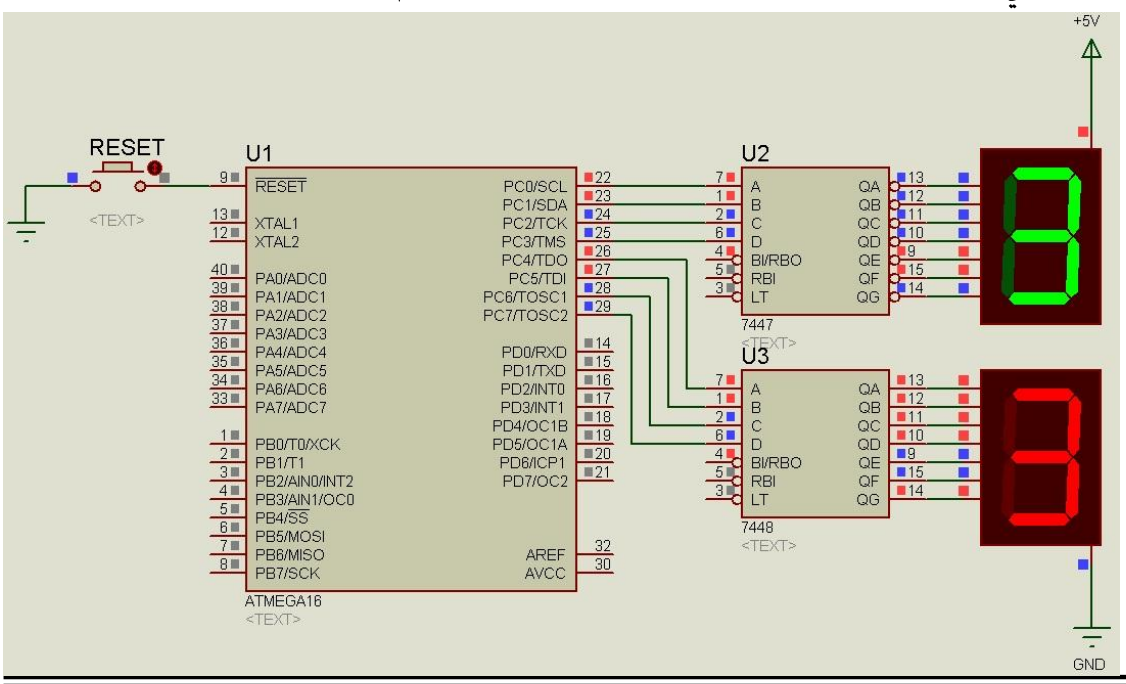
```
$regfile = "m16def.dat"
$crystal = 4000000

Config Portc = Output
Config Portd = Output

Dim Digit As Byte
Digit = 0

Do
'*****************************************************
Select Case Digit

Case 1:
Portc = &H79
Portd = &H06

Case 2:
Portc = &H24
Portd = &H5B

Case 3:
Portc = &H30
Portd = &H4F

Case 4:
Portc = &H19
Portd = &H66

Case 5:
Portc = &H12
Portd = &H6D

Case 6:
Portc = &H02
Portd = &H7D

Case 7:
Portc = &H78
Portd = &H07

Case 8:
Portc = &H00
Portd = &H7F
```

```
Case 9:
Portc = &H10
Portd = &H6F

Case Else
Digit = 0
Portc = &H40
Portd = &H3F

End Select
'*****************************************************
Wait 1

Digit = Digit + 1

Loop
End
```

الميزات والعيوب :

ليس لهذه الطريقة ميزات إلا إذا أخذنا بعين الاعتبار أنها لا تحتاج لعناصر إضافية في الدارة , فهي غير مستخدمة في الحياة العملية إذ أنه من غير المعقول استهلاك أقطاب نافذة متحكم صغري كاملة من أجل قيادة خانة واحدة من شاشة (SSD) كما أن وصل قطب المتحكم مع مصعد أو مهبط الليد بشكل مباشر يُعرضه للعطب نتيجة مرور تيار زائد عبره , كل هذه الأسباب وأسباب أخرى تجعل من هذه الطريقة طريقة مستبعدة التطبيق في الحياة العملية .

ملاحظة :يُمكن اختصار عدد أقطاب المتحكم التي تقود خانة شاشة (SSD) بطريقة الوصل المباشر وذلك عبر وصل دارة متكاملة (IC) بين المتحكم والخانة حيث تتولى هذه الـ(IC) عملية تحويل شيفرة الرقم المُدخل إليها (BCD Number) إلى الشيفرة التي تُظهره على خانة شاشة السبع قطع , ويتوفر رقمان من هذه الـ(IC) هما : الدارة المتكاملة (7447) لشاشة المصعد المشترك . والدارة المتكاملة (7448) لشاشة المهبط المشترك , ويُبين المخطط التالي طريقة وصل هذه الدارة المتكاملة بين المتحكم وخانة (SSD) :



وظيفة

اكتب برنامج المتحكم (ATmega16) لمخطط الوصل المُبين جانباً بحيث تقوم الدارة بنفس وظيفة الدارة السابقة

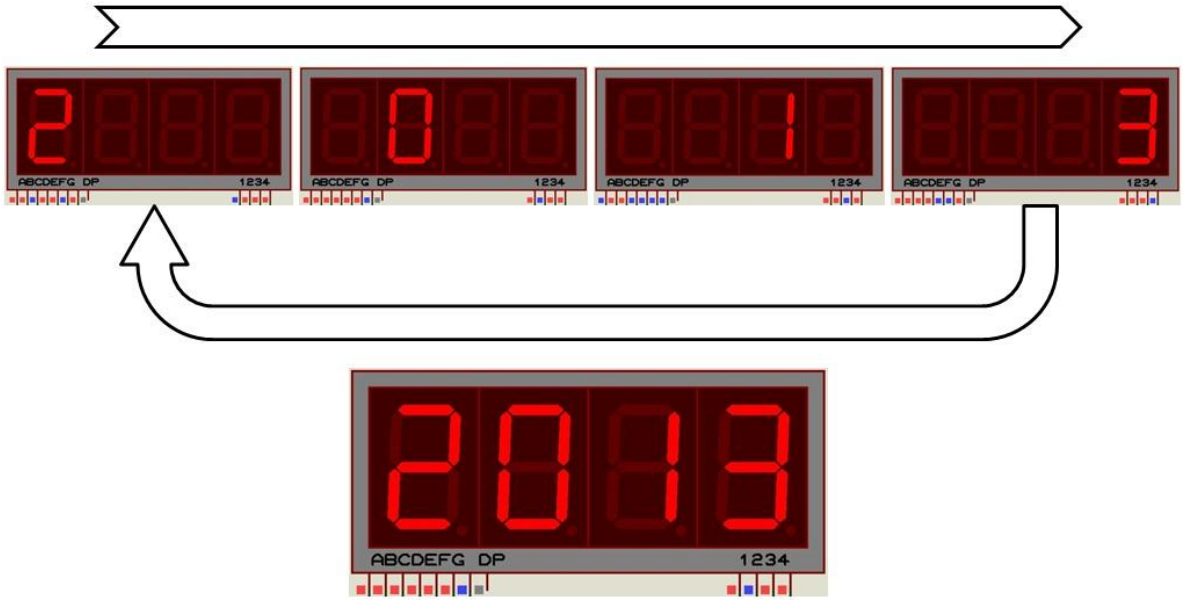
ثانياً : نمط المسح السريع :

في الطريقة السابقة وجدنا أنه لقيادة خانة واحدة من شاشة (SSD) يستلزم ذلك وصل سبعة أقطاب خرج من المتحكم الصغري وفي أحسن الأحوال أربعة أقطاب (باستخدام الدارة (7447)) , هذا يعني أنه لقيادة أربع خانات (SSD) وفق الطريقة السابقة نحتاج إلى (16) قطب خرج من المتحكم الصغري (إذا افترضنا استخدام الدارة (7447)) وهذا يعني استهلاك كبير جداً لأقطاب المتحكم مما أوجب البحث عن طريقة أخرى .

تُستخدم طريقة المسح السريع عادةً لقيادة شاشات (SSD) المتعددة الخانات (خانتين أو أكثر) وهي طريقة عالمية ومُتّبعة بكثرة في الحياة العملية وتعتمد في عملها على الحقيقة العلمية التي تقول بأن العين البشرية لا تستطيع إدراك الحوادث التي تتكرر أمامها لعدد كبير من المرات (60 مرة مثلاً) وهو المبدأ ذاته المستخدم في شاشات الحاسب والتلفاز حيث تتكرر الصورة الواحدة على شاشة الحاسب لعدد كبير من المرات (80 أو 100 مرة مثلاً) مما يؤدي إلى انطباع صورة ثابتة للشاشة على شبكية العين بسبب عدم قدرتها على ملاحقة التغيرات السريعة في الصورة .

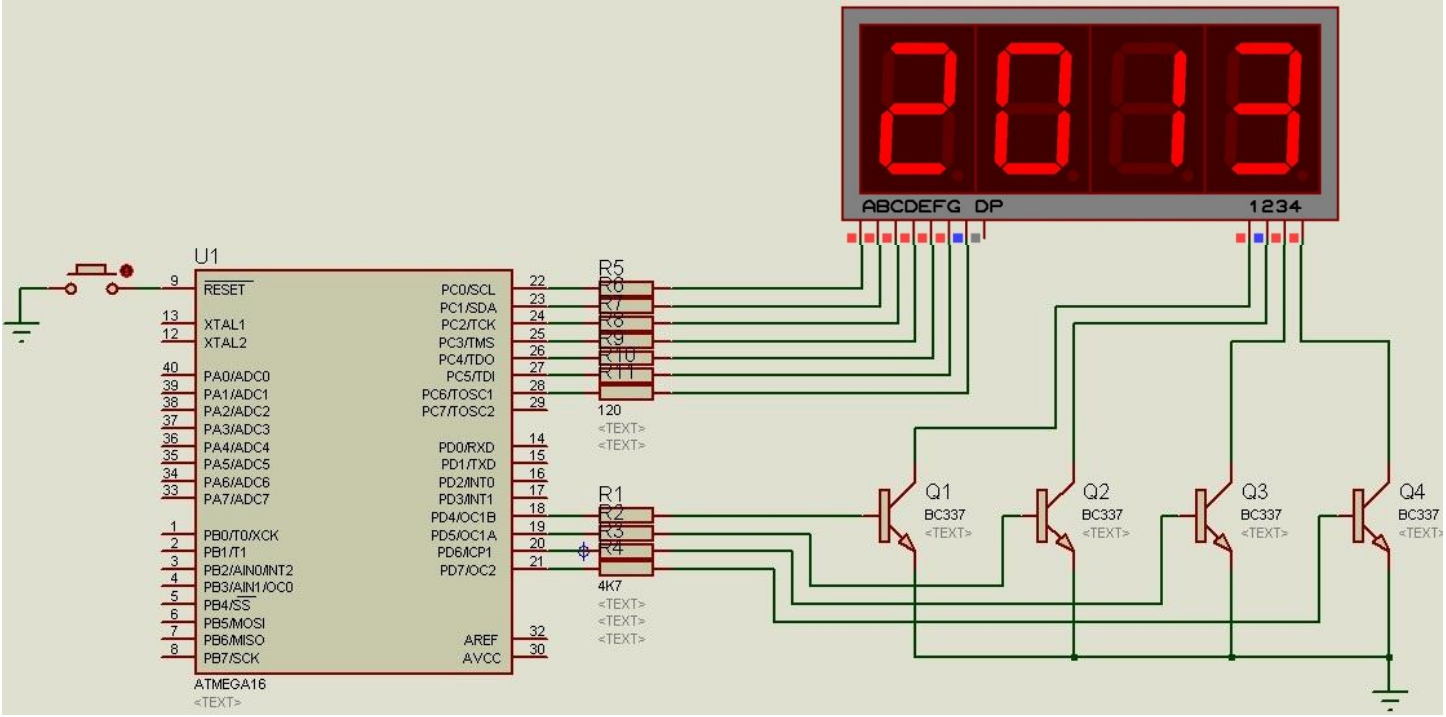
في هذه الطريقة نقوم بوصل أقطاب القطع المتقابلة من كل خانة ليشكلوا معاً قطباً واحداً مشتركاً للقطعة المطلوبة فمثلاً إذا كانت لدينا شاشة (SSD) بأربع خانات نقوم بوصل قطب القطعة (A) من كل خانة مع بعضها البعض ليتشكل لدينا قطب مشترك للقطعة (A) في الخانات الأربع وهكذا بالنسبة للقطع المتبقية (B,C,D,E,F,G) , عندها سيصبح لدينا سبعة أقطاب للشاشة (SSD) بالإضافة إلى أربعة أقطاب تفعيل كل قطب مسؤول عن تفعيل خانة من الخانات الأربع للشاشة .

بعد هذا التوصيل و وفقاً لهذه الطريقة فإننا نقوم بتفعيل الخانة الأولى من الشاشة فقط مع إلغاء تفعيل الخانات الثلاث الأخرى ونمرر المعطيات الخاصة بالخانة الأولى ومن ثم نفعّل الخانة الثانية ونلغي تفعيل جميع الخانات الأخرى ونمرر المعطيات الخاصة بالخانة الثانية وهكذا ..... وعند الانتهاء نعود للخانة الأولى ونكرر نفس العمل . حيث يكون لدينا في اللحظة الزمنية الواحدة خانة واحدة مفعلة فقط وبالتالي نستطيع تمرير المعطيات التي نريد إليها , وبسرعة مسح مناسبة تظهر للمستخدم شاشة (SSD) معروض عليها الأرقام الأربعة بشكل ثابت ومستقر .



مخطط التوصيل :

سنقوم بوصل شاشة (SSD) (مهبط مشترك) مؤلفة من أربع خانات مع المتحكم صغري (ATmrga16) حيث يلزمنا منه سبعة أقطاب خرج لوصلها مع ممر معطيات الشاشة (A,B,C,D,E,F,G) بالإضافة إلى أربعة أقطاب خرج لوصلها مع أقطاب تفعيل الشاشة , ولتجنب تعرّض أقطاب المتحكم لتيارات زائد نقوم باستخدام ترانزستورات (NPN) كوسيط بين أقطاب المتحكم وأقطاب تفعيل الشاشة فيصبح لدينا المخطط على الشكل التالي :



كتابة البرنامج :

نقوم بكتابة برنامج بلغة (BASCOM AVR) للمتحكم (ATmega16) المتوضع في مخطط الدارة السابقة بحيث يقوم هذا البرنامج بإظهار العدد (2013) على خانات شاشة (SDD) وذلك بطريقة المسح السريع (Scanning) , في البداية علينا الاستعانة بالجدول السابق الخاص بشيفرات إظهار الأرقام على شاشة (SDD) من النوع (مهبط مشترك) , سنقوم بتخزين قيم هذا الجدول ضمن ذاكرة البرنامج للمتحكم (تُخزّن في ذاكرة البرنامج كمعطيات وليس كشيفرات تعليمات) ونقوم خلال البرنامج بقراءة هذه القيم من مكان تخزينها لنقوم بعرض الأرقام على شاشة (SDD) .

إن التعليمات المسؤولة عن تخزين قيم في ذاكرة البرنامج ومن ثم قراءتها منه مُلخصة بالجدول التالي :

شرح التعليمة	التعليمة
تخزين بيانات متنوعة (رقمية أو محرفية) في ذاكرة البرنامج (Flash Memory) وذلك عند اللافتة (Label_x) , حيث تمتلك كل قيمة منها دليل (Index) يأخذ أرقام متسلسلة تبدأ من الصفر (0) .	`Label_x:` `Data val_1,val_2,.....,  val_n`

جلب قيمة *رقميّة* من جدول مُخزّن في ذاكرة البرنامج عند اللافتة (label) ولها الدليل (index) ووضعها في المتحول (Var) .	**Var = Lookup(index,label)**
جلب قيمة *محرفيّة* من جدول مُخزّن في ذاكرة البرنامج عند اللافتة (label) ولها الدليل (index) ووضعها في المتحول (Var) .	**Var = Lookupstr(index,label)**
تُعيد هذه التعليمة دليل (Index) القيمة (value) من الجدول المتوضع عند اللافتة (label) بحيث تبحث عن هذا الدليل ضمن عدد من القيم يُحدد ضمن المتحول (Entries) وذلك ابتداءاً من القيمة الأولى في الجدول	**Index = Lookdown(value,label,Entries)**

البرنامج :

```
'********* تعريف نوع المتحكم وقيمة الكريستالة الموصولة معه *********
$regfile = "m16def.dat"
$crystal = 4000000
'************************************************************

'*************** تعريف أقطاب الدخل والخرج ضمن المتحكم **************
Config Portc = Output
Config Portd.4 = Output
Config Portd.5 = Output
Config Portd.6 = Output
Config Portd.7 = Output
'************************************************************

'***** تعريف المتحولات والتوابع والإجراءات اللازمة في البرنامج ******
Dim My_number As Integer
Dim S As String * 4
Dim S1 As String * 1
Dim I As Byte
Dim Digit(4) As Byte

Declare Sub Display_on_7seg(number As Byte , Dis_num As Byte)
'************************************************************

'************ تهيئة المتحولات المطلوبة بالقيم الابتدائية ************
My_number = 2013
'************************************************************

'********************* حلقة البرنامج الرئيسية ***********************
Do

S = Str(My_number)
S = Format(s , "0000")
```

```
For I = 1 To 4
S1 = Mid(s , I , 1)
Digit(i) = Val(s1)
Next I

For I = 1 To 4
Call Display_on_7seg(digit(i) , I)
Waitms 2
Next I

Loop
End
'****************** نهاية حلقة البرنامج الرئيسية ******************


'************ Display_on_7Seg() كتابة تعليمات الإجراء ***************

Sub Display_on_7seg(number As Byte , Dis_num As Byte)

 Dim X As Byte

 Reset Portd.4
 Reset Portd.5
 Reset Portd.6
 Reset Portd.7

 X = Lookup(number , 7seg_cc)
 Portc = X

 Select Case Dis_num

 Case 1:
 Set Portd.4

 Case 2:
 Set Portd.5

 Case 3:
 Set Portd.6

 Case 4:
 Set Portd.7

 End Select

 End Sub
'*****************************************************************

7seg_cc:
Data &H3F , &H06 , &H5B , &H4F , &H66 , &H6D , &H7D , &H07 , &H7F , &H6F
```

الميزات والعيوب :

تُعتبر هذه الطريقة طريقة عالمية في قيادة شاشات (SSD) وهي مُستخدمة بكثرة في الأجهزة الالكترونية والكهربائية وتمتاز بأنها لا تحتاج لعناصر إضافية في الدارة كما أن استقرار العرض فيها جيد بالنسبة للشاشات التي تمتلك عدد خانات متوسط (من 2 إلى 6 خانات) إلا أن هذه الطريقة تملك بعض العيوب نذكر منها :

- عند زيادة عدد الخانات المُقادة بواسطة هذه الطريقة لعدد كبير تصبح عملية المسح مُلاحظة ويمكن أن يشعر المستخدم برفّة بسيطة عند عرض الخانات .
- نلاحظ من البرنامج أن عملية المسح يجب أن تكون دائمة ومستمرة ولا يقطعها أي تأخير زمني كبير أو حلقة تعليمات طويلة التنفيذ فعندها سيلاحظ المستخدم وجود وميض في عرض الخانات نتيجة لانقطاع عملية المسح . ويمكن حل هذه المشكلة بوصل ماسك (Latch) بين أقطاب المتحكم ومر معطيات شاشة (SSD) .

ثالثاً : طريقة مسجلات الإزاحة :

وهي أفضل طريقة لقيادة خانات شاشة (SSD) وخصوصاً إذا كان عدد هذه الخانات كبيراً , حيث يتم فيها نقل بيانات عرض الرقم على الخانة تسلسلياً عن طريق مسجل إزاحة يلعب دور الوسيط بين المتحكم و شاشة (SSD) . وسنستخدم هنا مسجل الإزاحة (74HC595) الذي يمتاز بوجود ماسك (Latch) على أقطاب خرجه مع إمكانية التحكم بلحظة تخريج هذه البتات على أقطابه التفرعيّة .

مسجل الإزاحة (74HC595) :

هو عبارة عن دارة متكاملة تقوم بتخريج البتات الواردة إليها تسلسلياً على أقطاب خرجها التفرعيّة . ويُبيّن الشكل التالي المخطط المنطقي لهذه الدارة المتكاملة بالإضافة إلى شكل توضع الأقطاب فيها :

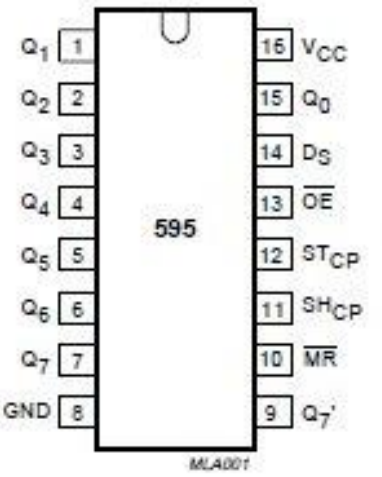


Fig.1 Pin configuration.

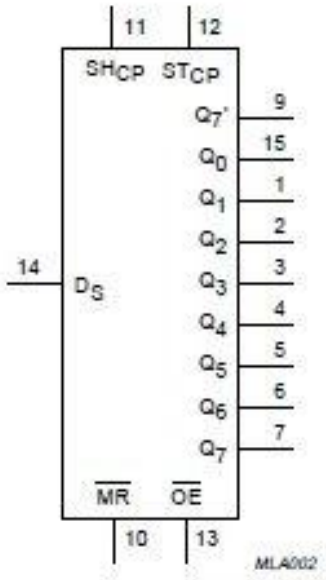
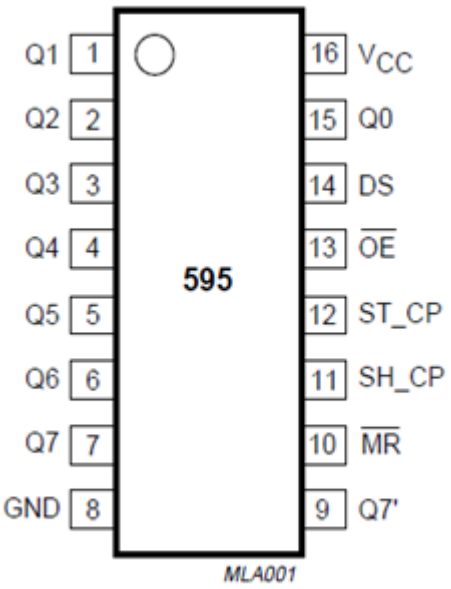


Fig.2 Logic symbol.

كما يُبيّن الشكل التالي وظيفة كل قطب من أقطاب هذه الدارة المتكاملة :



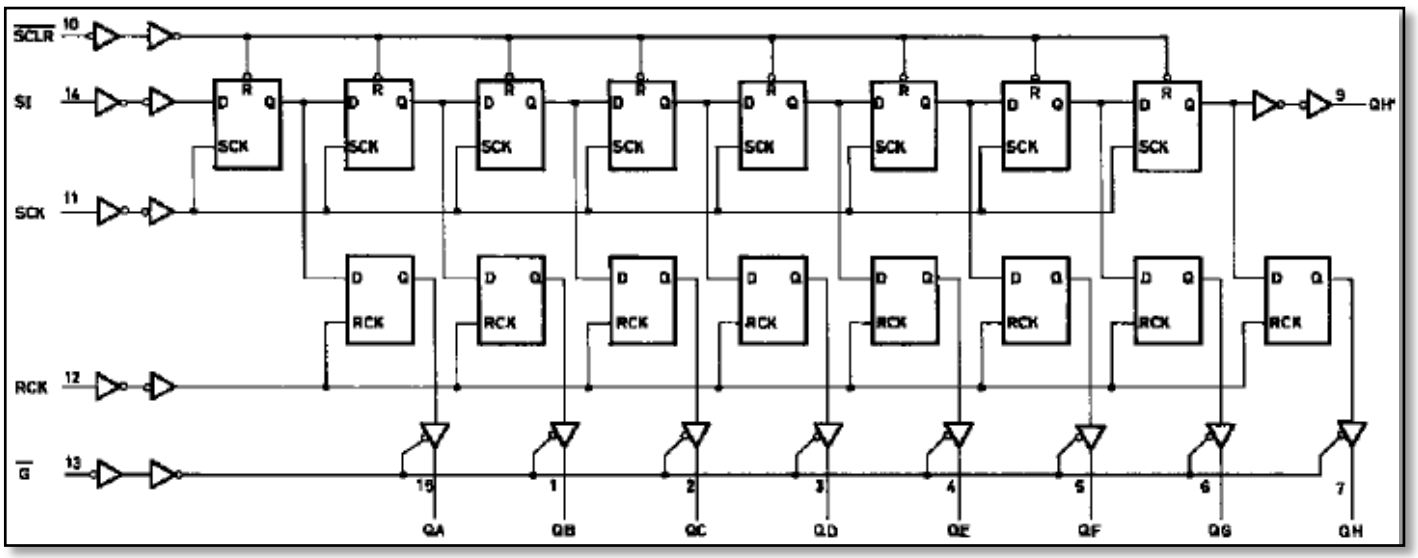
PINNING

PIN	SYMBOL	DESCRIPTION
1	Q1	parallel data output
2	Q2	parallel data output
3	Q3	parallel data output
4	Q4	parallel data output
5	Q5	parallel data output
6	Q6	parallel data output
7	Q7	parallel data output
8	GND	ground (0 V)
9	Q7'	serial data output
10	$\overline{MR}$	master reset (active LOW)
11	SH_CP	shift register clock input
12	ST_CP	storage register clock input
13	$\overline{OE}$	output enable (active LOW)
14	DS	serial data input
15	Q0	parallel data output
16	V_{CC}	positive supply voltage

تتألف هذه الدارة من مرحلتين من القلابات من النوع (D) , بحيث تحتوي كل مرحلة ثمانية قلابات متصلة مع بعضها البعض بشكل متسلسل (خرج القلاب الأول متصل مع دخل القلاب الثاني و خرج القلاب الثاني متصل مع دخل القلاب الثالث ..... وهكذا) , يتحكم قطب نبضات الساعة (SH_CP) بقلابات المرحلة الأولى حيث يقوم بنقل الحالة المنطقية المتوضعة على دخل كل قلاب إلى خرجه عند ورود نبضة صاعدة عليه .

أما القطب (ST_CP) فيتحكم بقلابات المرحلة الثانية ويعمل عند النبضة الصاعدة أيضاً , حيث يقوم بتخريج البتات المتوضعة على مخارج قلابات المرحلة الأولى إلى أقطاب خرج الدارة المتكاملة . وتُحافظ قلابات المرحلة الثانية على الحالة المنطقية للخرج إلى أن تأتي نبضة صاعدة على القطب (ST_CP) ليتم تخريج بتات جديدة وهكذا .... .

ويُبيّن الشكل التالي المخطط المنطقي الداخلي للدارة (74HC595) :



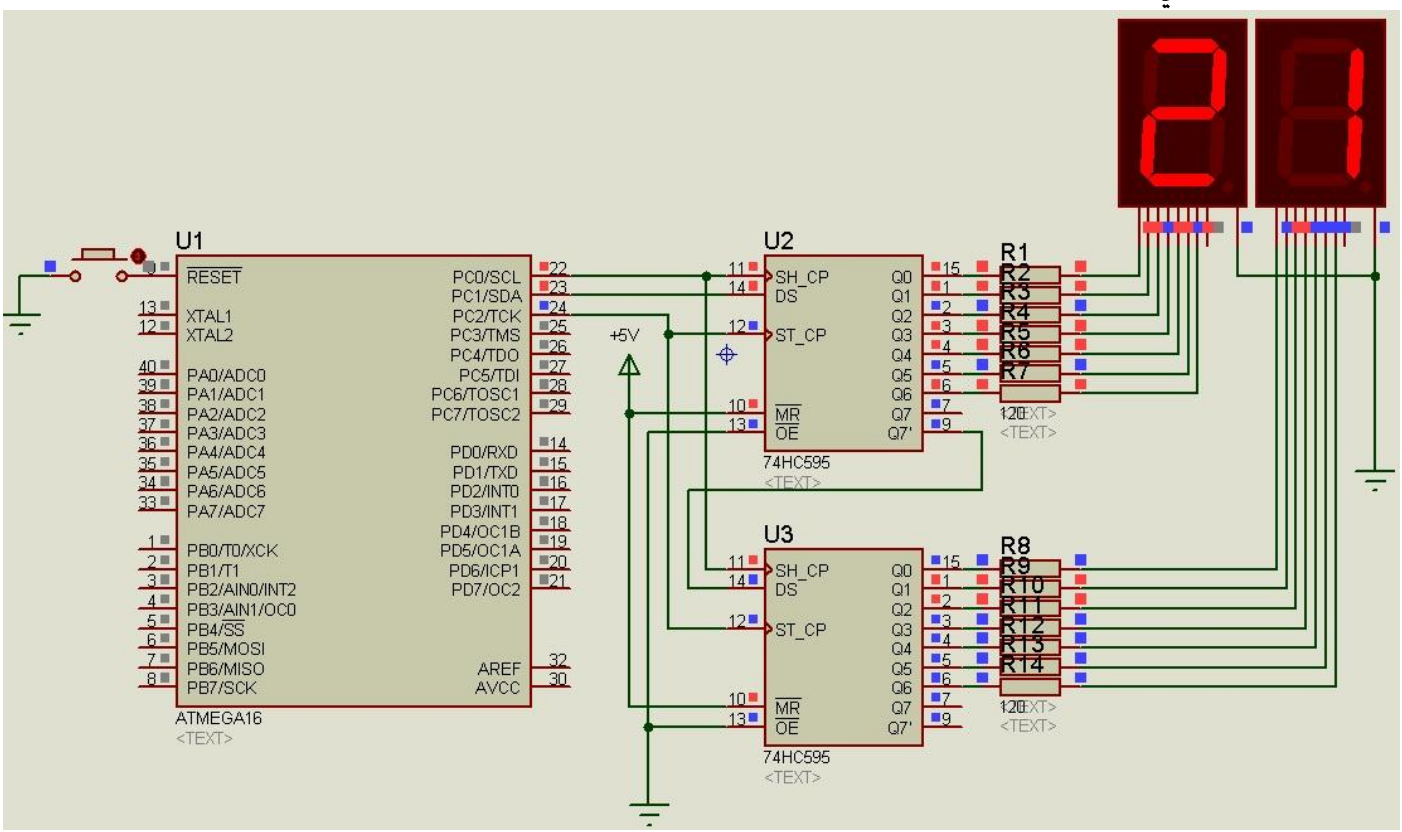
مخطط التوصيل :

سنقوم بوصل شاشتي (SSD) (مهبط مشترك) كل واحدة مؤلفة من خانة واحدة مع المتحكم الصغري (ATmega16) وذلك بواسطة مسجلي إزاحة (74HC595) وبحيث نقوم بوصل خرج الدارة الأولى مع دخل الدارة الثانية كما نصل القطبين (SH_CP) من كلا الدارتين مع بعضها البعض وكذلك القطبين (ST_CP) ينطبق عليهما نفس الأمر .

الآن كل ما يلزمنا من أقطاب المتحكم ثلاثة أقطاب فقط :

- قطب خرج يتصل بدخل الدارة المتكاملة الأولى (DS) ليقوم المتحكم عبره بإرسال بتات بيانات الخانتين تسلسلياً .
- قطب خرج يتصل بقطب نبضات الساعة (SH_CP) وهو مسؤول عن إعطاء نبضة إزاحة البتات (نبضة صاعدة) داخل قلابات الدارتين .
- قطب خرج يتصل بقطب نبضات الساعة (ST_CP) وهو مسؤول عن إعطاء نبضة تخريج البتات (نبضة صاعدة) على أقطاب خرج الدارة المتكاملة .

يُبيّن الشكل التالي مخطط التوصيل :



نلاحظ من هذا المخطط وجود قطبين في الدارة المتكاملة (74HC595) هما (MR و OE) ونلاحظ أن هذين القطبين فعّالين عند (0) منطقي :

القطب (MR) (Master Reset) : وهو قطب تصفير الدارة حيث أنه لدى تطبيق (0) منطقي على هذا القطب تُصفّر جميع مخارج الدارة (74HC595) بغض النظر عن القيم المتوضعة على دخلها التسلسلي .

القطب (OE) (Output Enable) : وهو قطب تمكين مخارج الدارة حيث أنه لدى تطبيق (0) منطقي على هذا القطب تُصبح مخارج الدارة (74HC595) جاهزة للعمل وإلاّ فإنها تُحجب عن العمل.

البرنامج :

نقوم بكتابة برنامج بلغة (BASCOM AVR) للمتحكم (ATmega16) المتوضع في مخطط الدارة السابقة بحيث يقوم هذا البرنامج بتشكيل عدّاد بخانتين يعد الأرقام تلقائيّاً من (00) إلى (99) وذلك بفاصل زمني مناسب .

هناك عدة طُرق لقيادة الدارة المتكاملة (74HC595) إلا أنها تعتمد جميعاً نفس المبدأ وسنستخدم في هذا البرنامج تعليمة (Shiftout) التي تقوم بكامل خوارزمية عملية الإزاحة على القطب (SH_CP) (إدخال البتات تسلسلياً تمهيداً إخراجها بشكل تفرعي) ولا يتبقى إلا إرسال نبضة صاعدة على القطب (ST_CP) ليتم تخريج هذه البتات بشكل تفرعي على أقطاب الدارة (74HC595) .

التعليمة (Shiftout) :

تمتلك هذه التعليمة الشكل العام التالي :

```
Shiftout  Serial_Output_Pin , Clock_Pin , Wanted_Var , Option
[, bits ,delay]
```

حيث أنّ :

- **Serial_Output_Pin** : هو قطب الخرج من المتحكم الذي سيتم إرسال بتات المعطيات التسلسليّة عبره .
- **Clock_Pin** : هو قطب الخرج من المتحكم الذي سيتولى عملية إرسال نبضات الإزاحة .
- **Wanted_Var** : هو المتحول المُراد إرسال بتاته تسلسلياً لتتحوّل تفرعياً فيما بعد ويمكن أن يكون بايت أو كلمة أو ..... الخ .
- **Option** : هنا نكتب رقماً (0 or 1 or 2 or 3) وتُعبّر هذه الأرقام عن الخيارات التالية :

(0) : في هذه الحالة سيتم إرسال الخانة الأكثر أهمية (MSB) **أولاً** من المتحول المُراد إرساله (Wanted_Var) , والإرسال يتم عند النبضة **الهابطة** للقطب (Clock_Pin) .

(1) : في هذه الحالة سيتم إرسال الخانة الأكثر أهمية (MSB) **أولاً** من المتحول المُراد إرساله (Wanted_Var) , والإرسال يتم عند النبضة **الصاعدة** للقطب (Clock_Pin) .

(2) : في هذه الحالة سيتم إرسال الخانة الأقل أهمية (LSB) **أولاً** من المتحول المُراد إرساله (Wanted_Var) , والإرسال يتم عند النبضة **الهابطة** للقطب (Clock_Pin) .

(3) : في هذه الحالة سيتم إرسال الخانة الأقل أهمية (LSB) **أولاً** من المتحول المُراد إرساله (Wanted_Var) , والإرسال يتم عند النبضة **الصاعدة** للقطب (Clock_Pin) .

- **Bits** (اختياري) : وهو يعبّر عن عدد البتات التي سيتم إرسالها من خلال هذه التعليمة وهو بارامتر اختياري عند عدم ذكره يعتمد عدد البتات المُرسلة على نوع المتحول المُراد إرساله (Wanted_Var) فإذا كان المتحول بايت مثلاً فيتم إرسال (8) بت وإذا كان كلمة يتم إرسال (16) بت وإذا كان كلمة مُضاعفة يتم إرسال (32) بت وهكذا ..... بمعنىً آخر في حال عدم ذكر هذا البارامتر يتم إرسال كامل بتات المتحول (Wanted_Var) .
- **delay** (اختياري) : وهو يُعبّر عن سرعة إرسال البتات (بالميكرو ثانية) وهو بارامتر اختياري عند عدم ذكره تكون سرعة نقل البتات (2 µS) (الزمن الفاصل بين إرسال البت والبت الذي يليه) .

يمكن الآن كتابة البرنامج لمخطط التوصيل السابق على الشكل التالي :

```
'********* تعريف نوع المتحكم وقيمة الكريستالة الموصولة معه *********

$regfile = "m16def.dat"
$crystal = 4000000

'************************************************************************

'*************** تعريف أقطاب الدخل والخرج ضمن المتحكم **************

Config Portc.0 = Output     'Shift Clock Pin
Config Portc.1 = Output     'Data Send Pin
Config Portc.2 = Output     'Latch Clock Pin

'************************************************************************

'***** تعريف المتحولات والتوابع والإجراءات اللازمة في البرنامج ******

Dim Digit As Byte
Dim I As Byte
Dim J As Byte

'************************************************************************

'********************* حلقة البرنامج الرئيسية ***********************
Do

   For I = 0 To 9
       For J = 0 To 9

           Digit = Lookup(j , 7seg_cc)
           Shiftout Portc.1 , Portc.0 , Digit , 1 , 8 , 200

           Digit = Lookup(i , 7seg_cc)
           Shiftout Portc.1 , Portc.0 , Digit , 1 , 8 , 200

           Reset Portc.2
           Waitms 2
           Set Portc.2

           Wait 1

       Next J
   Next I

Loop
End
'************************************************************************

7seg_cc:
Data &H3F , &H06 , &H5B , &H4F , &H66 , &H6D , &H7D , &H07 , &H7F , &H6F
```

الميزات والعيوب :

تُعتبر هذه الطريقة من أفضل الطرق في قيادة شاشات (SSD) من ناحية الأداء ومن ناحية استقرار العرض وخصوصاً إذا كان لدي عدد كبير من الخانات المُراد قيادتها بواسطة متحكم صغري واحد . عندها لن يلزمنا من المتحكم أكثر من ثلاثة أقطاب لقيادتها بالإضافة إلى دارة مسجل إزاحة (74HC595) لكل خانة من هذه الخانات .

لا تتأثر هذه الطريقة بانقطاع تعليمات العرض في البرنامج (لوجود تعليمة تأخير زمني مثلاً أو أن المتحكم يُنفذ حلقة تعليمات طويلة العدد) وذلك بسبب وجود الماسك على خرج مسجلات الإزاحة (74HC595) الذي يقوم بالمحافظة على آخر أرقام تم عرضها على الخانات إلى أن تأتي أرقام أخرى من المتحكم الصغري .

يُعاب على هذه الطريقة التكلفة المادية الكبيرة نسبيّاً لها , حيث أن مسجلات الإزاحة (74HC595) مصنوعة من ذواكر القلابات والتي عادةً ما تكون تكلفتها عالية مقارنة مع غيرها من الذواكر .

رابعاً : طريقة الدمج بين المسح السريع و مسجلات الإزاحة :

يمكن الدمج بين الطريقتين السابقتين بحيث ينتج لدينا دارة تستخدم أقل عدد ممكن من أقطاب المتحكم وبأقل تكلفة ممكنة .

وظيفة :

تصميم دارة الكترونية يقوم فيها المتحكم (Atmega16) بقيادة أربع خانات من شاشة (SSD) وذلك وفق هذه الطريقة (دمج الطريقتين السابقتين) مع كتابة برنامج القيادة بلغة (BASCOM AVR) وتصميم الدارة على برنامج (Proteus) .

الجلسة الرابعة

قيادة شاشة الإظهار الكريستاليّة المحرفيّة
(LCD Character Display)

تُعتبر شاشة الإظهار الكريستالية (LCD) من أشهر أدوات العرض المُستخدمة في الدارات الالكترونية عموماً وفي دارات التحكم خصوصاً , حيث أنها قادرة على عرض جميع رموز شيفرة الأسكي (ASCII Characters) من أرقام و أحرف و محارف خاصة .... الخ , كما أنها تستهلك تياراً منخفضاً جداً أثناء عملها (حوالي 3 ميلي أمبير) مما جعلها الحل الأنسب للعرض في الأجهزة التي تعتمد في تغذيتها على بطاريات بخلاف شاشة العرض ذات السبع قطع (SSD) التي تتألف من تجمّع عدد من الليدات الضوئية تستهلك تياراً كبيراً أثناء تشغيلها إذا ما قُورنت بشاشة الإظهار الكريستالية (LCD) .

بخلاف شاشة الإظهار ذات السبع قطع (SSD) ذات البنية البسيطة فإن شاشة الإظهار الكريستاليّة (LCD) هي شاشة معقّدة تأتي مزروعةً على دارة الكترونيّة مطبوعة تحتوي هذه الدارة على شريحة ذاكرة و شريحة معالج إظهار يتولى تطبيق عملية الإظهار على الشاشة . حيث أنّ عملية الإظهار فيها عملية معقدة تحتاج إلى تخصيص معالج صغري كامل لها إذ يتلقّى هذا المعالج أوامر الإظهار من الوسط الخارجي عن طريق أقطاب الشاشة ويقوم بتطبيقها على أرض الواقع .

هناك أنواع كثيرة من شاشات الإظهار (LCD) نُميّز من ناحية الاستخدام نوعان أساسيان لها : شاشات الاستخدام الخاص (Custom LCD) و شاشات الاستخدام العام أو الشاشات المعيارية (Standard LCD) . حيث أنه في النوع الأول يتم تصنيع الشاشة بحيث تعمل لصالح منتج خاص بعينه ولا يمكن أن تعمل الشاشة على منتج آخر وأما النوع الثاني فهي الشاشات المعياريّة التي يمكن وصلها مع أي متحكم صغري وقيادتها لإظهار المحارف عليها وهي موضوع دراستنا في هذه الجلسة بمشيئة اللّه تعالى.

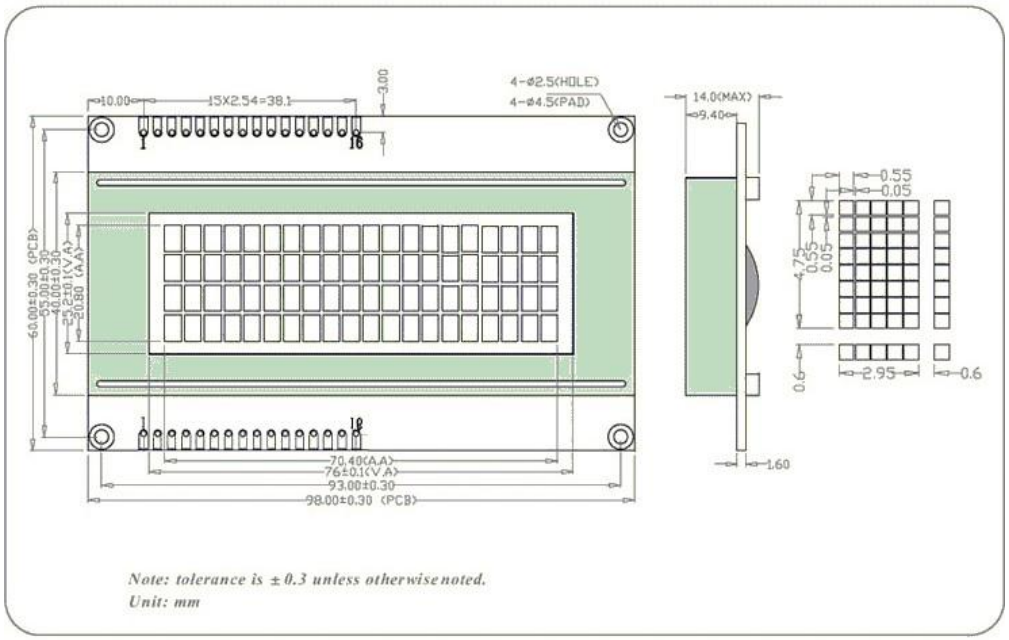
شاشات الإظهار المعيارية

شاشات الإظهار الخاصة

تُقسم شاشات الإظهار الكريستالية (LCD) من ناحية القدرة على الإظهار إلى نوعين :

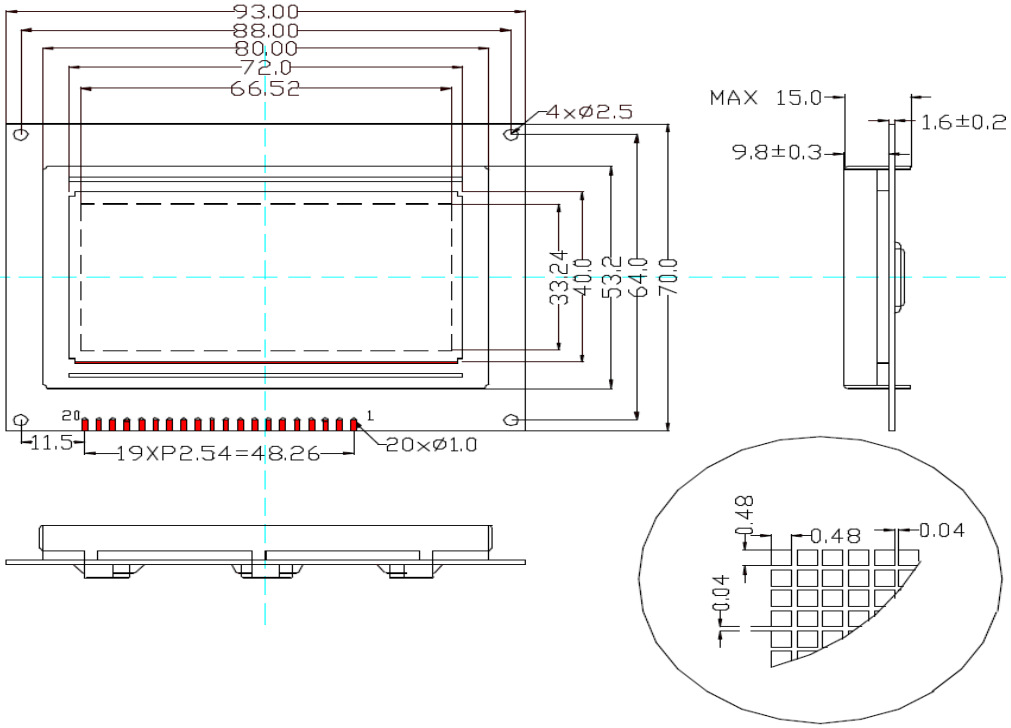
- **شاشة الإظهار المحرفيّة (LCD Character Display) :**

وتتألف من مصفوفة من الخانات (أسطر وأعمدة) تتألف فيها كل خانة عادةً من (5×8) بكسل تنفصل هذه الخانات عن بعضها البعض بفواصل وبالتالي لا يمكن إظهار الرسوم عليها , ويتولى معالج الشاشة إظهار المحارف على الخانات (كل محرف على خانة) وذلك بحسب الأوامر الآتية له من المتحكم الصغري الذي يقود هذه الشاشة .



- **شاشة الإظهار الرسومية (LCD Graphic Display) :**

يتألف هذا النوع من الشاشات من مصفوفة من البكسلات (وليس الخانات) المتوضعة بعدد من الأسطر والأعمدة تتناسب مع أبعاد الشاشة . و تمتلك قدرة عالية على إظهار الرسوميات بالإضافة إلى النصوص والأرقام .



بنية شاشة الإظهار الكريستاليّة المحرفيّة (LCD) :

كما ذكرنا سابقاً يتألف هذا النوع من الشاشات من اجتماع مصفوفة من الخانات (أسطر×أعمدة) وتتوفر في الأسواق ضمن قياسات معيارية سنذكرها لاحقاً إلا أنها وعلى اختلاف قياساتها تمتلك بنية واحدة :

تتوضّع شاشة الإظهار الكريستاليّة (LCD) على دارة مطبوعة تحتوي بشكل أساسي على شريحة معالج و شريحة ذاكرة ويمكن أن تحتوي على شرائح أخرى إضافية .

يتولى معالج الإظهار عملية إظهار المحرف المطلوب على الشاشة حيث أنه يستقبل شيفرة المحرف بالأسكي على مر معطيات الشاشة ليقوم بإظهاره بالموضع المحدد على الشاشة .

تلعب شريحة الذاكرة دور الذاكرة المؤقتة (Buffer) لشاشة الإظهار (LCD) حيث أنها تحتفظ دوماً بشيفرات آخر المحارف المعروضة على الشاشة , فكل خانة من خانات الشاشة تُقابل حجرة من حجرات الذاكرة يتوضع فيها شيفرة المحرف المعروض على هذه الخانة و لا يتم محو هذا المحرف إلى أن يتم كتابة شيفرة محرف جديد في حجرة الذاكرة الخاصة به أو عند محو محارف الشاشة بالكامل (محي جميع حجرات الذاكرة).

يتوضع خلف الشاشة في معظم شاشات (LCD) ليد ضوئي مُسطّح (BackLight LED) بحيث يُمكن لدى تغذيته بتغذية مناسبة إنارة شاشة العرض لرؤية المحارف عليها في حال كانت الشاشة موضوعة في مكانٍ مظلم . ويُمكن أن تكون هذه الإضاءة بألوان مختلفة ليختلف معها لون العرض على الشاشة .

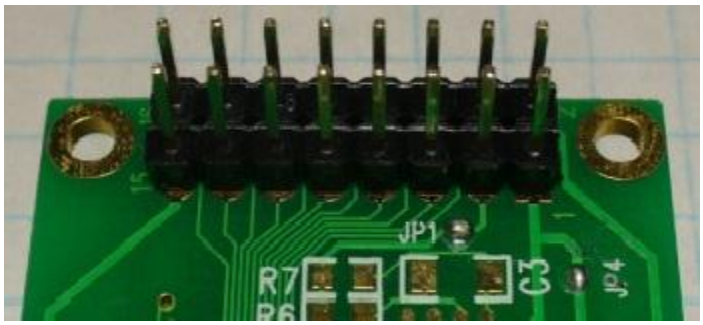
قياسات شاشة الإظهار الكريستاليّة المحرفيّة (LCD) :

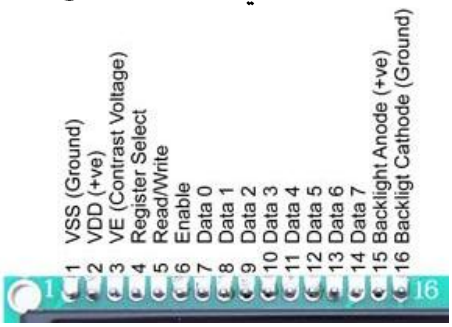
تُقاس شاشة الإظهار المحرفيّة (LCD) بعدد الأسطر والأعمدة الموجودة فيها , حيث أنها يمكن أن تتألف من سطر واحد أو سطرين أو أربعة أسطر . ومن ناحية الأعمدة يمكن أن يكون فيها (8) أعمدة أو (12) أو (16) أو (20) أو (24) أو (32) أو (40) عمود . و تُنتج شركات التصنيع حول العالم عدة قياسات معيارية من شاشة الإظهار (LCD) . سنستعرض في هذا الجدول القياسات العالمية المُنتجة من قِبل شركة (NewHaven) :

قياس الشاشة	صورة الشاشة
LCD (1×8)	
LCD (1×12)	
LCD (1×16)	
LCD (2×8)	
LCD (2×12)	
LCD (2×16)	
LCD (2×20)	
LCD (2×24)	
LCD (2×40)	
LCD (4×16)	
LCD (4×20)	
LCD (4×40)	

أقطاب شاشة الإظهار الكريستاليّة المحرفيّة (LCD) :

بشكل عام تمتلك شاشة الإظهار (LCD) ستة عشر قطباً بغض النظر عن قياساتها (عدد الأسطر والأعمدة) تتوضع هذه الأقطاب إما على صفٍّ واحد أفقيّاً أو على شكل صفّين متجاورين متوضعين بشكل عمودي وتُرقّم ابتداءاً من القطب الأول وحتى الأخير بشكل تسلسلي حيث يُكتب عادةً رقم القطب الأول و الأخير منها . ويُبيّن الشكل التالي كيفية توضع أقطاب شاشة (LCD) :





يُبيّن الجدول التالي أسماء أقطاب شاشة (LCD) مع شرح مُبسّط لكل قطب :

Pin	Symbol	Name	Description
1	VSS	Ground	0V (GND)
2	VDD	Power Supply	Power supply for logic circuit and LCD (+4.5V ~ +5.5V)
3	VEE	LCD Contrast Pin	Bias voltage level for LCD driving
4	RS	Register Select	Register select input. When RS =" High" , Data register is selected. When RS =" Low " , Instruction register is selected.
5	R/W	Read/Write	Read/Write selection input. When RW = " High " , Read operation. When RW = " Low " , Write operation.
6	E	Read/Write Enable	Start enable signal to read or write the data
7	DB0	Data Bus 0-7	DB0 - DB3 , In 8-bit bus mode, used as low order bi-directional data bus. During 4-bit bus mode , Open these pins
8	DB1		
9	DB2		
10	DB3		
11	DB4		DB4 - DB7, In 8-bit bus mode, used as high order bi-directional data bus . In case of 4-bit bus mode , used as both high and low order.
12	DB5		
13	DB6		
14	DB7		
15	A		BackLight Anode
16	K		BackLight Cathode

شرح الأقطاب :

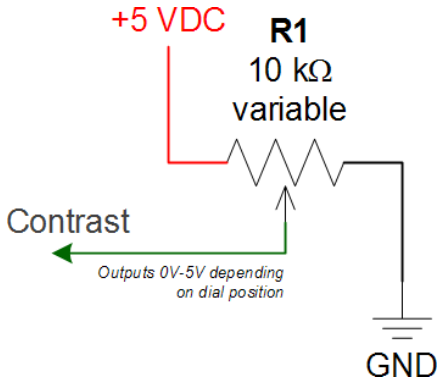
- *القطب رقم (1) (VSS) :*

قطب أرضي الشاشة ويتصل بالقطب الأرضي للدارة (GND) .

- *القطب رقم (2) (VDD) :*

قطب التغذية الموجبة للشاشة ويتصل بقطب تغذية الدارة (+5V) .

- *القطب رقم (3) (VEE) :*



قطب جهد التباين (Contrast) للشاشة (V_O) , ويقصد بالتباين حدة ظهور المحرف على الشاشة حيث أنّ أقل تباين أن لا نرى شيئاً على الشاشة ويكون لدى تطبيق (+5V) على هذا القطب , وأعلى تباين للشاشة يكون لدى وصل هذا القطب إلى الأرضي (0V) وعندها ستظهر المحارف على الشاشة بأعلى حدة ظهور لها .

ويمكن التحكم بتباين الشاشة عن طريق وصل هذا القطب إلى مقاومة متغيرة بقيمة (10KΩ) كما هو مُبيّن في الشكل المجاور :

- *القطب رقم (4) (RS) :*

قطب اختيار مسجل الدخل وهو قطب تحكم يتم عبره تحديد نوع البيانات المُرسلة عبر مر معطيات الشاشة وذلك وفق ما يلي :

عند تطبيق *(0) منطقي* على هذا القطب يتم اعتبار البيانات المُستقبلة عبر مر معطيات الشاشة على أنها *بايتات تحكم (Control Bytes)* .

عند تطبيق *(1) منطقي* على هذا القطب يتم اعتبار البيانات المُستقبلة عبر مر معطيات الشاشة على أنها *بايتات معطيات (Data Bytes)* .

- *القطب رقم (5) (R/W) :*

قطب تحديد القراءة أو الكتابة من أو على ذاكرة الشاشة حيث أنه :

عند تطبيق *(0) منطقي* على هذا القطب فهذا يعني أننا نريد *الكتابة* على ذاكرة الشاشة .

عند تطبيق *(1) منطقي* على هذا القطب فهذا يعني أننا نريد *القراءة* من ذاكرة الشاشة .

بشكل عام نقوم بوصل هذا القطب إلى القطب الأرضي (GND) بشكل دائم .

- *القطب رقم (6) (E) :*

قطب نبضة التمكين وهو قطب تحكم هام جداً في تنظيم عمل شاشة (LCD) , إذ أن أي عملية كتابة أو قراءة من ذاكرة الشاشة تحتاج بعدها إلى نبضة تمكين عند الجبهة الهابطة على هذا القطب لتأكيدها (نرفع هذا القطب إلى (1) منطقي ومن ثم نعيده إلى (0) بعد تأخير زمني مناسب) .

- *الأقطاب ذوو الأرقام (من (7) إلى (14)) (DB0 → DB7) :*

وهي أقطاب مر معطيات شاشة (LCD) وعددها (8) أقطاب تُستخدم جميعها كممر معطيات في حال كون الشاشة تعمل في نمط (8) بت وتُستخدم الأقطاب الأربعة العُليا منها فقط (DB4 → DB7) في حال كون الشاشة تعمل في نمط (4) بت .

- *القطب رقم (15) (A) :*

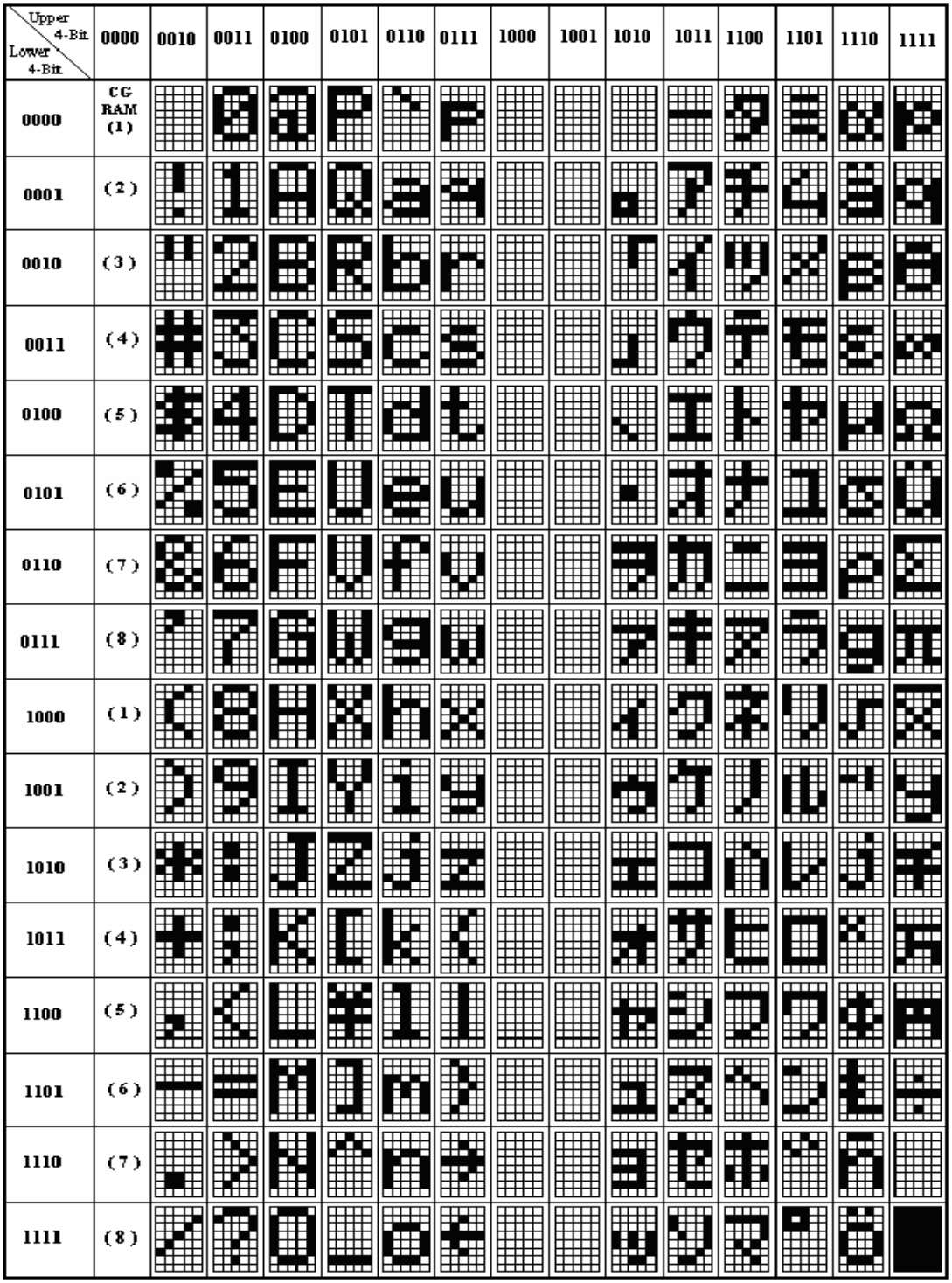
قطب مصعد الليد المُسطّح الخاص بالإضاءة الخلفية للشاشة .

- *القطب رقم (16) (K) :*

قطب مهبط الليد المُسطح الخاص بالإضاءة الخلفية للشاشة .

جدول شيفرة المحارف التي يُمكن إظهارها على شاشة (LCD) :

يُمكن عرض (189) محرفاً من شيفرة (ASCII) على شاشة الإظهار (LCD) وهي تتضمن أحرف الأبجدية الانكليزية واليابانيّة وبعض المحارف الأخرى الخاصة , وهي مُبيّنة جميعاً مع شيفرات إظهارها في الجدول التالي :

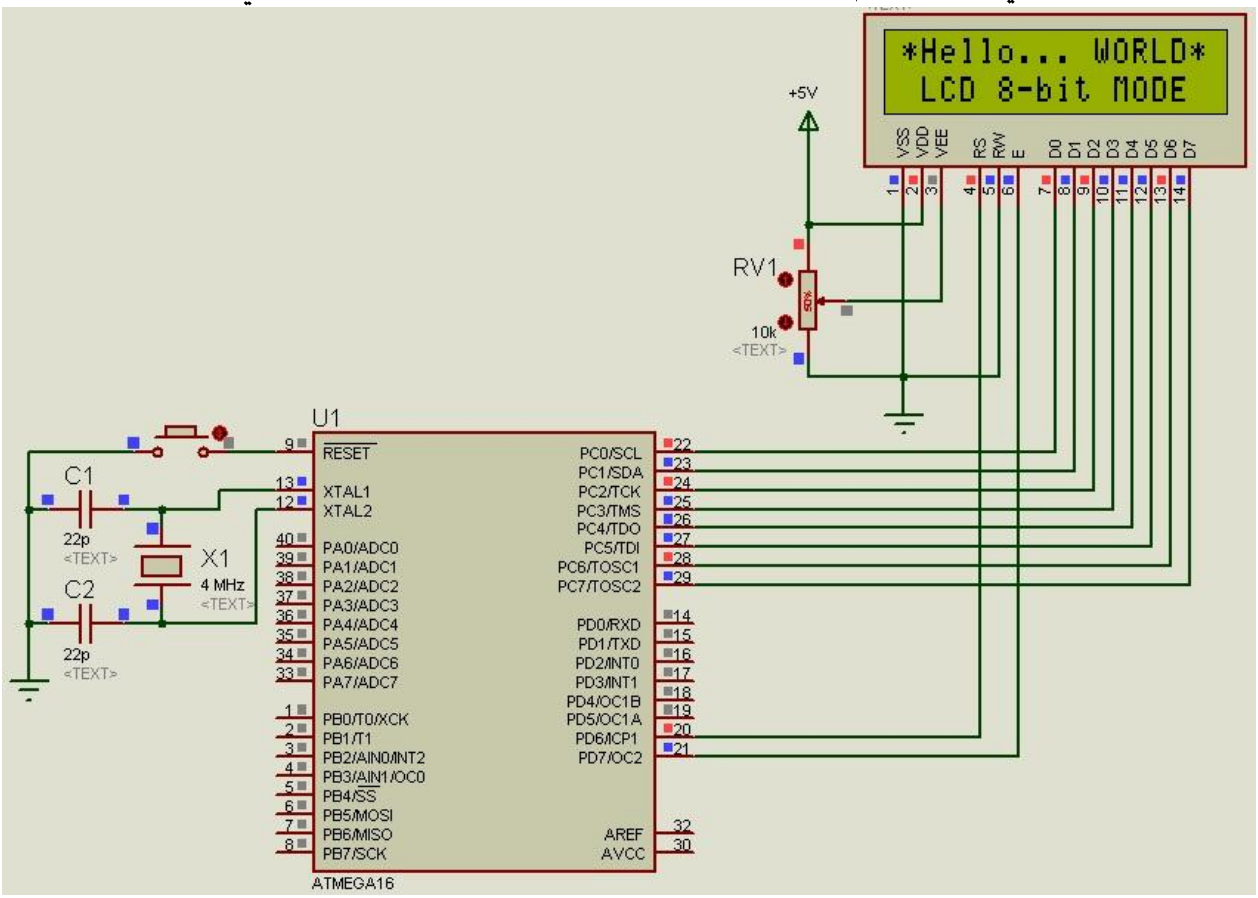


أنماط عمل شاشة (LCD) وطريقة توصيلها مع المتحكم في كل نمط :

تعمل شاشة الإظهار المحرفيّة (LCD) بأحد نمطي العمل (نمط (8)بت أو نمط (4)بت) ولكل نمط طريقة توصيل مختلفة مع المتحكم الصغري الذي يقود هذه الشاشة , وسنستعرض فيما يلي هذين النمطين :

- **نمط العمل (8) بت :**

في هذا النمط يتم توصيل كامل مر معطيات شاشة الإظهار (LCD) (DB0 → DB7) مع إحدى نوافذ الدخل/الخرج في المتحكم الصغري حيث يتم نقل بايت المعطيات / التحكم بين المتحكم وشاشة (LCD) دفعة واحدة على خطوط المعطيات الثمانية . كما ويتم توصيل قطبي التحكم (RS , E) إلى قطبي خرج من أقطاب المتحكم الصغري بينما نقوم بوصل قطب التحكم (R/W) إلى قطب الأرضي (GND) بشكل دائم .

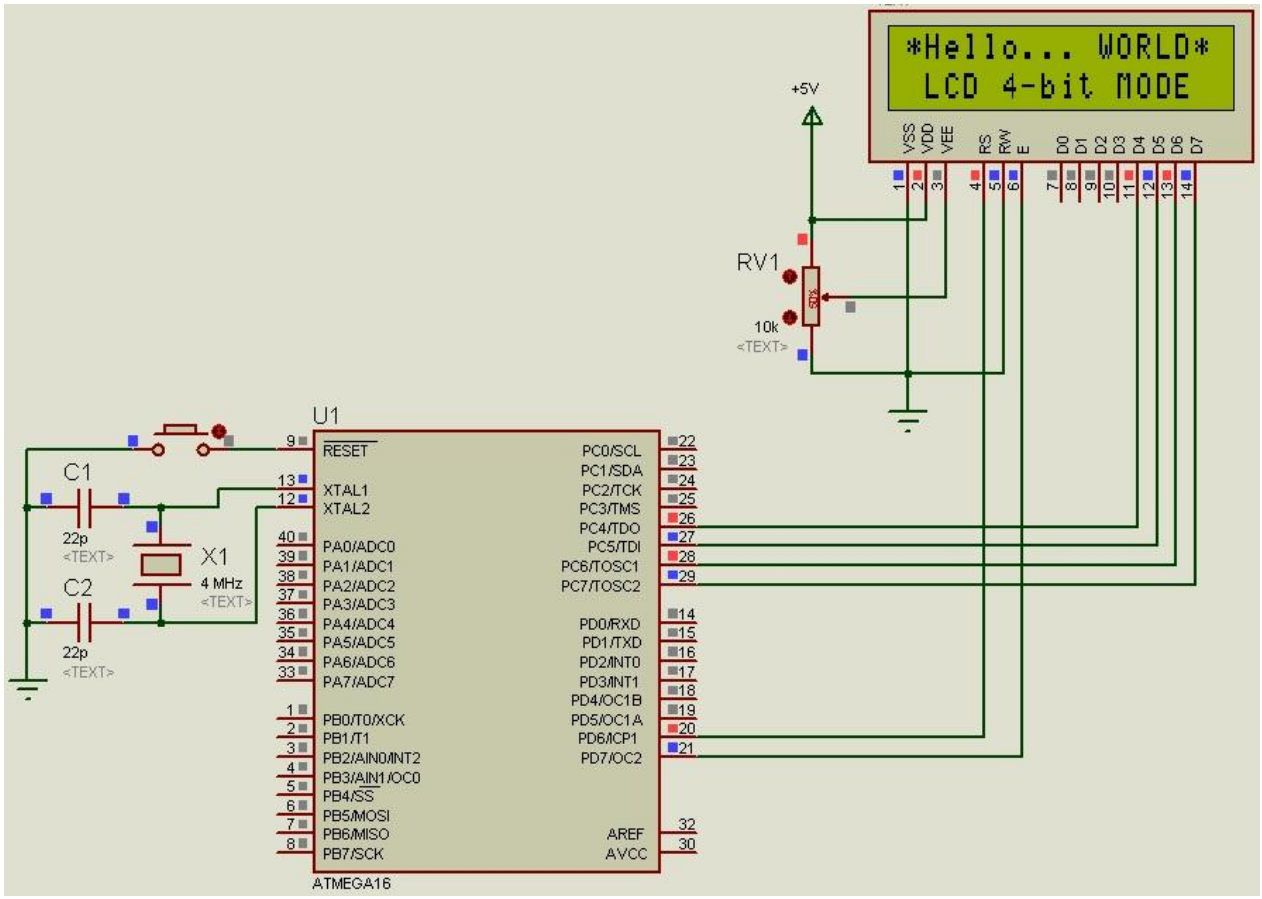


مما سبق نجد أنه يلزمنا وفق هذا النمط (10) أقطاب خرج من المتحكم لقيادة شاشة (LCD) واحدة .

- **نمط العمل (4) بت :**

في هذا النمط يتم توصيل الخطوط الأربعة العُليا فقط من مر معطيات شاشة الإظهار (LCD) (DB4 → DB7) مع أربعة أقطاب خرج في المتحكم الصغري حيث يتم نقل بايت المعطيات / التحكم بين المتحكم وشاشة (LCD) على دفعتين (يتم نقل النبل العلوي ثم السفلي) على خطوط المعطيات الأربعة . كما ويتم توصيل قطبي التحكم (RS , E) إلى قطبي خرج من أقطاب المتحكم الصغري بينما نقوم بوصل قطب التحكم (R/W) إلى قطب الأرضي (GND) بشكل دائم .

بشكل عام يُستخدم هذا النمط في قيادة شاشات (LCD) إذ يلزمنا فيه (6) أقطاب خرج من المتحكم لقيادة شاشة (LCD) واحدة ولا يُستخدم النمط السابق إلا نادراً لأن فيه إهداراً غير مبرراً لأقطاب المتحكم الصغري .



تعليمات التعامل مع شاشة (LCD) في لغة البرمجة (BASCOM AVR) :

إن قيادة شاشة الإظهار الكريستاليّة (LCD) عن طريق تعليمات الإسمبلي أمرٌ صعب ويستلزم كتابة أكواد طويلة جداً , فمجرد كتابة عبارة واحدة فيها يستلزم ربما أكثر من صفحتين من التعليمات .

بسّطت شركة (MCS Electronics) في مترجمها (BASCOM AVR) عملية قيادة شاشة (LCD) بشكل كبير حيث تتوفر تعليمات وإجراءات برمجيّة جاهزة لا تحتاج من المبرمج إلا أن يضعها في المكان الذي يُريد ليختصر بذلك صفحات كاملة من تعليمات الإسمبلي بتعليمة واحدة أو تعليمتين .

تُقسم تعليمات قيادة شاشة (LCD) في مترجم (BASCOM AVR) إلى قسمين رئيسيين :

- تعليمات التهيئة (Configuration Instructions) .
- تعليمات العرض (Display Instructions) .

أولاً : تعليمات التهيئة :

تحديد أبعاد شاشة (LCD) المتصلة مع المتحكم الصغري	`Config Lcd = 16 * 4`
تحديد أقطاب المتحكم المُتصلة بأقطاب شاشة (LCD) وذلك في نمط عمل (8-bit)	`Config Lcdpin = Pin , Port = Portc , E = Portd.7 , Rs = Portd.6`
تحديد أقطاب المتحكم المُتصلة بأقطاب شاشة (LCD) وذلك في نمط عمل (4-bit)	`Config Lcdpin = Pin , Db4 = Porta.7 , Db5 = Portb.3 , Db6 = Portd.0 , Db7 = Portc.4 , E = Porta.2 , Rs = Portc.6`

ثانياً : تعليمات العرض :

`Cls`	مسح كامل العرض على الشاشة وإعادة المؤشر إلى الخانة الأولى منها (الخانة التي هي في السطر الأول / العمود الأول)
`Locate X , Y`	وضع مؤشر الكتابة عند موضع مُحدد من الشاشة (X,Y) حيث أن (X) رقم السطر و (Y) رقم العمود .
`Lcd "You are WELCOME"`	طباعة العبارة التي بين الإشارتين " " على شاشة (LCD)
`Lcd Var`	طباعة قيمة المتحول (Var) على شاشة (LCD)
`Display Off`	إطفاء العرض على شاشة (LCD)
`Display ON`	تشغيل العرض على شاشة (LCD)
`Cursor ON [Blink or NoBlink]`	إظهار مُؤشر الكتابة على شاشة (LCD) مع وميض أو بدونه
`Cursor Off`	إخفاء مؤشر الكتابة على شاشة (LCD)
`Shiftlcd Right`	إزاحة *كامل العرض* على الشاشة خانة واحدة نحو *اليمين*
`Shiftlcd Left`	إزاحة *كامل العرض* على الشاشة خانة واحدة نحو *اليسار*
`Shiftcursor Right`	إزاحة *مُؤشر الكتابة* على الشاشة خانة واحدة نحو *اليمين*
`Shiftcursor Left`	إزاحة *مُؤشر الكتابة* على الشاشة خانة واحدة نحو *اليسار*
`Home`	إعادة المؤشر إلى الخانة الأولى من الشاشة (دون مسح الشاشة)
`Home Upper`	إعادة المؤشر إلى الخانة الأولى في السطر الأعلى من الموقع الحالي
`Home Lower`	إعادة المؤشر إلى الخانة الأولى في السطر الأدنى من الموقع الحالي
`Home Third`	نقل مؤشر الكتابة إلى بداية السطر الثالث
`Home Fourth`	نقل مؤشر الكتابة إلى بداية السطر الرابع
`UpperLine`	نقل مؤشر الكتابة إلى السطر الأعلى من السطر الحالي
`LowerLine`	نقل مؤشر الكتابة إلى السطر الأدنى من السطر الحالي
`ThirdLine`	نقل مؤشر الكتابة إلى السطر الثالث
`FourthLine`	نقل مؤشر الكتابة إلى السطر الرابع
`Deflcdchar S ,32,6,32,4,11,3,4,32`	تعريف شيفرة محرف جديد عن طريق الأداة (LCD Designer)
`Lcd Chr(S)`	عرض المحرف الجديد الذي تم إنشاؤه عن طريق التعليمة السابقة

وظيفة :

تصميم دارة الكترونية على برنامج (Proteus) فيها متحكم صغري (ATmega16) يقود شاشة (LCD) بأبعاد (16×4) وفق نمط العمل (4-bit) وكتابة برنامج على المترجم (BASCOM AVR) يقوم باستعراض تعليمات العرض الواردة في الجدول السابق .

الجلسة الخامسة

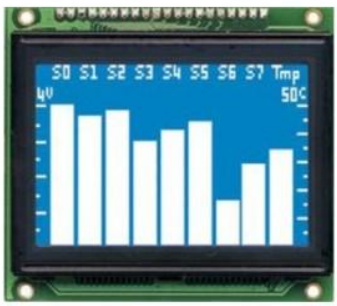
قيادة شاشة الإظهار الكريستاليّة الرسوميّة
(LCD Graphic Display)

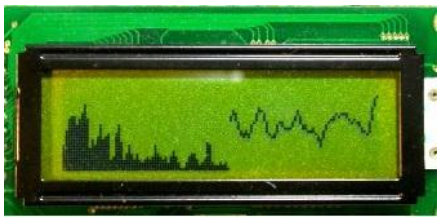
تعلمنا في الدرس السابق كيفيّة قيادة شاشة الإظهار الكريستاليّة المحرفيّة (LCD Character Display) و وجدنا أنها شاشة مخصّصة لإظهار المحارف فقط ولا يمكن إظهار الرسوم عليها بسبب وجود فواصل بين الخانات فيها حيث أن كل خانة مُخصصة لإظهار محرف واحد , وبالتالي لإظهار رسوميات معينة على شاشة (LCD) علينا استخدام شاشة الإظهار الكريستاليّة الرسوميّة (LCD Graphic Display) التي هي موضوع بحثنا في هذه الجلسة بمشيئة اللّه تعالى .

تمتلك شاشات الإظهار الرسومية قدرة كبيرة على العرض مقارناً بالشاشة المحرفيّة حيث أنها قادرة على عرض الرسوميّات بكافة أشكالها فضلاً عن عرض النصوص والمحارف بأحجام وأبعاد وأنماط مختلفة .

لا تختلف شاشة الإظهار الرسوميّة عن شاشة الإظهار المحرفيّة في تقنية العرض , حيث أن الاثنتين مصنوعتين بتقنية عرض واحدة , إلا أن شاشة الإظهار الرسومية مؤلفة من تجمّع مصفوفة من البكسلات تتوزع بشكل متساوي على طول وعرض الشاشة وبالتالي هي قادرة عن طريق إظهار بعضها وإخفاء بعضها الآخر على عرض الرسوميات والنصوص بشكل انسيابي يسمح بقدرة عرض عالية وبخاصة في شاشات الإظهار الرسوميّة الملونة .

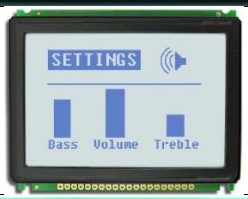
كما في الشاشات المحرفيّة تتواجد من الشاشات الرسوميّة شاشات مخصصة لمنتج معيّن لا يمكن استخدامها لأغراض أخرى و شاشات أخرى معياريّة يمكن وصلها مع أي متحكم صغري ليقوم بقيادتها وإظهار الرسوم والمحارف عليها .





قياسات شاشة الإظهار الكريستاليّة الرسوميّة (GLCD) :

تُقاس شاشة الإظهار الرسوميّة (GLCD) بعدد أسطر وأعمدة البكسلات الموجودة فيها , ويمكن أن تُصنع هذه الشاشات بأبعاد مختلفة ومتنوعة . و تُنتج شركات التصنيع حول العالم عدة قياسات معيارية من شاشات الإظهار الرسوميّة (GLCD) . سنستعرض في هذا الجدول القياسات العالمية المُنتجة من قِبل شركة (NewHaven) :

قياس الشاشة	صورة الشاشة
GLCD (32×122)	
GLCD (32×144)	
GLCD (32×160)	
GLCD (32×192)	
GLCD (64×128)	
GLCD (128×160)	
GLCD (128×240)	
GLCD (240×320)	

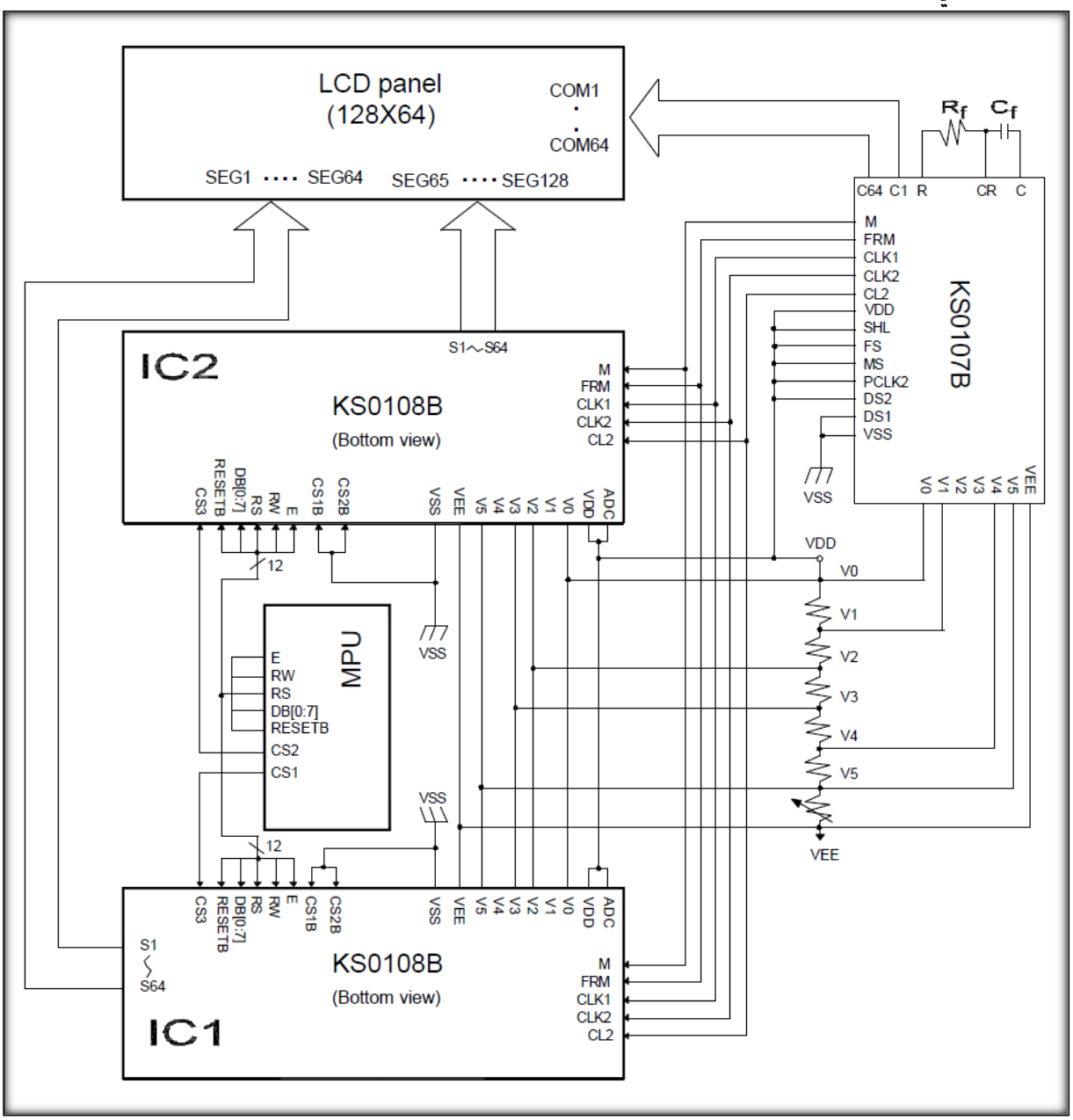
بنية شاشة الإظهار الكريستاليّة الرسوميّة (GLCD) :

تختلف شاشة الإظهار الكريستاليّة الرسومية اختلافاً كُليّاً في البنية عن شاشة الإظهار المحرفيّة وإن اتفقتا بتكنولوجيا العرض . حيث أن الشاشة الرسوميّة تأتي مزروعة (كما في الشاشة المحرفيّة) على دارة الكترونية مطبوعة تحتوي العديد من الدارات المتكاملة (ICs) التي تُؤمّن عمل الشاشة والتي تتألف بشكل رئيسي مما يلي :

- معالج صغري (MPU) لقيادة عملية الإظهار على الشاشة .
- دارات متكاملة لقيادة الأسطر (Rows Drivers) .
- دارات متكاملة لقيادة الأعمدة (Columns Drivers) .

ولتوضيح بنية الشاشات الرسوميّة بشكل أكبر نأخذ بنية الشاشة الرسومية ذات الأبعاد (128×64) وندرسها كمثال عن بنية هذه الشاشات :

يُبيّن الشكل التالي بنية الشاشة الرسومية (GDM12864A-LCM) :



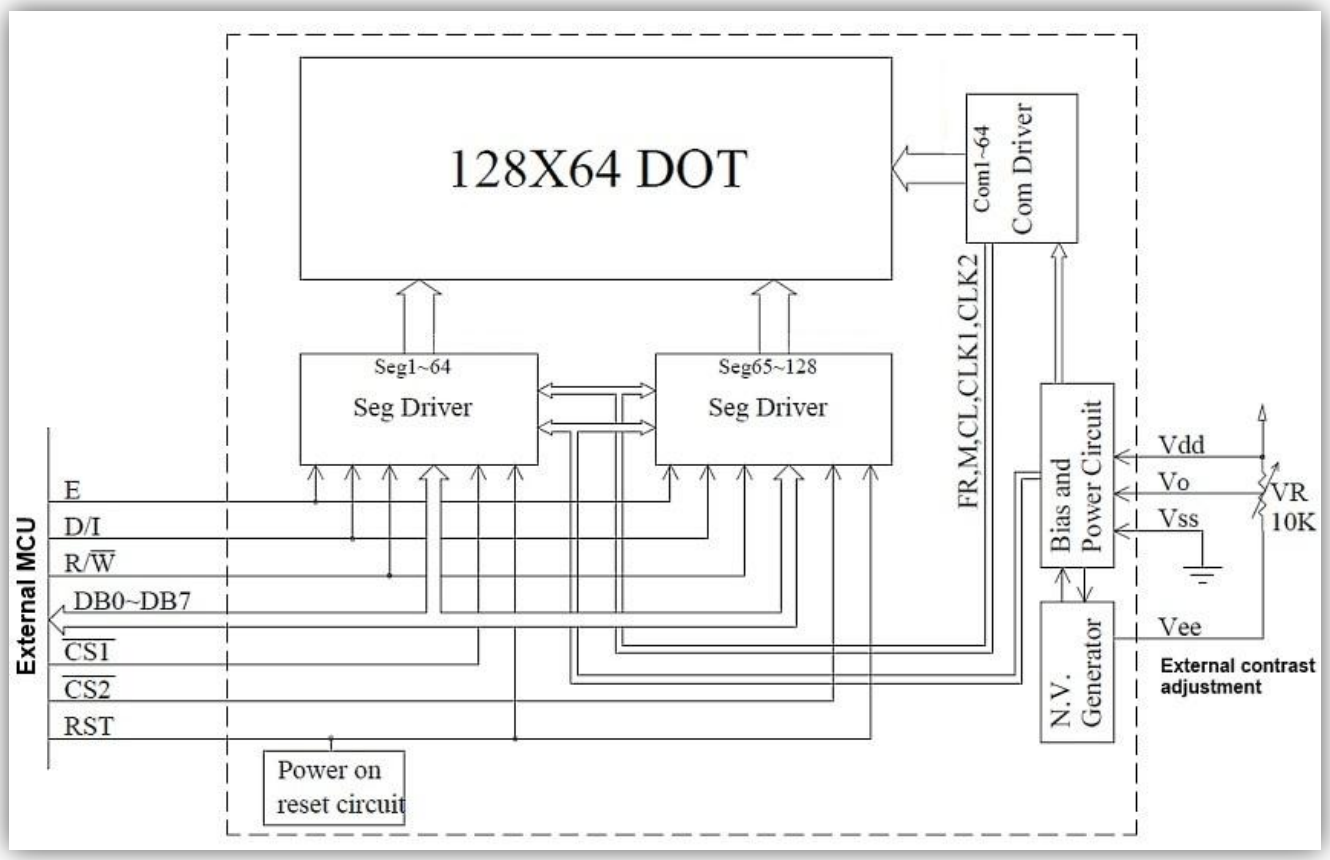
نلاحظ من مخطط البنية السابقة وجود ما يلي :

- معالج صغري (MPU) لقيادة عملية الإظهار .
- دارتين متكاملتين من النوع (KS0108B) لقيادة الأعمدة في الشاشة .
- دارة متكاملة واحدة من النوع (KS0107B) لقيادة الأسطر في الشاشة .

تُستخدم الدارة المتكاملة (KS0108B) لقيادة أعمدة الشاشة الرسوميّة حيث أنها دارة من صنع شركة (Samsung) تمتلك (64) خرجاً للقيادة بحيث يتصل كل خرج مع عمود من أعمدة الشاشة وعلى اعتبار أن الشاشة ذات الأبعاد (64×128) فيها (128) عمود وبالتالي نحن بحاجة هنا إلى دارتين متكاملتين لقيادة الأعمدة في الشاشة .

بالمقابل تُستخدم الدارة المتكاملة (KS0107B) لقيادة أسطر الشاشة الرسوميّة وهي أيضاً من صنع شركة (Samsung) تمتلك (64) خرجاً للقيادة بحيث يتصل كل خرج مع سطر من أسطر الشاشة وعلى اعتبار أن الشاشة ذات الأبعاد (64×128) فيها (64) سطر وبالتالي نحن بحاجة هنا إلى دارة متكاملة واحدة فقط لقيادة الأسطر في الشاشة .

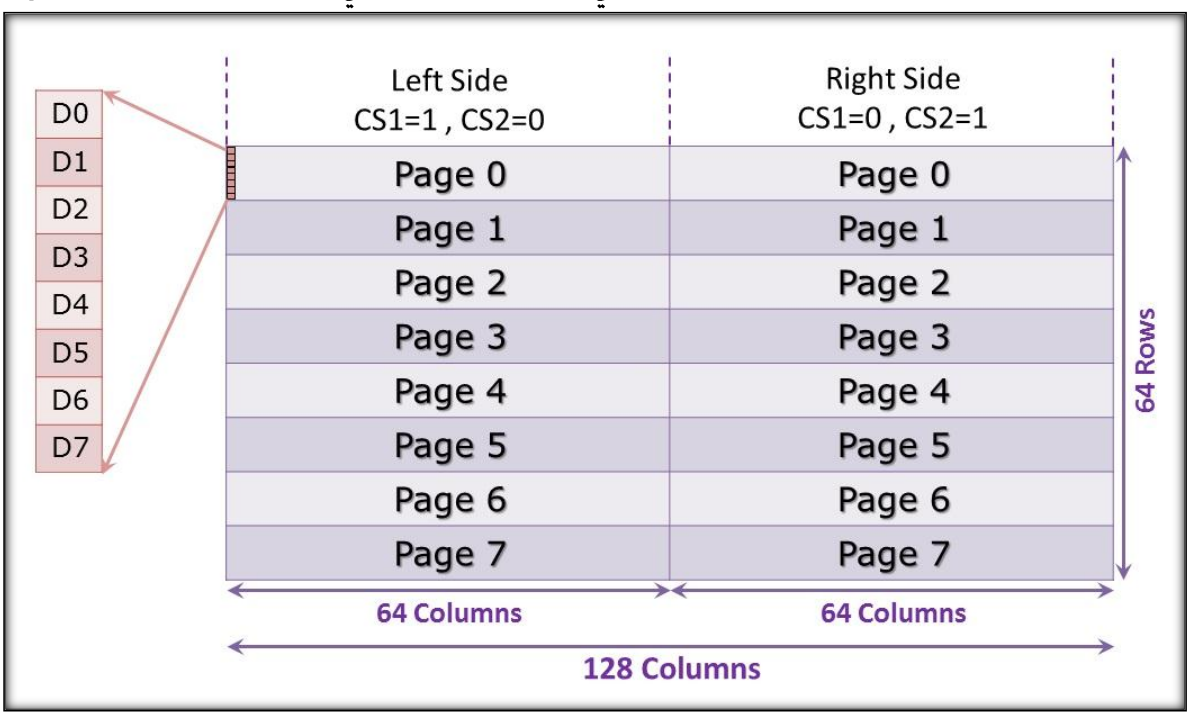
يمكن استخدام دارات قيادة أخرى للأسطر والأعمدة غير الدارات المتكاملة السابقة إلا أنها جميعاً تقوم بنفس الوظيفة وتتصل مع بعضها البعض ومع المتحكم الخارجي وفق المخطط التالي :



يقوم المعالج الصغري في الشاشة بتنسيق عمل الدارات المتكاملة في اللوحة المطبوعة كما يتحكم بعملية العرض حيث أنه يستقبل البيانات على مر معطيات الشاشة ويُقوم بعرضها على الشاشة في الموضع المُحدد.

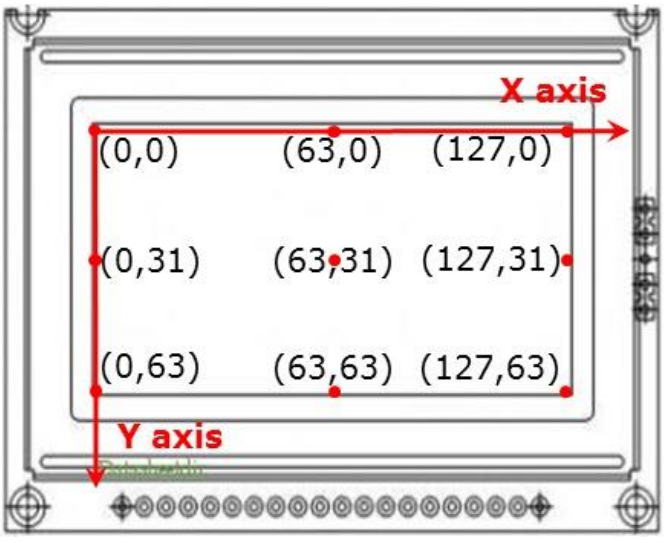
تقسيمات الشاشة في شاشة الإظهار الكريستاليّة الرسوميّة (GLCD) :

على خلاف شاشة الإظهار المحرفيّة فإن شاشة الإظهار الرسومية تتألف من مصفوفة من البكسلات تتوزع على كامل أبعاد الشاشة بشكل أسطر وأعمدة , ولسهولة التعامل معها برمجيّاً تم تقسيم الشاشة إلى أجزاء تتناسب مع أبعادها وتقسيم الجزء الواحد إلى عدة صفحات (Pages) من البكسلات , وكمثال عن ذلك لنأخذ الشاشة الرسومية ذات الأبعاد (64×128) التي يُبيّن الشكل التالي تقسيمات الشاشة فيها :



نلاحظ من الشكل السابق أن الشاشة مُؤلّفة من قسمين رئيسيين : الجانب اليميني (Right Side) والجانب اليساري (Left Side) و تتولى دارة قيادة الأسطر قيادة أسطر الشاشة كاملةً أما الأعمدة فتتولى دارتين قيادتهما كل دارة مسؤولة عن قيادة قسم .

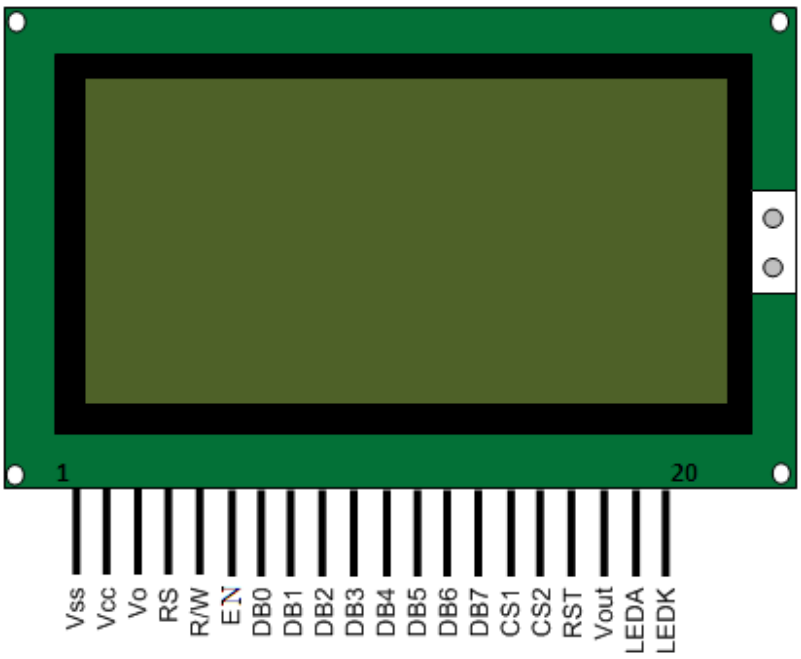
يُقسم كل جانب من الشاشة إلى ثماني صفحات (Page0 → Page7) تحتوي الصفحة الواحدة على (64) بايت وبالتالي (64×8=512 bits) أو بمعنى آخر (512) بكسل حيث أن كل بت يُمثل بكسل واحد يظهر هذا البكسل عندما تكون قيمة البت الخاص به (1) منطقي ويختفي عندما تكون قيمته (0) منطقي



يُفيد تقسيم الشاشة إلى صفحات في عملية إظهار النصوص عليها حيث أننا نتعامل معها حينها كصفحات , ويمكن للسطر الواحد أن يأخذ صفحة واحدة أو اثنتين أو أكثر بحسب حجم الخط المكتوب به . بينما يتم التعامل مع الشاشة كمصفوفة من البكسلات عندما نُريد إظهار الرسوم عليها وبالتالي فإننا نتعامل مع البكسلات حينها بحسب إحداثيات كل بكسل . ويُبيّن الشكل المجاور بعض إحداثيات النقاط في شاشة GLCD (64×128) .

أقطاب شاشة الإظهار الكريستاليّة الرسوميّة (GLCD) :

بشكل عام تحتوي أي شاشة إظهار رسوميّة على (20) قطب تتوضّع في الغالب على صف واحد فوق شاشة (GLCD) أو تحتها كما هو مُبيّن في الشكل التالي :



ويُبيّن الجدول التالي أسماء الأقطاب مع وظيفة كل قطب في شاشة (GLCD) :

Pin no.	Name	Function
1	Vss	Ground (0 V)
2	Vcc	Supply voltage (5V)
3	Vo	Contrast adjustment
4	Register Select (RS)	High for display data - Low for instruction code
5	Read/Write (R/W)	Low for write to the register High for read from the register
6	Enable (EN)	Reads data when high Writes data at high to low transition (falling edge)
7	DB0	Data Bus Pins
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	CS1	Chip selection for IC1 (columns 0 → 63)
16	CS2	Chip selection for IC2 (columns 64 → 127)
17	RST	Reset signal Pin
18	Vout	Output voltage for LCD driving
19	LED A	Backlight V_{cc} (5V)
20	LED K	Backlight Ground (0V)

شرح الأقطاب :

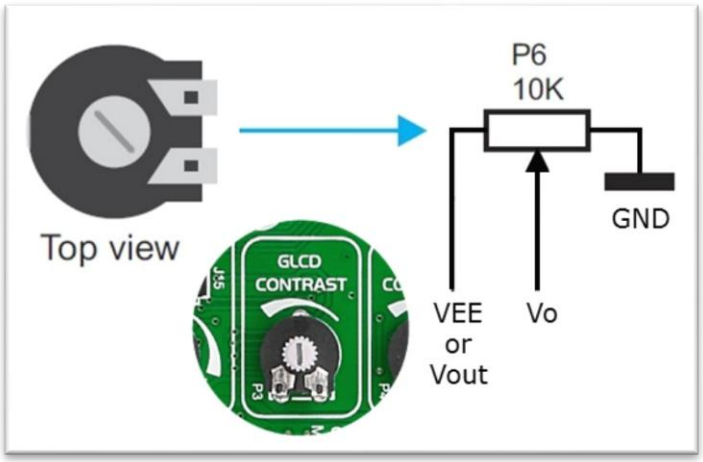
- *القطب رقم (1) (VSS) :*

قطب أرضي الشاشة ويتصل بالقطب الأرضي للدارة (GND) .

- *القطب رقم (2) (VDD) :*

قطب التغذية الموجبة للشاشة ويتصل بقطب تغذية الدارة (+5V) .

- *القطب رقم (3) (VO) :*



يشترك هذا القطب مع القطب رقم (18) من الشاشة (VEE) في ضبط جهد التباين (Contrast) للشاشة وذلك وفقاً للمخطط المُبيّن جانباً .

ويقصد بالتباين حدة ظهور البكسل على الشاشة حيث أنّ أقل تباين أن لا نرى شيئاً على الشاشة وأعلى تباين للشاشة أن تظهر البكسلات بأعلى حدة ظهور لها .

- *القطب رقم (4) (RS) :*

قطب اختيار مسجل الدخل وهو قطب تحكم يتم عبره تحديد نوع البيانات المُرسلة عبر مر معطيات الشاشة وذلك وفق ما يلي :

عند تطبيق *(0) منطقي* على هذا القطب يتم اعتبار البيانات المُستقبلة عبر مر معطيات الشاشة على أنها *بايتات تحكم (Control Bytes)* .

عند تطبيق *(1) منطقي* على هذا القطب يتم اعتبار البيانات المُستقبلة عبر مر معطيات الشاشة على أنها *بايتات معطيات (Data Bytes)* .

- *القطب رقم (5) (R/W) :*

قطب تحديد القراءة / الكتابة من / على ذاكرة الشاشة حيث أنه :

عند تطبيق *(0) منطقي* على هذا القطب فهذا يعني أننا نريد *الكتابة* على ذاكرة الشاشة .

عند تطبيق *(1) منطقي* على هذا القطب فهذا يعني أننا نريد *القراءة* من ذاكرة الشاشة .

بشكل عام نقوم بوصل هذا القطب إلى القطب الأرضي (GND) بشكل دائم .

- *القطب رقم (6) (E) :*

قطب نبضة التمكين وهو قطب تحكم هام جداً في تنظيم عمل شاشة (LCD) , إذ أن أي عملية كتابة على ذاكرة الشاشة تحتاج بعدها إلى نبضة تمكين عند الجبهة الهابطة على هذا القطب لتأكيدها (نرفع هذا القطب إلى (1) منطقي ومن ثم نعيده إلى (0) بعد تأخير زمني مناسب) .

- *الأقطاب ذوو الأرقام (من (7) إلى (14)) (DB0 → DB7) :*

وهي أقطاب مر معطيات شاشة (GLCD) وعددها (8) أقطاب تُستخدم جميعها كممر معطيات لنقل معطيات العرض وبايتات التحكم إلى الشاشة .

- *القطب رقم (15) (CS1) :*

قطب اختيار الشريحة الأولى من شريحتي قيادة الأعمدة في الشاشة ويُستخدم لتفعيل الإظهار في الجانب الأيسر من الشاشة (Columns from 0 to 63) .

- *القطب رقم (16) (CS2) :*

قطب اختيار الشريحة الثانية من شريحتي قيادة الأعمدة في الشاشة ويُستخدم لتفعيل الإظهار في الجانب الأيمن من الشاشة (Columns from 64 to 127) .

- *القطب رقم (17) (RST) :*

قطب تصفير شاشة الإظهار وهو قطب فعّال عند (0) منطقي , حيث أنه لدى تصفير شاشة الإظهار يتم مسح العرض فيها بالكامل وتصفير مسجلاتها وإعادة مؤشر الرسم إلى البكسل الأول فيها ذو الإحداثيات (0,0) .

- *القطب رقم (18) (Vout) :*

ويُسمّى أيضاً (VEE) ويشترك هذا القطب مع القطب رقم (3) من الشاشة (Vo) في ضبط جهد التباين (Contrast) للشاشة .

- *القطب رقم (19) (LED A) :*

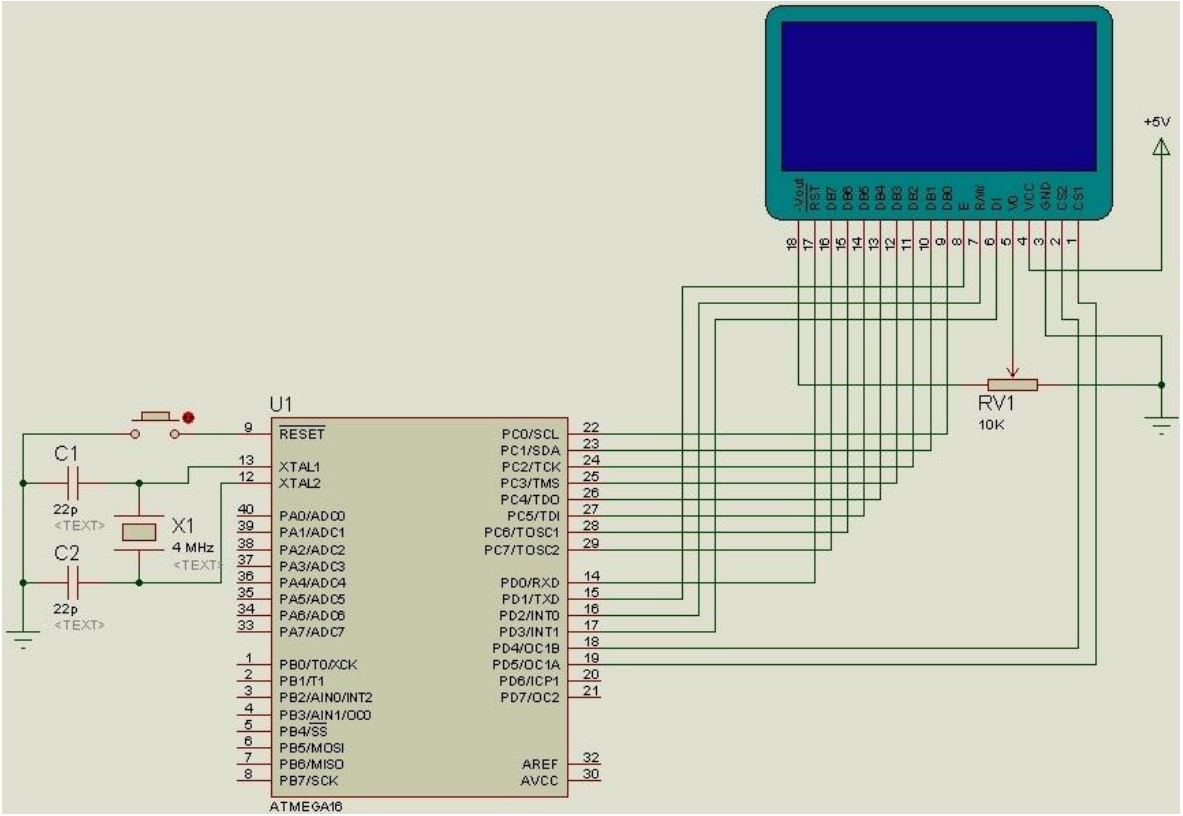
قطب مصعد الليد المُسطّح الخاص بالإضاءة الخلفية (Back-Light) للشاشة .

- *القطب رقم (20) (LED K) :*

قطب مهبط الليد المُسطّح الخاص بالإضاءة الخلفية (Back-Light) للشاشة .

مخطط وصل شاشة الإظهار الرسوميّة (GLCD) مع المتحكم الصغري :

يتم وصل شاشة الإظهار الكريستاليّة الرسوميّة (GLCD) مع المتحكم الصغري وفق المخطط التالي :



تعليمات قيادة شاشة الإظهار الرسوميّة (GLCD) في مترجم (BASCOM AVR) :

زوّدت شركة (MCS Electronics) مترجمها (BASCOM AVR) بتوابع وإجراءات جاهزة لقيادة شاشة الإظهار الرسوميّة (GLCD) وكما في شاشة الإظهار المحرفيّة (LCD) تُقسم تعليمات القيادة للشاشة الرسوميّة إلى قسمين رئيسيين :

- تعليمات التهيئة (Configuration Instructions) .
- تعليمات العرض (Display Instructions) .

أولاً : تعليمات التهيئة :

تضمين تعليمات مكتبة شاشة الإظهار الرسوميّة (GLCD)	`$lib "glcdKS108.lib"`
تضمين ملف الخطوط (Font5x8.font) تمهيداً لاستخدامه في الكتابة على شاشة الإظهار الرسوميّة (GLCD)	`$include "Font5x8.font"`
تضمين ملف الصورة الخارجية (Face1.bgf) تمهيداً لعرض تلك الصورة على شاشة الإظهار الرسوميّة (GLCD)	`Lable1:` `$bgf "Face1.bgf"`
تحديد أقطاب المتحكم الموصولة مع شاشة الإظهار الرسوميّة (GLCD)	`Config Graphlcd = 128 * 64sed , Dataport = Portc , Controlport = Portd , Ce = 5 , Ce2 = 4 , Cd = 3 , Rd = 2 , Reset = 0 , Enable = 1`

ثانياً : تعليمات العرض :

مسح كامل العرض على الشاشة من نصوص ورسومات	`Cls`
مسح النصوص المعروضة على الشاشة فقط دون الرسومات	`Cls Text`
مسح الرسوم المعروضة على الشاشة فقط دون النصوص	`Cls Graph`
طباعة النص الواقع بين الإشارتين ""على الشاشة ابتداءاً من السطر (Y) والبكسل (X) وإما أن تكون الطباعة طباعة عادية (inv=0) أو طباعة معكوسة (inv=1)	`Lcdat Y , X , "Hello" [,inv]`
طباعة قيمة المتحول (var) على الشاشة ابتداءاً من السطر (Y) والبكسل (X) وإما أن تكون الطباعة طباعة عادية (inv=0) أو طباعة معكوسة (inv=1)	`Lcdat Y , X , var [,inv]`
رسم خط على الشاشة يمتد من نقطة البداية (X1,Y1) إلى نقطة النهاية (X2,Y2) مع إعطاء لوناً له	`Line(X1 , Y1)-(X2 , Y2) , Color`
التحكم بإخفاء (Value=0) أو إظهار (Value=1) النقطة ذات الإحداثيات (X,Y) على الشاشة	`Pset X , Y , Value`
رسم دائرة على الشاشة يقع مركزها في النقطة (X,Y) وبنصف قطر مقداره (r) مع شيفرة اللون في المتحول (Color)	`Circle(X , Y) , r , Color`

إظهار الصورة الموجودة عند اللافتة (Lable1) ابتداءاً من النقطة (X,Y) على الشاشة	`Showpic X , Y , Lable1`
تحديد الملف المُستخدم في كتابة الخطوط على الشاشة وبالتالي فإن أي نص يُكتب عليها سيُكتب بتنسيق ملف الخطوط المُحدد (Font8x8)	`Setfont Font8x8`

البرنامج :

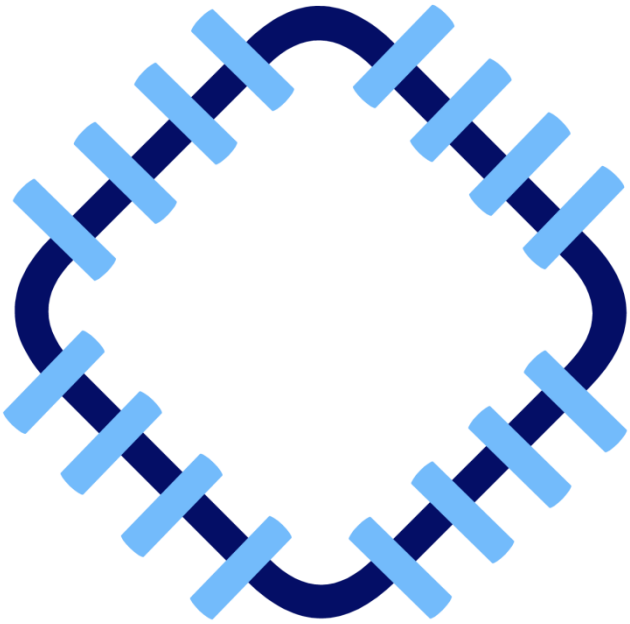
نقوم في هذا البرنامج باستعراض تعليمات التعامل مع شاشة الإظهار الرسوميّة (GLCD) , حيث نقوم أولاً بكتابة نصوص بأحجام مختلفة ومن ثم رسم خطوط من نقطة بداية (X1,Y1) إلى نقطة نهاية (X2,Y2) ومن ثم نقوم برسم بكسلات بترتيب معين ثم إخفاؤها وبعدها نرسم دوائر متحدة المركز (X,Y) بأنصاف أقطار مختلفة ومن ثم نقوم أخيراً بعرض مجموعة صور خاصة بالشاشة الرسومية (128×64) GLCD .

```
$regfile = "m16def.dat"
$crystal = 4000000
$lib "glcdKS108.lib"
'-----------------------------------------------------

Config Graphlcd = 128 * 64sed , Dataport = Portc , Controlport = Portd ,
Ce = 4 , Ce2 = 5 , Cd = 3 , Rd = 2 , Reset = 0 , Enable = 1

'-----------------------------------------------------

Dim X As Byte
Dim Y As Byte

Do

Cls
Cls Text
Cls Graph

'8 x 8 Font >> If GLCD = 64 x 128 Then >> 8-Row(Y) x 16-Col(X)

Setfont Font8x8
   Lcdat 3 , 11 , "Font 5x8 Test" , 1
   Lcdat 4 , 11 , "Font 6x8 Test" , 1
   Lcdat 5 , 11 , "Font 8x8 Test" , 1
   Wait 2
   Cls

   Setfont Font5x8
      Lcdat 1 , 1 , "LINE1" : Wait 1
      Lcdat 2 , 1 , "LINE2" : Wait 1
      Lcdat 3 , 1 , "LINE3" : Wait 1
```

```
   Setfont Font6x8
      Lcdat 4 , 1 , "LINE4" : Wait 1
      Lcdat 5 , 1 , "LINE5" : Wait 1
      Lcdat 6 , 1 , "LINE6" : Wait 1

   Setfont Font8x8
      Lcdat 7 , 1 , "LINE7" : Wait 1
      Lcdat 8 , 1 , "LINE8" : Wait 1
'*****************************************************
Cls
'16 x 16 Font >> If GLCD = 64 x 128 Then >> 4-Row(Y) x 8-Col(X)
Setfont Font16x16
   Lcdat 4 , 23 , "16x16" :
   Wait 2
   Cls
      Lcdat 1 , 1 , "LINE1" : Wait 1
      Lcdat 3 , 1 , "LINE2" : Wait 1
      Lcdat 5 , 1 , "LINE3" : Wait 1
      Lcdat 7 , 1 , "LINE4" : Wait 1
   Cls
'*****************************************************
Setfont Font8x8
   Lcdat 4 , 11 , "Drawing Lines" : Wait 4
   Cls Text
      Line(0 , 0) -(127 , 0) , 255
      Wait 1
      Line(0 , 63) -(127 , 63) , 255
      Wait 1
      Line(0 , 0) -(0 , 63) , 255
      Wait 1
      Line(127 , 0) -(127 , 63) , 255
      Wait 1
      Line(0 , 0) -(127 , 63) , 255
      Wait 1
      Line(0 , 63) -(127 , 0) , 255
      Wait 1
   Cls Graph
'*****************************************************
   Lcdat 4 , 11 , "SET/RST Pixel" : Wait 4
   Cls Text

      For X = 0 To 127
         Pset X , 20 , 255
         Pset X , 43 , 255
         Waitms 50
      Next X


      For Y = 0 To 63
         Pset 42 , Y , 255
         Pset 86 , Y , 255
         Waitms 50
```

```
      Next Y


      For X = 127 To 0 Step -1
         Pset X , 20 , 0
         Pset X , 43 , 0
         Waitms 50
      Next X

      For Y = 63 To 0 Step -1
         Pset 42 , Y , 0
         Pset 86 , Y , 0
         Waitms 50
      Next X
   Cls Graph
'*******************************************************
   Lcdat 4 , 11 , "Drawing Circle" : Wait 4
   Cls : Cls Text : Cls Graph
   For X = 1 To 31
      Circle(63 , 31) , X , 255
      Waitms 100
      Circle(63 , 31) , X , 0
   Next X
   Cls Graph
'*******************************************************
   Showpic 0 , 0 , Smiley1
   Wait 2
   Cls Graph

   Showpic 0 , 0 , Smiley2
   Wait 2
   Cls Graph

   Showpic 0 , 0 , Smiley3
   Wait 2
   Cls Graph

Loop
End
'*******************************************************
$include "Font5x8.font"
$include "Font6x8.font"
$include "Font8x8.font"
$include "Font16x16.font"
'*******************************************************

Smiley1:
$bgf "Smiley1.bgf"

Smiley2:
$bgf "Smiley2.bgf"
```

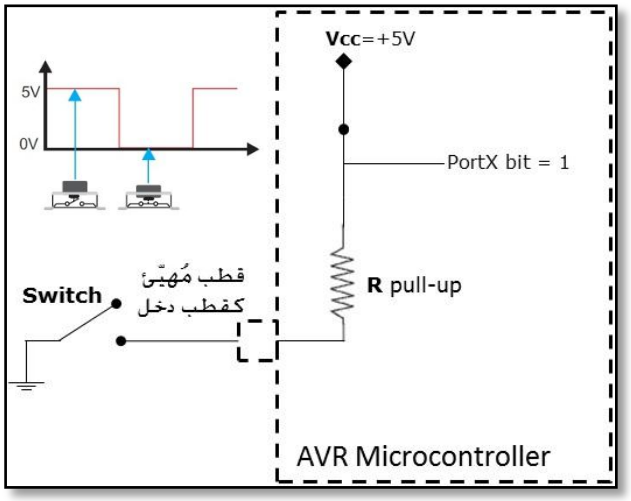
```
Smiley3:
$bgf "Smiley3.bgf"
'*****************************************************
```



الجلسة السادسة

وصل مصفوفة أزرار لحظيّة (Keypad) ولوحة مفاتيح حاسب (Keyboard) مع المتحكم الصغري

أولاً : مصفوفة الأزرار اللحظيّة (Keypad)



استعرضنا في إحدى الدروس السابقة كيفيّة وصل أزرار لحظية مع المتحكم الصغري , حيث أننا نقوم بوصل أحد أطراف الزر اللحظي مع قطب دخل من المتحكم والطرف الآخر منه مع الأرضي (GND) , وبوصل مقاومة الرفع الداخليّة لهذا القطب يتوضع عليه (1) منطقي بشكل افتراضي إلى أن يتم ضغط الزر الموصول به عندها سيتم وصل قطب الدخل إلى الأرضي (GND) وبالتالي تتغير حالته المنطقية إلى الصفر ليتم إعلام المتحكم بضغط هذا الزر اللحظي .

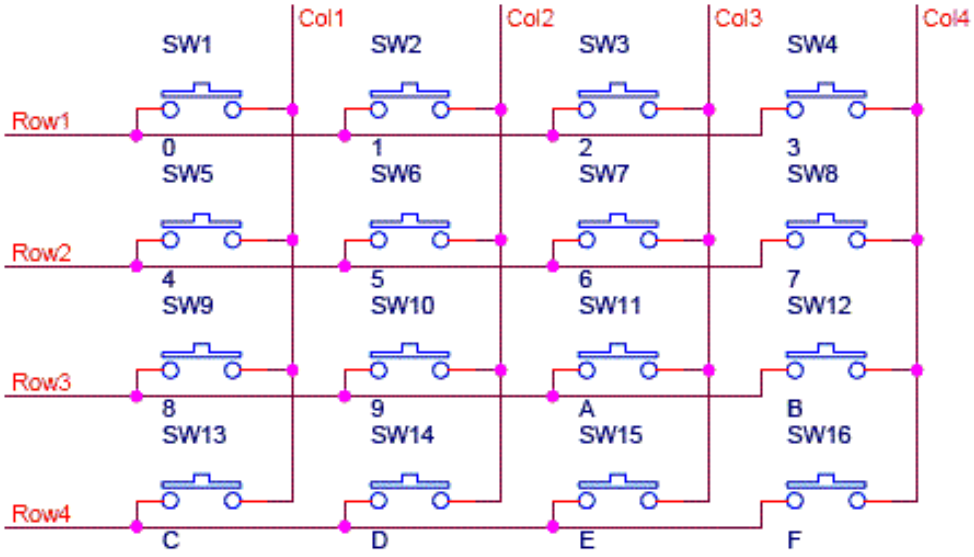
إن ما سبق من الكلام يصلح في حال وصلنا زر لحظي واحد أو اثنين أو ثلاثة أو حتى ثمانية أزرار مع المتحكم الصغري أما في حال أردنا وصل أكثر من ذلك من الأزرار اللحظية فعندها علينا البحث عن طريقة أخرى للوصل لأن وصلها بالطريقة السابقة يستهلك عدد كبير من أقطاب المتحكم الصغري .

نتّبع في حالة أردنا وصل عدة أزرار لحظية مع المتحكم (تسعة أزرار فما فوق) طريقة الوصل المصفوفيّة حيث أننا نقوم بوصل هذه الأزرار مع بعضها البعض على شكل مصفوفة مؤلفة من عدد من الأسطر والأعمدة وبوصلها مع أقطاب المتحكم نعتمد طريقة المسح السريع من خلال البرنامج في تحديد الزر المضغوط منها في حال تم ضغط أحد هذه الأزرار .

تتوفر في الأسواق مجموعة واسعة من مصفوفة الأزرار اللحظيّة (Matrix Keypad) وبأشكال عديدة وأحجام مختلفة تُبيّن الصور التالية بعضاً منها :

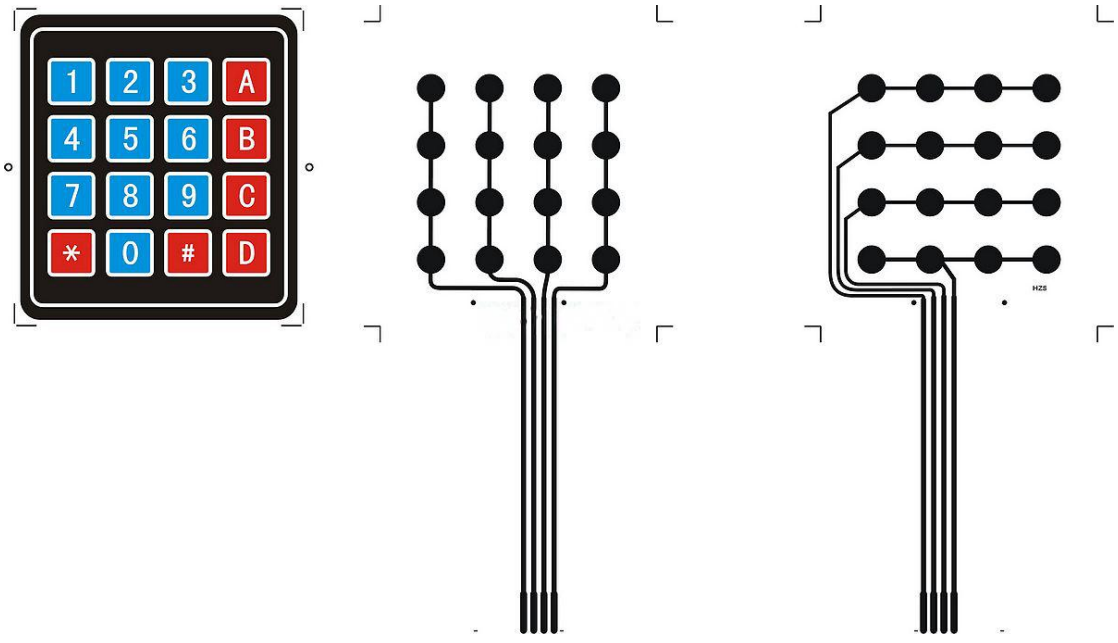
مخطط الوصل الداخلي لمصفوفة الأزرار اللحظية (Keypad) :

تعتمد فكرة الوصل الداخلي لهذه المصفوفة على وصل أقطاب الأزرار الموجودة في السطر الواحد مع بعضها البعض ومن ثم أقطاب الأزرار الموجودة في العمود الواحد مع بعضها البعض ليتشكّل لدينا مخطط كما في الشكل التالي :



نلاحظ في المخطط السابق ونتيجةً لهذا الوصل تشكّل لدينا مصفوفة مؤلفة من أربعة أسطر وأربعة أعمدة من الأزرار اللحظية , وهي طريقة وصل عالميّة مُتّبعة في معظم لوحات المفاتيح اللحظيّة في العالم ابتداءاً من لوحة مفاتيح الهاتف وانتهاءاً بلوحة مفاتيح الحاسب . أما عن كيفيّة تحديد الزر المضغوط من هذه المصفوفة فهذا يقع على عاتق المُبرمج حيث أنه يتولى كتابة تعليمات برنامج مسح المصفوفة الذي يقوم بهذه المهمة .

يُمكن تشكيل مصفوفة أزرار لحظيّة (Keypad) انطلاقاً من مجموعة من الأزرار حيث يقوم المصمّم بتأمين عملية وصل المصفوفة على الدارة المطبوعة بشكل مطابق لمخطط الوصل السابق , إلا أنه ولتسهيل العمل تتوفر في الأسواق مصفوفات جاهزة من الأزرار اللحظية متّصلة داخليّاً وبقياسات معيارية مختلفة .

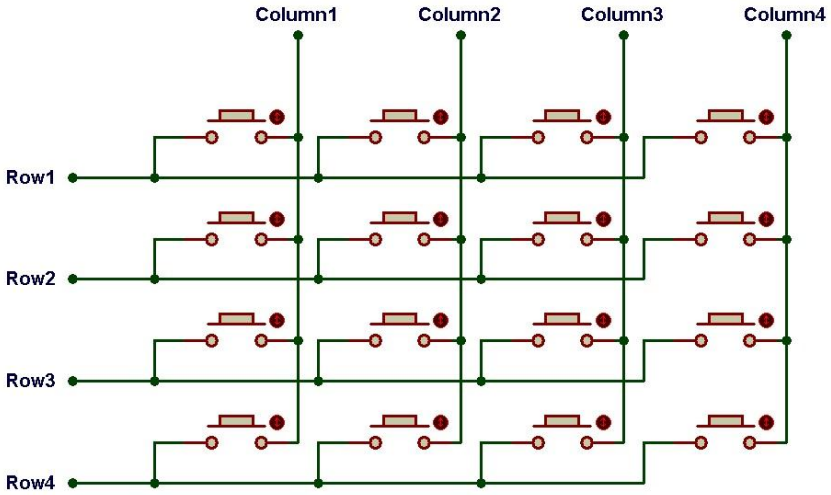


مبدأ عملية المسح السريع في قيادة مصفوفة الأزرار اللحظية (Keypad) :

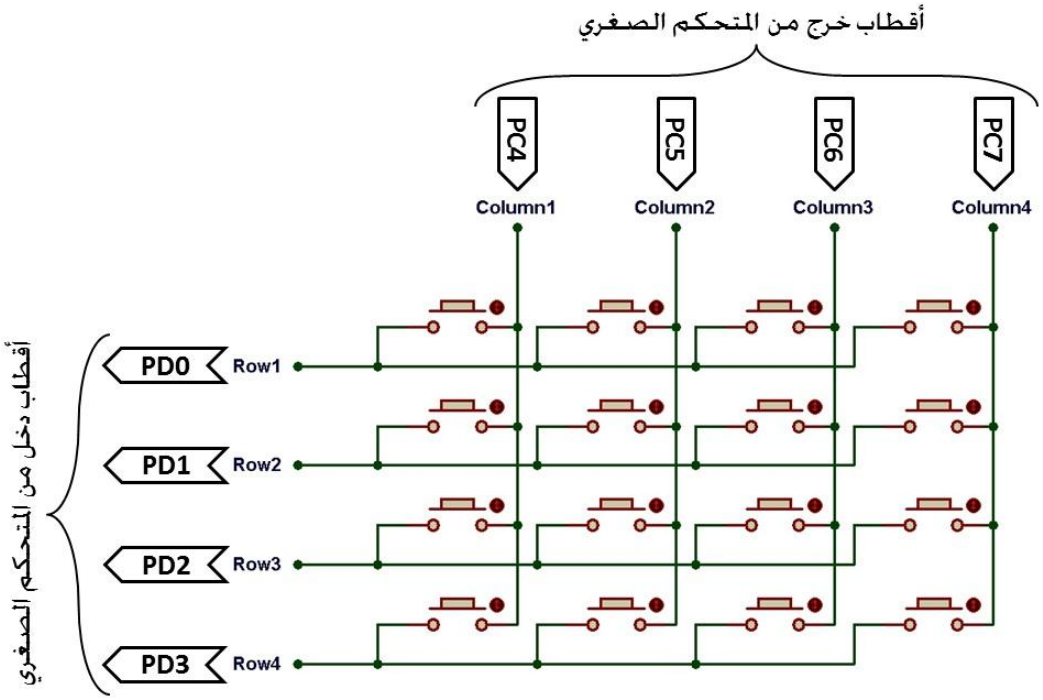
لاحظنا في مخطط الوصل السابق كيف أن وصل الأزرار ضمن مصفوفة أدى إلى اختصار كبير في عدد الأقطاب اللازمة لقيادة هذه الأزرار , حيث أننا نحتاج مثلاً إلى ثمانية أقطاب فقط لقيادة مصفوفة (4×4) فيها (16) زراً لحظيّاً , وسنستعرض في هذه الفقرة الطريقة التي يتم فيها تمييز الزر المضغوط من هذه المصفوفة . ولابد من الإشارة هنا إلى وجود طرق عديدة لهذا الأمر إلا أنها جميعها تستخدم نفس المبدأ المنطقي في عملية المسح .

خطوات عملية المسح السريع لتحديد الزر المضغوط من المصفوفة :

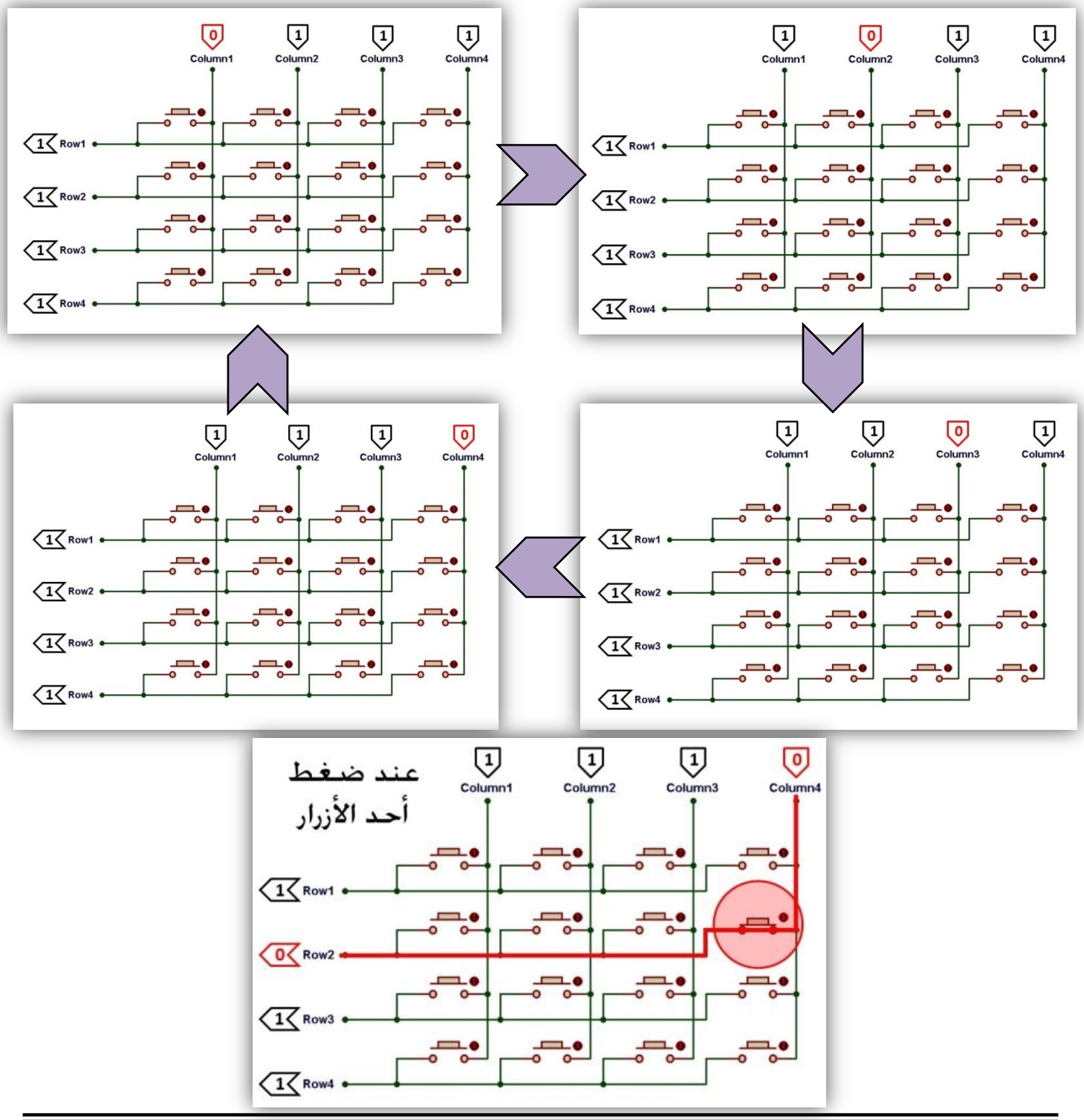
- نقوم بتأمين وصل الأزرار اللحظية كمصفوفة يخرج منها مجموعة من الأسطر والأعمدة (ليس شرطاً أن يكونا متساويين بالعدد) , وإذا كنّا نستخدم مصفوفة أزرار جاهزة نقوم بتحديد أقطاب الأسطر والأعمدة منها , فيكون لدينا المخطط التالي :



- نقوم بوصل أسطر المصفوفة إلى أقطاب **دخل** من المتحكم الصغري *(مع وصل مقاومة الرفع الداخليّة لكل قطب)* وأعمدة المصفوفة إلى أقطاب **خرج** من المتحكم الصغري كما في المخطط التالي :

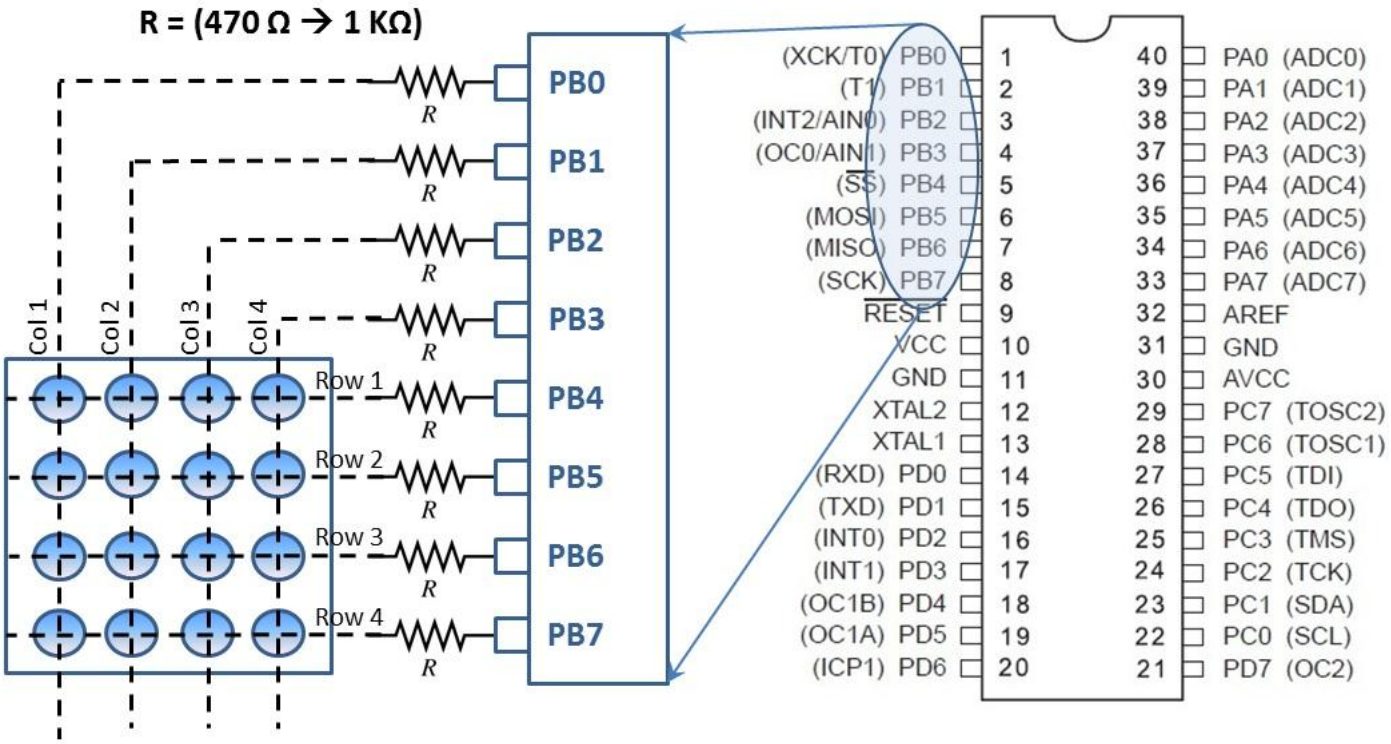


- على اعتبار أن أقطاب دخل المتحكم الموصولة مع أسطر المصفوفة مرفوعة إلى (1) منطقي (كل قطب متصل بمقاومة الرفع الداخلية الخاصة به) نقوم الآن ومن خلال تعليمات البرنامج بإدخال المتحكم ضمن حلقة خاصة لمسح أزرار المصفوفة , حيث يقوم فيها بتخريج القيمة (1) منطقي على جميع أعمدة المصفوفة عدا عموداً واحداً يُخرّج عليه القيمة (0) منطقي بحيث يمر على جميع أعمدة المصفوفة بالتناوب وفي كل مرة يقوم بقراءة أقطاب الدخل المتصلة مع أسطر المصفوفة , وعندما نحصل على (0) منطقي على أحد أسطر المصفوفة فيمكن عندها عن طريق القيمة المُخرّجة على الأعمدة والقيمة المُحصّلة من الأسطر تحديد أي زر تم ضغطه من مصفوفة الأزرار اللحظيّة .



تعليمات التعامل مع مصفوفة الأزرار اللحظية في لغة (BASCOM AVR) :

لحسن الحظ تُقدّم لنا لغة (BASCOM AVR) تعليمات جاهزة لتأمين عملية مسح مصفوفة الأزرار اللحظية حيث أنها تُوفّر على المُبرمج كتابة تعليمات المسح بنفسه وتختصر حلقة المسح كاملةً بتعليمة واحدة فقط . إلا أنها تشترط عليك في البداية وصل مصفوفة الأزرار اللحظيّة (Keypad 4×4) مع نافذة أقطاب كاملة من المتحكم الصغري وذلك وفق مخطط ثابت حصراً حيث يتم وصل القطب الأول من النافذة مع العمود الأول من المصفوفة والقطب الثاني منها مع العمود الثاني ..... الخ , ومن ناحية الأسطر يتم وصل القطب الخامس من النافذة مع السطر الأول والقطب السادس مع السطر الثاني ..... الخ , كما هو مُبيّن في المخطط التالي :



بعد تأمين وصل المخطط السابق يمكن استخدام تعليمات لغة البرمجة (BASCOM AVR) المُخصّصة لقيادة مصفوفة الأزرار اللحظيّة . وكالعادة تُقسم هذه التعليمات إلى تعليمات تهيئة وتعليمات عمل :

أولاً : تعليمات التهيئة :

تحديد نافذة أقطاب المتحكم الموصولة مع مصفوفة الأزرار اللحظيّة وفق المخطط السابق , حيث أن :
البارامتر (Debounce) يُستخدم لاختيار تأخير زمني معين (mS) للتخلص من العطالة الميكانيكيّة للزر اللحظي حيث يكون هذا الزمن في الحالة الافتراضية (25 mS) .
البارامتر (Delay) يُحدد سرعة مسح المصفوفة (زمن الانتقال في المسح من عمودٍ ما إلى العمود الذي يليه) وتكون قيمته أيضاً بالميلي ثانية (mS) .
	`Config Kbd = Portb ,`
	`Debounce = 100 ,`
	`Delay = 100`

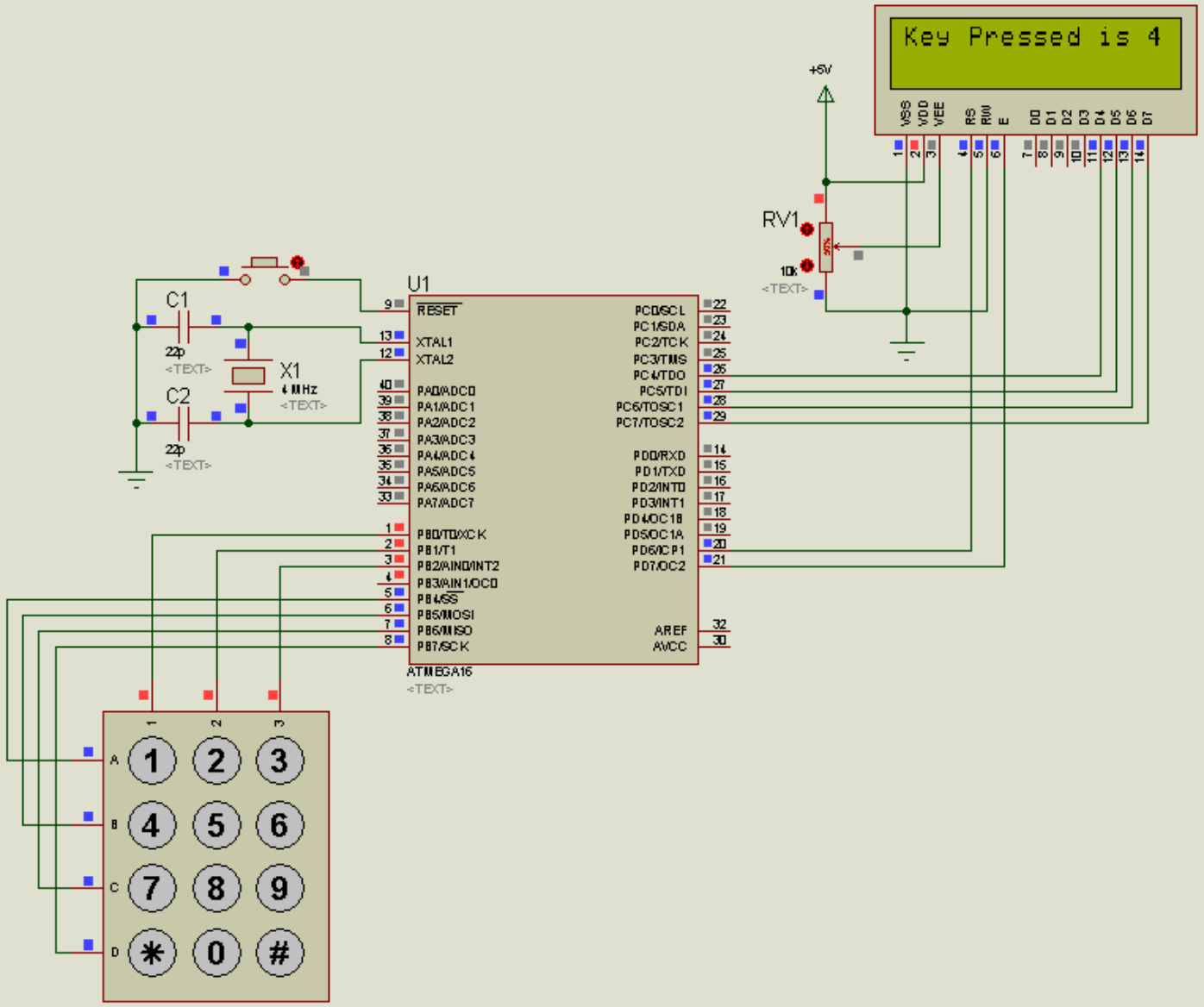
ثانياً : تعليمات العمل :

تقوم هذه التعليمة بقراءة المفتاح المضغوط من مصفوفة أزرار لحظية (Keypad 4×4) موصولة مع المتحكم الصغري وفق المخطط السابق . حيث أنها تُعيد قيمة من نوع بايت تُخزّن في المتحول (Var) ويُمكن تحديد الزر المضغوط من المصفوفة من خلال قيمة المتحول (Var) وذلك وفق الجدول التالي :

```
Var = Getkbd()
```

قيمة المتحول (Var)	الزر المضغوط
0	1
1	2
2	3
3	A
4	4
5	5
6	6
7	B
8	7
9	8
10	9
11	C
12	*
13	0
14	#
15	D
16	لم يتم ضغط أي زر من أزرار المصفوفة

الدارة العملية :



البرنامج :

```
$regfile = "m16def.dat"
$crystal = 4000000

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 ,
Db7 = Portc.7 , E = Portd.7 , Rs=Portd.6

Config Kbd = Portb , Debounce = 50 , Delay = 50

Dim X As Byte

Cls
Cursor Off
```

```
Do

Locate 1 , 1

X = Getkbd()

  Select Case X
      Case 0 : Lcd "Key Pressed is 1"
      Case 1 : Lcd "Key Pressed is 2"
      Case 2 : Lcd "Key Pressed is 3"
      Case 3 : Lcd "Key Pressed is A"
      Case 4 : Lcd "Key Pressed is 4"
      Case 5 : Lcd "Key Pressed is 5"
      Case 6 : Lcd "Key Pressed is 6"
      Case 7 : Lcd "Key Pressed is B"
      Case 8 : Lcd "Key Pressed is 7"
      Case 9 : Lcd "Key Pressed is 8"
      Case 10 : Lcd "Key Pressed is 9"
      Case 11 : Lcd "Key Pressed is C"
      Case 12 : Lcd "Key Pressed is *"
      Case 13 : Lcd "Key Pressed is 0"
      Case 14 : Lcd "Key Pressed is #"
      Case 15 : Lcd "Key Pressed is D"
   End Select

Loop
End
```

ثانياً : لوحة مفاتيح الحاسب (Keyboard)

مبدأ لوحة مفاتيح الحاسب (PC Keyboard) :

تتألف لوحة المفاتيح من عدد كبير من الأسطر والأعمدة تتقاطع عند أزرار اللوحة لتكون مُهيَّأة للتحسس لضغط أي زر فيها حيث يؤدي ضغط الزر إلى حدوث تماس بين سطر الزر وعموده . و يعتمد مبدأ العمل في هذه اللوحة على إرسال شيفرات معينة لدى ضغط إحدى أزرار لوحة المفاتيح إلى الحاسب واستقبال شيفرات الأوامر من الحاسب . حيث تحتوي أي لوحة مفاتيح على شريحة (Keyboard BIOS) خاصة بها تقوم بإرسال شيفرات المفاتيح المضغوطة إلى الحاسب واستقبال شيفرات الأوامر منه وتنفيذها. فمثلاً عندما تضغط على الزر (A) تقوم لوحة المفاتيح بإرسال الشيفرة الخاصة بهذا المحرف وهي (1C) إلى الحاسب وتستمر لوحة المفاتيح بإرسال الشيفرة (1C) إلى الحاسب إلى أن يقوم المستخدم برفع يده عن هذا الزر عندها تقوم لوحة المفاتيح بإرسال الشيفرة (F0) كإشارة إلى تحرير المستخدم للزر (A) . وهكذا...

بشكل عام يمتلك كل زر في لوحة المفاتيح شيفرة خاصة به يستطيع الحاسب عن طريقها التعرف على الزر المضغوط من اللوحة , هذه الشيفرة مؤلفة بالغالب من بايت واحد وتمتلك بعض الأزرار شيفرات مؤلفة من بايتين أو أكثر . يمتلك الزر (PAUSE BREAK) أطول شيفرة بين أزرار لوحة المفاتيح والممثلة بالشيفرة التالية : (E1,14,77,E1,F0,14,F0,77) .

شيفرات أوامر لوحة المفاتيح (Keyboard Commands) :

إن الاتصال بين لوحة المفاتيح والحاسب هو اتصال ثنائي الاتجاه أي أنه يمكن للطرفين (لوحة المفاتيح والحاسب) إرسال واستقبال شيفرات الأوامر المختلفة .

أولاً : شيفرات الأوامر التي تُرسل من الحاسب إلى لوحة المفاتيح :

تتضمن هذه الأوامر في الغالب معلومات تهيئة وإعادة تهيئة مؤشرات الحالة في لوحة المفاتيح (مثل ليدات أزرار اللوحة (Num lock , Caps Lock & Scroll Lock LEDs)) وتتضمن هذه الأوامر الشيفرات التالية :

(ED) Command

وهو أمر تهيئة ليدات لوحة المفاتيح إذ أنه عند استقبال هذا البايت من الحاسب تقوم شريحة لوحة المفاتيح بإعادة البايت (FA) إلى الحاسب كبايت تأكيد . ومن ثم يقوم الحاسب بإرسال بايت آخر يتضمن حالة ليدات اللوحة إذ يتحكم البت الأول منه بحالة ليد الزر (Scroll Lock) والبت الثاني بحالة ليد الزر (Num lock) والثالث بحالة ليد الزر (Caps Lock) .

(EE) Command

وهو عبارة عن أمر محاكاة يرسله الحاسب إلى لوحة المفاتيح لتقوم لوحة المفاتيح بإرسال الأمر ذاته (EE) إلى الحاسب كرد على محاكاته .

(F0) Command

يُحدد هذا الأمر طريقة المسح المتبعة في إرسال شيفرات الأحرف إلى الحاسب .

(F3) Command

يُحدد هذا الأمر معدل تكرار الطباعة للحاسب في حال استمرار الضغط على زر معين .

(F4) Command

يقوم هذا الأمر بتفعيل (Enable) لوحة المفاتيح حيث يقوم بتصفير الذواكر المؤقتة فيها (Buffers) .

(F5) Command

يقوم هذا الأمر بحجب لوحة المفاتيح عن العمل .

(FE) Command

أمر طلب إعادة إرسال آخر بايت تم إرساله من لوحة المفاتيح إلى الحاسب .

(FF) Command

أمر تصفير (Reset) لوحة المفاتيح .

ثانياً : الشيفرات التي تُرسل من لوحة المفاتيح إلى الحاسب :

(FA) Byte

وهو بايت تأكيد (Acknowledge) ترسله لوحة المفاتيح إلى الحاسب كإشارة على استلام بايت منه .

(AA) Byte

تُرسل لوحة المفاتيح هذا البايت كإشارة على نجاح اختبار (Power On Self Test) .

(EE) Byte

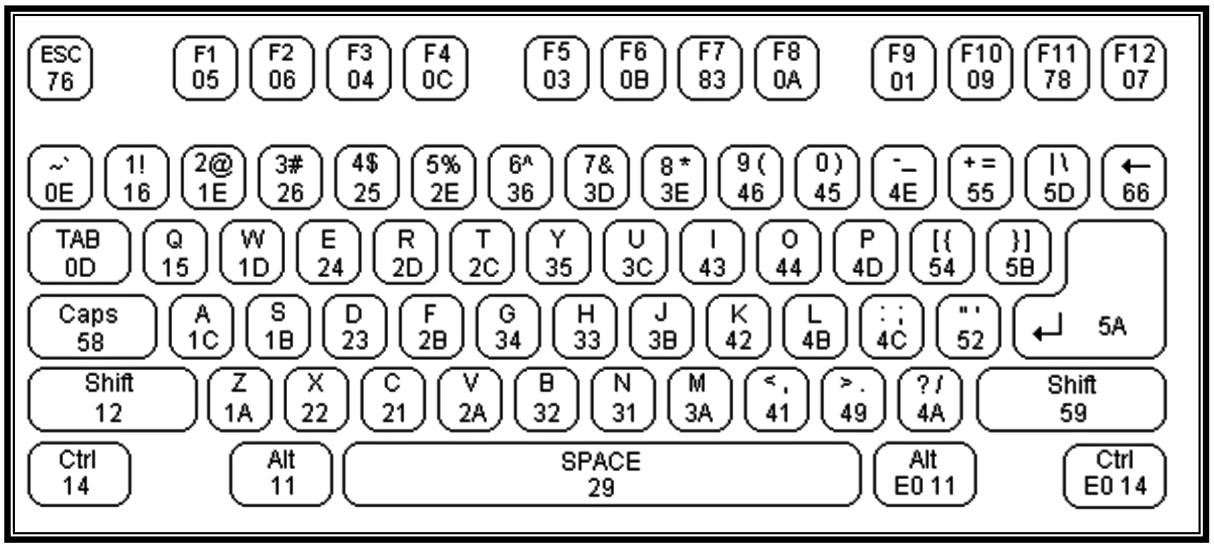
بايت محاكاة لبايت محاكاة آخر مرسل من الحاسب .

(FF) Byte

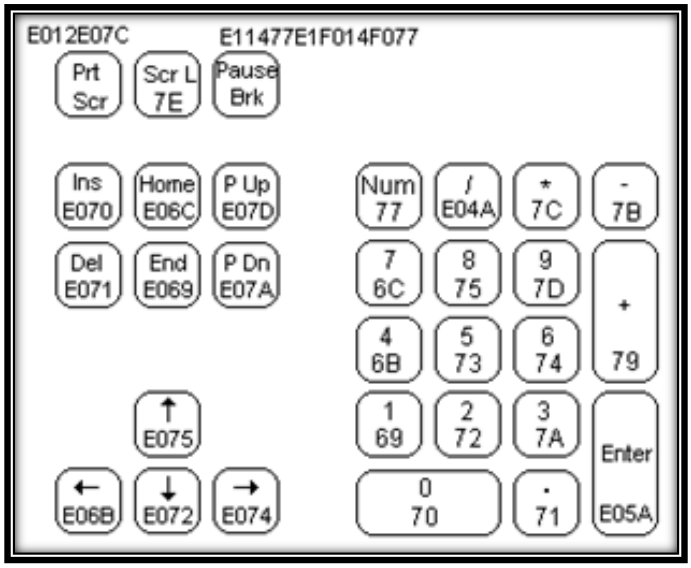
يُشير هذا البايت إلى وجود خطأ أو حالة طفحان في ذواكر اللوحة المؤقتة (Buffers) .

شيفرات مسح أزرار لوحة المفاتيح :

المفاتيح الرئيسية :

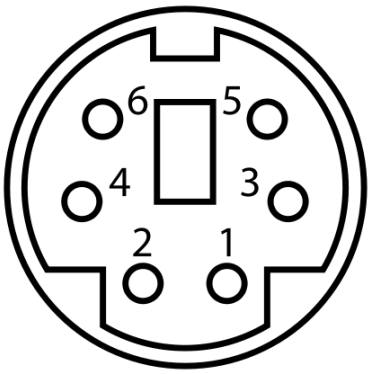


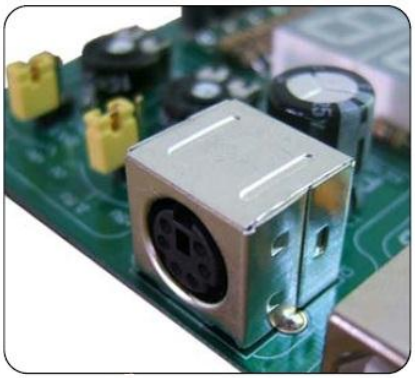
المفاتيح التوسعة :



أقطاب منفذ لوحة المفاتيح (PS/2 Port Pins) :

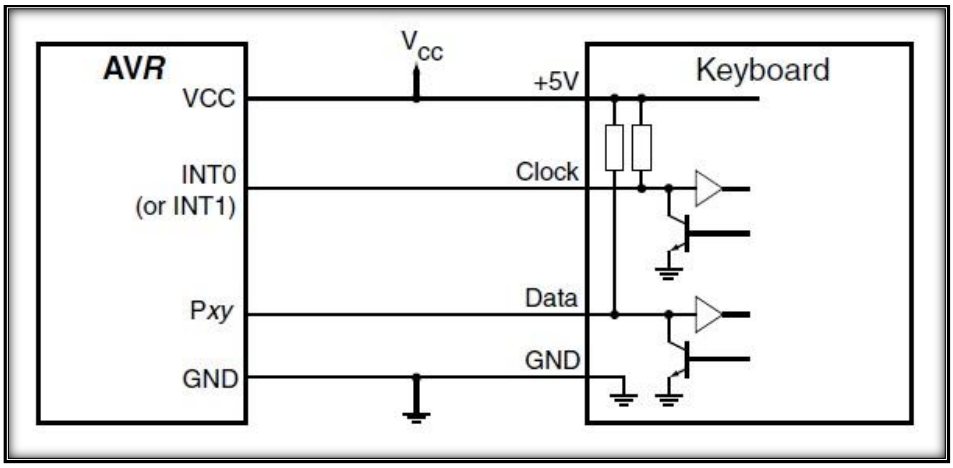
Pins	Function
Pin 1	Data Pin
Pin 2	Not Connected
Pin 3	GND Pin
Pin 4	Vcc Pin
Pin 5	CLK Pin
Pin 6	Not Connected





طريقة وصل لوحة المفاتيح عن طريق منفذ (PS/2) مع المتحكم الصغري :

عند وصل لوحة مفاتيح حاسب (Keyboard) مع متحكم صغري عن طريق منفذ (PS/2) نقوم بوصل قطب تغذية المنفذ (Vcc) والقطب الأرضي فيه (GND) إلى تغذية الدارة والقطب الأرضي فيها . ومن ثم نقوم بوصل قطب المعطيات (Data) من المنفذ مع أحد أقطاب الدخل / الخرج (I/O) من المتحكم أما قطب نبضات الساعة (Clock) الخاص بالمنفذ فإننا نقوم بوصله مع أحد أقطاب المقاطعات الخارجية في المتحكم الصغري (INT0 , INT1 , INT2 , ... etc) حصراً .



أوامر التعامل مع لوحة المفاتيح في لغة (BASCOM AVR) :

زوّدت شركة (MCS Electronics) مترجمها (BASCOM AVR) بتعليمات جاهزة للتعامل المباشر مع منفذ (PS/2) وذلك لوصل لوحة مفاتيح حاسب (Keyboard) أو فأرة حاسب (Mouse) مع المتحكم الصغري .

تحديد أقطاب المتحكم المُتّصلة مع أقطاب منفذ (PS/2) . في هذا المثال يتصل القطب (PD2)(INT0) مع قطب (Clock) من المنفذ والقطب (PD4) مع القطب (Data) منه . بينما تُشير اللافتة (Keydata) إلى توضع جدول شيفرات الأسكي (ASCII) الخاصة بأزرار لوحة المفاتيح ضمن ذاكرة البرنامج .	`Config Keyboard = Pind.2 , Data = Pind.4 , Keydata = Keydata`
تقوم هذه التعليمة بقراءة المفتاح المضغوط من لوحة مفاتيح حاسب (AT Keyboard) موصولة مع المتحكم الصغري عبر منفذ (PS/2) . حيث أنها تُعيد شيفرة الأسكي (ASCII) الخاصة بالزر المضغوط من اللوحة . وعلى اعتبار أن شيفرة الأسكي (ASCII) لأزرار لوحة المفاتيح يمكن أن تكون بطول بايت أو أكثر . فإن المتحول (Var) يمكن أن يكون من نوع (Byte) أو (String) .	`Var = Getatkbd()`

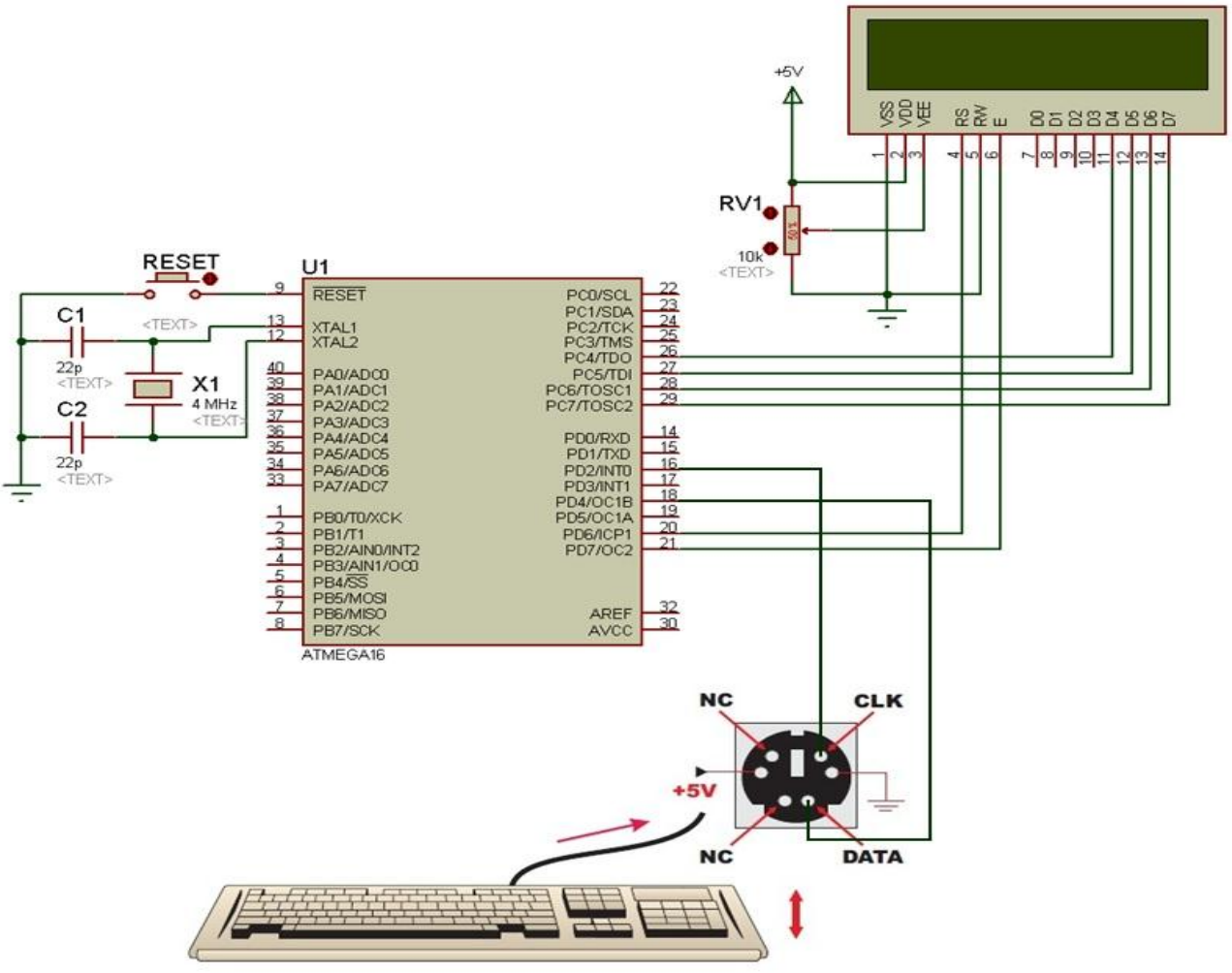
ملاحظة هامة :

عند ضغط أحد أزرار لوحة المفاتيح تقوم شريحة (Keyboard BIOS) الخاصة باللوحة بإرسال شيفرة الزر المضغوط عبر منفذ (PS/2) إلى المتحكم وبالتالي حتى يستطيع المتحكم الصغري معرفة الزر المضغوط عليه أن يمتلك جدول شيفرات الأسكي (ASCII) ضمن أحدى ذواكره . وبالتالي نقوم بتخزين جدول شيفرات الأسكي في ذاكرة البرنامج عبر التعليمة (Data) بحيث يعود إليه المتحكم لدى ضغط أي زر من اللوحة :

التعليمات التالية تقوم بتخزين جدول شيفرات الأسكي (ASCII) ضمن ذاكرة برنامج المتحكم عند اللافتة (Keydata) :

```
Keydata:
'normal keys lower case
Data 0 , 0 , 0 , 0 , 0 , 200 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , &H5E , 0
Data 0 , 0 , 0 , 0 , 0 , 113 , 49 , 0 , 0 , 0 , 122 , 115 , 97 , 119 , 50 , 0
Data 0 , 99 , 120 , 100 , 101 , 52 , 51 , 0 , 0 , 32 , 118 , 102 , 116 , 114 , 53 , 0
Data 0 , 110 , 98 , 104 , 103 , 121 , 54 , 7 , 8 , 44 , 109 , 106 , 117 , 55 , 56 , 0
Data 0 , 44 , 107 , 105 , 111 , 48 , 57 , 0 , 0 , 46 , 45 , 108 , 48 , 112 , 43 , 0
Data 0 , 0 , 0 , 0 , 0 , 92 , 0 , 0 , 0 , 0 , 13 , 0 , 0 , 92 , 0 , 0
Data 0 , 60 , 0 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0

'shifted keys UPPER case
Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
Data 0 , 0 , 0 , 0 , 0 , 81 , 33 , 0 , 0 , 0 , 90 , 83 , 65 , 87 , 34 , 0
Data 0 , 67 , 88 , 68 , 69 , 0 , 35 , 0 , 0 , 32 , 86 , 70 , 84 , 82 , 37 , 0
Data 0 , 78 , 66 , 72 , 71 , 89 , 38 , 0 , 0 , 76 , 77 , 74 , 85 , 47 , 40 , 0
Data 0 , 59 , 75 , 73 , 79 , 61 , 41 , 0 , 0 , 58 , 95 , 76 , 48 , 80 , 63 , 0
Data 0 , 0 , 0 , 0 , 0 , 96 , 0 , 0 , 0 , 0 , 13 , 94 , 0 , 42 , 0 , 0
Data 0 , 62 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0 , 0
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0
```

الدارة العملية :



البرنامج :

سنقوم بكتابة برنامج للمتحكم الصغري (ATmega16) يقوم باستقبال محارف من لوحة مفاتيح حاسب (AT Keyboard) موصولة معه وفق المخطط السابق ويُظهرها على شاشة إظهار محرفيّة (LCD) بأبعاد (16×2) . ولتحقيق ذلك سنعتمد طريقة سهلة يقدمها مترجم (BASCOM AVR) في استقبال الأحرف من لوحة المفاتيح وعرضها على شاشة (LCD) . حيث أننا سنقوم بإعادة توجيه النافذة التسلسليّة الموجودة ضمن المتحكم الصغري ليكون دخلها من لوحة المفاتيح وخرجها على شاشة الإظهار (LCD) مباشرةً بدلاً من أن يكون دخلها وخرجها من أقطاب المتحكم الصغري . وهذا يتطلّب مجموعة من التعليمات المعقدة تتضمّن تعليمات بلغة الإسمبلي . و سنقوم خلال هذا البرنامج بتوضيح بعض التعليمات الجديدة علماً بأن النافذة التسلسليّة سيتم شرحها كاملة مع تعليماتها في جلسات قادمة بمشيئة اللّه عز وجل .

```
$regfile = "m16def.dat"
$crystal = 4000000
$baud = 9600
$hwstack = 32
$swstack = 10
$framesize = 40

Config Lcd = 16 * 2

Config Lcdpin = Pin , Db4 = Portb.3 , Db5 = Portb.2 , Db6 = Portb.1 ,
Db7 = Portb.0 , Rs = Porta.3 , E = Porta.4

Config Keyboard = Pind.2 , Data = Pind.4 , Keydata = Keydata

Dim Text As String * 16
Dim I As Byte
Dim L As Byte

$serialinput = Kbdinput
$serialinput2lcd

Text = "This is a Test!"
Cursor Off
Cls
Print Text
Wait 3

Do
Cls
Print "Enter Text :"
Locate 2 , 1
Cursor On Blink
Input Text
Print Text
If Len(text) = 16 Then Text = ""
Loop
End
```

```
Kbdinput:

$asm
push r16            ; save used register
push r25
push r26
push r27

Kbdinput1:
rCall _getatkbd    ; call the function
tst r24            ; check for zero
breq Kbdinput1      ; yes so try again
pop r27            ; we got a valid key so restore registers
pop r26
pop r25
pop r16
$end Asm

Return

I = Getatkbd()

Keydata:
'normal keys lower case
Data 0 , 0 , 0 , 0 , 0 , 200 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , &H5E , 0
Data 0 , 0 , 0 , 0 , 0 , 113 , 49 , 0 , 0 , 0 , 122 , 115 , 97 , 119 ,
50 , 0
Data 0 , 99 , 120 , 100 , 101 , 52 , 51 , 0 , 0 , 32 , 118 , 102 , 116 ,
114 , 53 , 0
Data 0 , 110 , 98 , 104 , 103 , 121 , 54 , 7 , 8 , 44 , 109 , 106 , 117
, 55 , 56 , 0
Data 0 , 44 , 107 , 105 , 111 , 48 , 57 , 0 , 0 , 46 , 45 , 108 , 48 ,
112 , 43 , 0
Data 0 , 0 , 0 , 0 , 0 , 92 , 0 , 0 , 0 , 0 , 13 , 0 , 0 , 92 , 0 , 0
Data 0 , 60 , 0 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 ,
0 , 0
'shifted keys UPPER case
Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
Data 0 , 0 , 0 , 0 , 0 , 81 , 33 , 0 , 0 , 0 , 90 , 83 , 65 , 87 , 34 ,
0
Data 0 , 67 , 88 , 68 , 69 , 0 , 35 , 0 , 0 , 32 , 86 , 70 , 84 , 82 ,
37 , 0
Data 0 , 78 , 66 , 72 , 71 , 89 , 38 , 0 , 0 , 76 , 77 , 74 , 85 , 47 ,
40 , 0
Data 0 , 59 , 75 , 73 , 79 , 61 , 41 , 0 , 0 , 58 , 95 , 76 , 48 , 80 ,
63 , 0
Data 0 , 0 , 0 , 0 , 0 , 96 , 0 , 0 , 0 , 0 , 13 , 94 , 0 , 42 , 0 , 0
Data 0 , 62 , 0 , 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0 , 0 , 0
Data 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0 , 0 , 43 , 51 , 45 , 42 , 57 ,
0 , 0
```

الجلسة السابعة

المقاطعات الخارجيّة – التعامل مع الذاكرة (EEPROM)

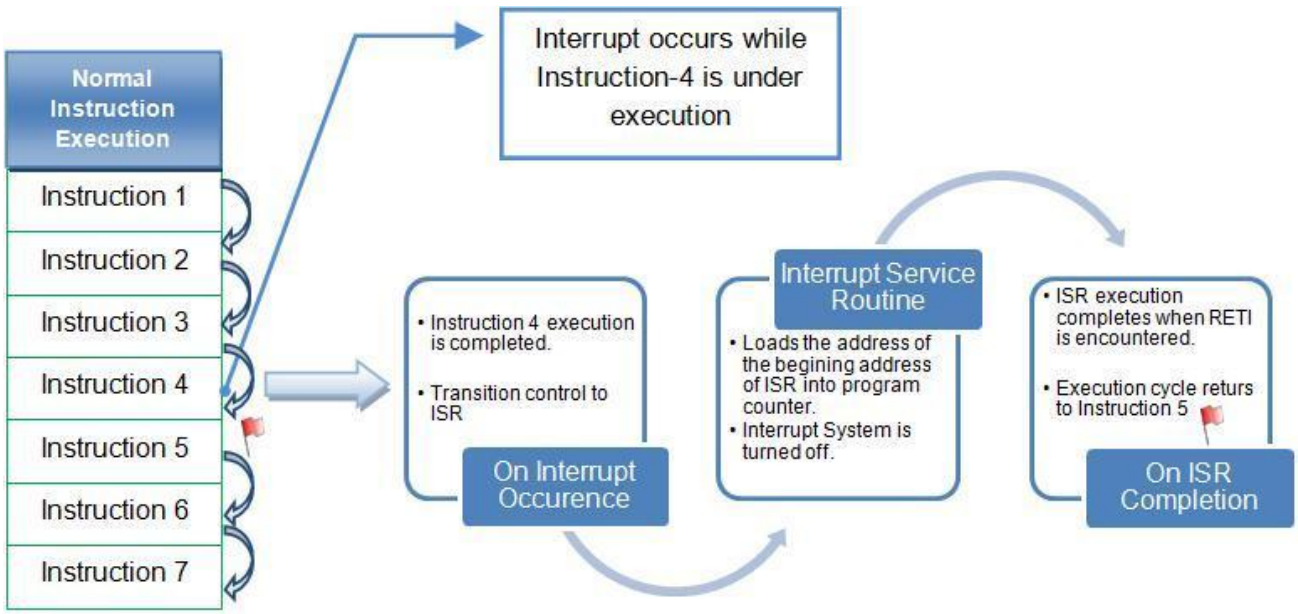
أولاً : المقاطعات الخارجية في متحكمات (ATMEL AVR)

نظرة عامة على المقاطعات (Interrupts) :

يمكن تعريف المقاطعة (Interrupt) بشكل عام بأنها حادثة تُسبّب توقف المتحكم عن تنفيذ البرنامج الرئيسي مؤقّتاً ليذهب لتنفيذ روتين خدمة المقاطعة (ISR) (Interrupt Service Routine) ابتداءاً من عنوان مُحدد في الذاكرة , وبعد الانتهاء منه يعود لتنفيذ البرنامج الرئيسي ابتداءاً من التعليمة التي توقّف عندها لدى حدوث المقاطعة .

يختلف نوع المقاطعة باختلاف مصدرها , فهناك *المقاطعة الداخليّة* والتي تنتج عن الوحدات الداخلية للمتحكم , كمقاطعة طفحان المؤقت / العداد (1) , ومقاطعة المقارن التشابهي , ومقاطعة نظير المؤقت / العداد (1) ..... الخ . وهناك *المقاطعة الخارجية* والتي تحدث نتيجة ورود إشارة خارجيّة على أحد أقطاب المتحكم المُخصّصة لاستقبال إشارات المقاطعة الخارجيّة كالمقاطعات الخارجية (Int0 , Int1) ومقاطعة تصفير المتحكم (Reset) .

لدى ورود إشارة مقاطعة ما (داخليّة أو خارجيّة) على المتحكم يقوم الأخير بإنهاء تنفيذ التعليمة الحالية من البرنامج الرئيسي ليدفع (PUSH) عنوان التعليمة التالية إلى المكدس (Stack) ثم يذهب إلى عنوان روتين خدمة المقاطعة (ISR Address) التي حدثت و المُحدد مُسبقاً بحسب نوع المقاطعة ليقوم بتنفيذ روتين (برنامج) المقاطعة وعند الانتهاء منه يقوم بسحب (POP) عنوان التعليمة التي توقف عندها في البرنامج الرئيسي من المكدس ليتابع تنفيذه ابتداءاً منها .



يمتلك كل متحكم من متحكمات (AVR) عدد من المقاطعات تختلف من متحكم إلى آخر , إلاّ أنّ طريقة التعامل معها واحدة . حيث تمتلك كل مقاطعة من هذه المقاطعات عنواناً محجوزاً لها في الذاكرة ليقوم المتحكم بالتوجّه إليه لدى حدوث هذه المقاطعة . وبالتالي فإن تعليمات روتين خدمة المقاطعة يجب أن تُكتب ابتداءاً من عنوان خدمة المقاطعة الخاص بها . ويُبيّن الجدول التالي المقاطعات الخاصة بالمتحكم (ATmega16) مع العنوان المحجوز لكل مقاطعة في ذاكرة البرنامج :

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

ملاحظة (1) :

قبل التعامل مع أي مقاطعة في البرنامج يجب في البداية تفعيل خانة المقاطعات العامة (I) في المتحكم المطلوب . حيث أن هذه الخانة تقع في البت الثامن من مسجل التحكم (SREG) ويؤدي تصفيرها إلى حجب كامل المقاطعات عن الحدوث داخل المتحكم الصغري :

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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	

ملاحظة (2) :

تتحدّد أولويّات المقاطعات السابقة بحسب عنوان كل مقاطعة وذلك وفق القاعدة التالية :

(المقاطعة ذات العنوان الأصغر لها الأولويّة على المقاطعة ذات العنوان الأكبر)

ملاحظة (3) :

تمتلك كل مقاطعة من المقاطعات السابقة بشكل أساسي خانتين تقعان ضمن مسجلات التحكم الخاصة بكل مقاطعة ويمكن شرح هاتين الخانتين على الشكل التالي :

- *الخانة الأولى* هي خانة تفعيل المقاطعة حيث أن لكل مقاطعة خانة تفعيل خاصة بها عند وضع القيمة (1) منطقي في هذه الخانة يتم تفعيل المقاطعة ويتم حجبها بوضع (0) منطقي في خانة التفعيل الخاصة بها .
- *الخانة الثانية* هي علم المقاطعة حيث يتم رفع هذه الخانة إلى (1) منطقي لدى دخول المتحكم إلى روتين خدمة المقاطعة (برنامج المقاطعة) وتبقى هذه الخانة (1) منطقي إلى أن ينتهي المتحكم من تنفيذ روتين خدمة المقاطعة عندها تعود إلى (0) منطقي لتكون مُهيّأةً لاستقبال إشارة المقاطعة من جديد وهكذا ..... . إن الغاية الأساسية من هذه الخانة هي ضمان عدم حدوث مقاطعة أثناء تنفيذ روتين خدمة مقاطعة من نفس النمط .

مثال : المقاطعات الخارجيّة (INT0 , INT1 , and INT2) تمتلك خانات تفعيل في مسجل التحكم (GICR) وخانات تعمل كأعلام لتلك المقاطعات موجودة ضمن مسجل التحكم (GIFR) . كما هو مُبيّن بالشكلين التاليين :

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	

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	

آليّة تنفيذ المقاطعات (Interrupts Handling) :

إن فهم آليّة تنفيذ المقاطعات في متحكمات (ATMEL AVR) يساعد على استثمارها في البرامج المكتوبة للمتحكم سواءاً أكانت تلك البرامج مكتوبةً بلغة التجميع (Assembly Language) أو بلغات أخرى عالية المستوى (High Level Programming Languages) . في البداية وقبل كل شيء إذا أردنا استخدام المقاطعات في برنامج ما علينا تفعيل خانة تمكين المقاطعات العامة (I) في بداية البرنامج . ومن ثمّ علينا تفعيل خانة تمكين المقاطعة المُراد استخدامها في البرنامج وجميع هذه التعليمات تُكتب في قسم التهيئة من البرنامج .

يمكن تمثيل خانة تمكين المقاطعات العامة (I) كقاطع رئيسي للمقاطعات في المتحكم

بالمقابل فإن خانات تمكين المقاطعات تُمثّل كقواطع فرعية تفصل وتوصل كل مقاطعة على حدى

وبالتالي فإن كل مقاطعة تُمثّل بمفتاح لدى تشغيله تعمل المقاطعة وبفصلة تتوقف عن العمل

لدى ورود إشارة مقاطعة ما إلى المتحكم يقوم معالج المتحكم بتنفيذ الإجراءات التالية :

- إكمال تنفيذ التعليمة المشغول بتنفيذها حاليّاً .
- دفع (PUSH) عنوان التعليمة التالية من البرنامج الرئيسي إلى المكدس (Stack) .
- تحميل عنوان روتين خدمة المقاطعة (ISR Address) التي حدثت في مسجّل عدّاد البرنامج (PC) .
- إلغاء تفعيل خانة تمكين المقاطعات العامة (I) بوضع القيمة (0) منطقي فيها وذلك لحجب حدوث المقاطعات الأخرى خلال تنفيذ روتين خدمة المقاطعة . وبالتالي إذا أراد المبرمج السماح بحدوث المقاطعات خلال تنفيذ روتين خدمة المقاطعة عليه تفعيل خانة تمكين المقاطعات العامة (I) في بداية برنامج المقاطعة .
- رفع خانة علم المقاطعة التي حدثت من (0) إلى (1) منطقي وذلك لحجب حدوث المقاطعة ذاتها مرة أخرى خلال تنفيذ روتين خدمة المقاطعة .
- تنفيذ تعليمات روتين خدمة المقاطعة تعليمة تعليمة إلى أن يصل إلى تعليمة العودة من برنامج خدمة المقاطعة (RETI) (بلغة التجميع).

لدى ورود تعليمة (RETI) في نهاية روتين خدمة المقاطعة يقوم المتحكم بما يلي :

- إعادة خانة تمكين المقاطعات العامة (I) إلى القيمة (1) منطقي لتهيئة المتحكم للاستجابة لحدوث مقاطعات أخرى .
- إنزال خانة علم المقاطعة التي تم تنفيذها من (1) إلى (0) منطقي وذلك لتهيئة المتحكم للاستجابة لحدوث تلك المقاطعة مرة أخرى .
- سحب (POP) قيمة من المكدس (وهي عنوان التعليمة التي سيتابع منها تنفيذ البرنامج الرئيسي) ووضعها في مُسجّل عداد البرنامج (PC) لتتم متابعة تنفيذ سير البرنامج الرئيسي .

ملاحظات :

❖ إذا تحقق شرط إحدى المقاطعات أو أكثر وكانت خانة تمكين المقاطعات العامة غير مُفعّلة (I=0) فإن أعلام المقاطعات التي حدثت سوف ترتفع إلى (1) منطقي وتبقى كذلك إلى أن يتم تفعيل خانة تمكين المقاطعات العامة (I) ليقوم عندها المتحكم بتنفيذ المقاطعات التي حدثت بالترتيب بحسب أولويّة كل مقاطعة .

❖ في حال كان المتحكم يقوم بتنفيذ روتين خدمة مقاطعة ما وحدثت خلال ذلك مقاطعة أخرى عندها يقوم المتحكم برفع علم المقاطعة التي حدثت ويُكمل تنفيذ روتين المقاطعة الحاليّة ومن ثم يعود إلى البرنامج الرئيسي ليقوم بالذهاب منه إلى روتين خدمة المقاطعة التي حدثت خلال التنفيذ ويقوم بمعالجتها . فإذا حدثت أكثر من مقاطعة لدى تنفيذه برنامج مقاطعة ما يقوم عندها بمراكمتها بحسب أولوياتها ليقوم بتنفيذها وفق تسلسل الأولويّة بعد خروجه من برنامج المقاطعة الحالية .

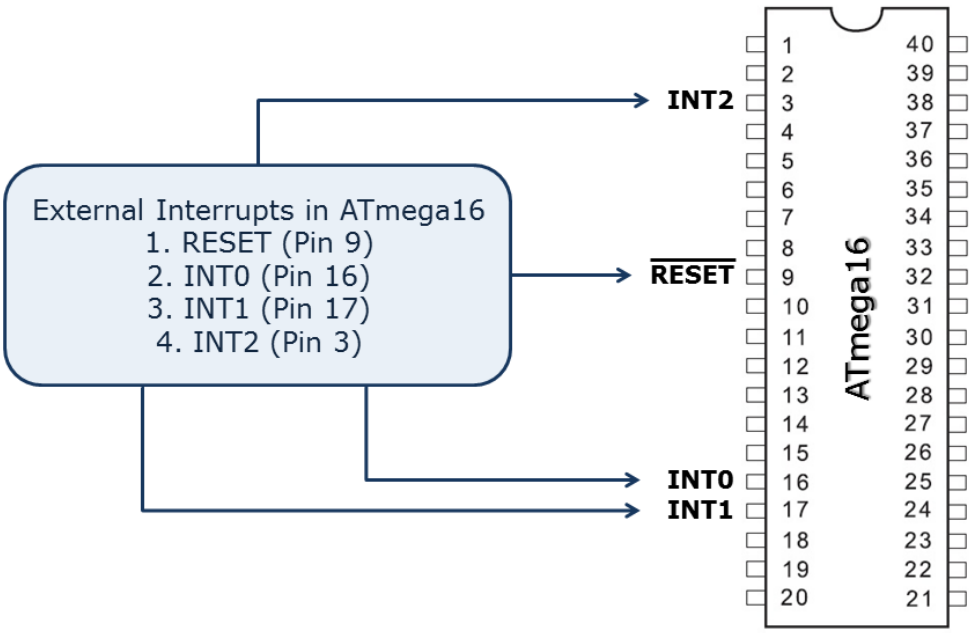
❖ يُوصّى المبرمجون دوماً بتقصير عدد تعليمات روتين خدمة المقاطعة بشكلٍ عام قدر الإمكان كي لا نتسبّب بدخول المتحكم في حلقة طويلة من التعليمات بعيداً عن البرنامج الرئيسي .

المقاطعات الخارجية (External Interrupts) :

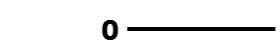
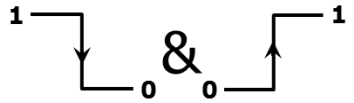
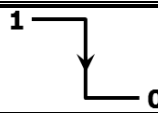
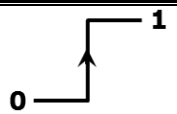
زُوّدت متحكمات شركة (ATMEL) من النوع (AVR) بأقطاب مُهيّأة لاستقبال إشارات مقاطعة خارجيّة . حيث أن هذه الأقطاب تستقبل إشارات من نوع معيّن تُسبب ورودها على القطب توليد حادثة المقاطعة التي تأخذ المتحكم إلى روتين خدمة المقاطعة حيث يُنفّذ تعليمات المقاطعة ومن ثمّ يعود إلى البرنامج الرئيسي.

يمتلك المتحكم (ATmega16) أربعة أقطاب مُهيأة لاستقبال إشارات مقاطعة خارجية هي :

- قطب التصفير (Reset) .
- قطب المقاطعة الخارجيّة (INT0) .
- قطب المقاطعة الخارجيّة (INT1) .
- قطب المقاطعة الخارجيّة (INT2) .

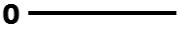
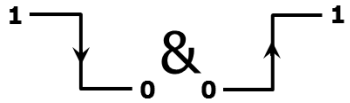
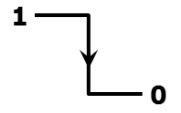
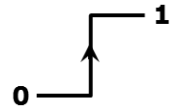


هناك أيضاً أربعة أنواع من الإشارات التي يُمكن أن ترد على أقطاب المقاطعة الخارجية في المتحكم (ATmega16) والتي تُسبب حدوث حالة المقاطعة وهذه الإشارات هي :

الحالة	شكل الإشارة التي تُسبب حدوث المقاطعة	الإشارة
Low Level		إشارة مستوي منخفض (0) منطقي
Change		تغيير الجبهة على القطب (إشارة جبهة هابطة و صاعدة)
Falling		إشارة جبهة هابطة
Rising		إشارة جبهة صاعدة

تعليمات المقاطعات الخارجية في لغة البرمجة (BASCOM AVR) :

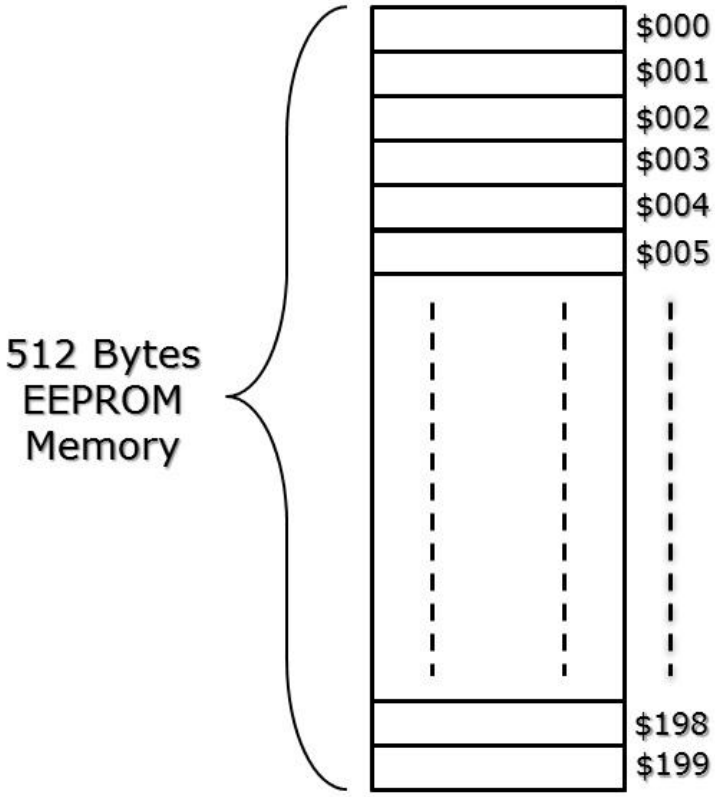
هناك عدة تعليمات في مترجم (BASCOM AVR) تتحكم بسير المقاطعات الخارجية وبتفعيل وإلغاء تفعيل كل مقاطعة على حدى , كما تُحدد نوع الإشارة الواردة على القطب والتي ستحدث عندها المقاطعة . يمكن تلخيص هذه التعليمات بالجدول التالي :

<table>
<tr><td>Enable Interrupts</td><td>تفعيل خانة تمكين المقاطعات العامة</td></tr>
<tr><td>Disable Interrupts</td><td>إلغاء تفعيل خانة تمكين المقاطعات العامة وبالتالي حجب جميع المقاطعات عن الحدوث في المتحكم الصغري</td></tr>
<tr><td>Enable INTx</td><td>تفعيل خانة تمكين المقاطعة الخارجيّة (INTx)</td></tr>
<tr><td>Disable INTx</td><td>إلغاء تفعيل خانة تمكين المقاطعة الخارجيّة (INTx) وبالتالي حجب هذه المقاطعة عن الحدوث</td></tr>
<tr><td>Config INTx = State</td><td>تحديد نوع الإشارة الواردة على قطب المقاطعة الخارجية (INTx) والمسؤولة عن توليد طلب المقاطعة. حيث أن (State) تأخذ الاحتمالات التالية :
<table>
<tr><th>State</th><th>Description</th><th>Interrupt Signal Form</th></tr>
<tr><td>Low Level</td><td>The low level of INTx generates an interrupt request.</td><td> 
</td></tr>
<tr><td>Change</td><td>Any logical change on INTx generates an .interrupt request</td><td> 
</td></tr>
<tr><td>Falling</td><td>The falling edge of INTx generates an interrupt .request</td><td> 
</td></tr>
<tr><td>Rising</td><td>The rising edge of INTx generates an interrupt .request</td><td> 
</td></tr>
</table></td></tr>
<tr><td>ON INTx Lable [NoSave]</td><td>تحديد لافتة (Lable) يقفز إليها البرنامج عند ورود إشارة مقاطعة على القطب (INTx) وبالتالي فإن برنامج خدمة المقاطعة سيكون مكتوباً ابتداءاً من هذه اللافتة .
عند وضع العبارة الاختيارية (NoSave) لن يقوم المتحكم بحفظ أي قيمة من قيم مسجلات الأغراض العامة أو مسجل الحالة (SREG) . وعند إهمال هذه العبارة سيقوم المتحكم بحفظ قيم تلك المسجلات السابقة لاستردادها في نهاية برنامج المقاطعة</td></tr>
<tr><td>Return</td><td>وهي تعليمة العودة من برنامج المقاطعة ويجب كتابتها دوماً كآخر تعليمة في برنامج المقاطعة (تُكافئ التعليمة (RETI) في لغة التجميع)</td></tr>
</table>

ثانياً : التعامل مع الذاكرة (EEPROM)

ذكرنا سابقاً أن متحكمات (ATMEL AVR) تمتلك ثلاثة أنواع من الذواكر وهي ذاكرة البرنامج (Flash Memory) , ذاكرة المعطيات (SRAM) , والذاكرة الدائمة (EEPROM) . سنقوم في هذه الفقرة بدراسة الذاكرة الدائمة (EEPROM) وكيفية تخزين البيانات عليها ومن ثمّ قراءتها منها .

بنية الذاكرة الدائمة (EEPROM) :



زُوّدت جميع متحكمات (ATMEL) من النوع (AVR) بذاكرة من النوع (EEPROM) يختلف حجمها من متحكم إلى آخر . ففي المتحكم (ATmega16) يبلغ حجم هذه الذاكرة (512) بايت وتمتاز بأنها ذاكرة دائمة لا تفقد معلوماتها بانقطاع التغذية الكهربائية عنها (Non-volatile Memory) .

يتوفر في متحكمات (AVR) نوعين من الذواكر الدائمة وهما ذاكرة البرنامج (Flash Memory) والذاكرة الدائمة (EEPROM) .

تُخصص ذاكرة البرنامج لتخزين شيفرات تعليمات برنامج المتحكم ولا يمكن تخزين بيانات عليها خلال سير عمل البرنامج من خلال تعليمات معينة وإنما يستلزم ذلك إعادة برمجة المتحكم لتخزين بيانات ثابتة عليها (عن طريق التعليمة (Data)) .

من هذا المنطلق برزت أهمية الذاكرة (EEPROM) . إذ أنها ذاكرة دائمة ويُمكن القراءة والكتابة عليها من داخل برنامج المتحكم دون الحاجة إلى إعادة برمجة المتحكم وذلك من خلال تعليمات قراءة وكتابة خاصة بها . إلا أنّ عيبها الوحيد هو الزمن الكبير الذي يستهلكه المتحكم في تنفيذ عملية الكتابة عليها بينما تستهلك عملية الكتابة على ذاكرة البرنامج (Flash Memory) زمناً أقل منه بكثير .أما في عملية القراءة فتستهلك كِلا الذاكرتين نفس الزمن في التنفيذ

ملاحظات :

- يتم حجز موقع المتحولات المُعرّفة من خلال تعليمة (Dim) في الذاكرة (SRAM) من المتحكم بشكل افتراضي وبالتالي فإن هذه المتحولات تفقد قِيَمها لدى فصل المتحكم عن التغذية أو عند ضغط زر (RESET) . وحتى نضمن الحفاظ على تلك القيم علينا تخزين متحولاتها في ذاكرة دائمة .
- من الشكل المُبيّن أعلاه نلاحظ أنّ أول حجرة في الذاكرة (EEPROM) عنوانها ($000) ست عشري أو (0) عشرياً وآخر حجرة منها عنوانها ($199) ست عشري أو (511) عشرياً وبالتالي علينا الحرص على أن تكون المتحولات المحجوزة في تلك الذاكرة ضمن هذا المجال من العناوين .
- يُوصى من شركة (ATMEL) بعدم استخدام البايت الأول من الذاكرة (EEPROM) (ذو العنوان ($000)) وذلك لعدم ضمان تغيير قيمته عند تصفير البرنامج .

تعليمات التعامل مع الذاكرة (EEPROM) في لغة البرمجة (BASCOM AVR) :

قدّمت لغة البرمجة (BASCOM AVR) طريقةً سهلةً وبسيطةً للتعامل مع الذاكرة (EEPROM) , إذ يكفي أن تُعرّف متحولاً ما ضمن هذه الذاكرة ليتم التعامل معه في القراءة والكتابة بمجرد إسناده إلى متحول آخر .

يُلخّص الجدول التالي تعليمات التعامل مع الذاكرة الدائمة (EEPROM) :

`Dim Var AS [SRAM] Type [at Location]`
 	
`Examples:`
`Dim X AS SRAM Byte`
`Dim Y AS Double at &H65`
`Dim A(12) AS Word at 110`
`Dim B(20) AS String * 4 at &H98`
	تعريف متحول ضمن الذاكرة (SRAM) من النوع (Type) عند الحجرة ذات العناون (Location) .
	نوع المتحول (Type) يمكن أن يكون ضمن الأصناف التالية: (bit , Byte , Integer , Word , Long , Single ,Double , String * x)
	يجب الانتباه هنا إلى عدم حجز مواقع في الذاكرة (SRAM) تتضارب مع مسجلات الأغراض العامة ومسجلات التحكم

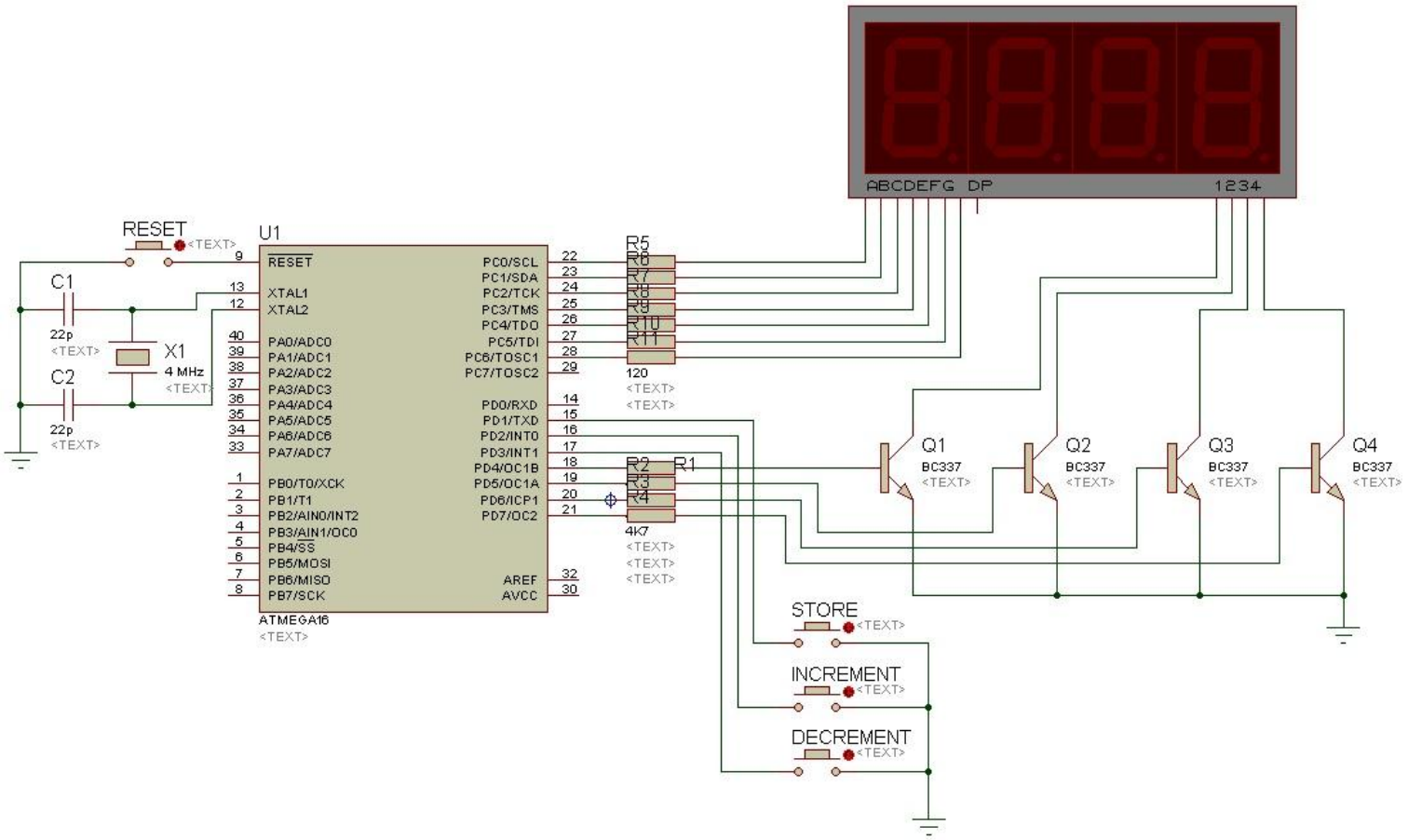
`Dim Var AS ERAM Type [at Location]`
 	
`Examples:`
`Dim X1 AS ERAM Byte at 12`
`Dim Y4 AS ERAM Single at &H60`
`Dim Arr(12) AS ERAM Word at 110`
	تعريف متحول ضمن الذاكرة (EEPROM) من النوع (Type) عند الحجرة ذات العناون (Location) .
	نوع المتحول (Type) يمكن أن يكون ضمن الأصناف التالية: (bit , Byte , Integer , Word , Long , Single ,Double , String * x)

`$eeprom`	توجيه للمترجم يقوم بعده بتخزين جميع المعطيات المكتوبة وفق تعليمة (Data) في ذاكرة (EEPROM) وليس في ذاكرة البرنامج . وعندها سيتم توليد ملف بمتداد (.EPP) يحتوي على المعطيات المُخزّنة في الذاكرة (EEPROM) بالصيغة الثنائية (Binary Code)
`$eepromhex`	توجيه للمترجم بأن يقوم بتخزين المعطيات في الملف ذو الامتداد (.EPP) بالصيغة الست عشريّة (Hexadecimal) وليس الصيغة الثنائيّة (Binary Code) .
`$data`	توجيه للمترجم يقوم بعده بتخزين جميع المعطيات المكتوبة وفق تعليمة (Data) في ذاكرة البرنامج (وهي الحالة الافتراضيّة للتعليمة (Data))
`$default Sram | Eram`	تعيين الموقع الافتراضي لحجز متحولات الذاكرة في البرنامج عن طريق تعليمة (Dim) .
`$end $default`	استعادة الوضع الافتراضي لحجز متحولات الذاكرة في البرنامج عن طريق تعليمة (Dim) . (الذاكرة (SRAM))
`Writeeeprom Var , Address`	كتابة قيمة المتحول (Var) في حجرة الذاكرة ذات العنوان (Address) من الذاكرة الدائمة (EEPROM)

كتابة قيمة المتحول (Var) في الذاكرة الدائمة (EEPROM) ابتداءاً من اللافتة (Lcabel)	**Writeeeprom** Var , Label
قراءة القيمة المُخزّنة في حجرة الذاكرة ذات العنوان (Address) من الذاكرة الدائمة (EEPROM) ووضعها في المتحول (Var)	**Readeeprom** Var , Address
قراءة القيمة المُخزّنة في حجرة الذاكرة الواقعة عند اللافتة (Label) من الذاكرة الدائمة (EEPROM) ووضعها في المتحول (Var)	**Readeeprom** Var , Label

الدارة العمليّة :

في هذه الدارة سنستثمر المقاطعتين الخارجيتين (INT0 , INT1) ونستخدم الذاكرة الدائمة (EEPROM) في تخزين رقم العد الظاهر على شاشة السبع قطع (SDD) . حيث أننا نقوم بوصل زر لحظي (INCREMENT) على قطب المقاطعة (INT0) وزر لحظي آخر (DECREMENT) على زر المقاطعة (INT1) وزر لحظي ثالث (STORE) على قطب الدخل (PD1) :



البرنامج :

في هذا البرنامج سنقوم بعرض رقم على شاشة السبع قطع (SSD) بحيث تخضع قيمته للقواعد التالية:

- عند ضغط زر (INCREMENT) يقوم المتحكم بزيادة الرقم المعروض بمقدار (1) .
- عند ضغط زر (DECREMENT) يقوم المتحكم بإنقاص الرقم المعروض بمقدار (1) .
- عند ضغط زر (STORE) يقوم المتحكم بتخزين الرقم المعروض في ذاكرة (EEPROM) .

سنقوم بالتعامل مع الزر اللحظي (STORE) فقط كقطب دخل أما الزرين اللحظيين (INCREMENT) و (DECREMENT) فسنقوم باستخدام المقاطعتين الخارجيتين (INT1 , INT0) لتأمين وظيفة كلاً من هاذين الزرين .

الجلسة الثامنة

المؤقت/العداد (0) (Timer/Counter 0)

نظرة عامة على المؤقتات / العدادات (Timers/Counters) :

تحتوي متحكمات (ATMEL AVR) على الكثير من الكيانات الداخليّة المزروعة على شريحة المتحكم الداخلية والتي يستخدمها معالج المتحكم بحسب البرنامج المكتوب بداخله . ويُعد كيان (المُؤقّت/العداد) (Timer/Counter) من أهم الكيانات المزروعة داخل المتحكم الصغري . فهو الكيان الذي يسمح للمتحكم بتعريف الوقت لديه وبالتالي استخدام ذلك في كتابة برامج توقيت وعدّ. كبرامج الساعات والعدادات والمؤقتات الالكترونية .

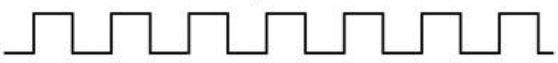
(المؤقت / العداد) وظيفتان مختلفتان لكيان واحد :

لابدّ أنّك تساءلت مِراراً وتكراراً عن سبب التصاق اسم *المؤقت بالعداد* في جميع العبارات التي تتكلم عن هذا الكيان في المتحكم الصغري . الحقيقة أن السبب في ذلك يعود إلى أن الكيان الداخلي الواحد (المؤقت/العداد) يمكن أن يعمل كمؤقت ويمكن أن يعمل كعدّاد وذلك بحسب تهيئته في بداية البرنامج . فعندما نحتاج إلى استخدام الوقت ضمن البرنامج يعمل عندها هذا الكيان على عدّ نبضات داخلية منتظمة ويُسمّى **مؤقتاً** . وعندما نحتاج إلى عدّ نبضات خارجيّة غير منتظمة نقوم بتشغيل هذا الكيان **كعدّاد** .

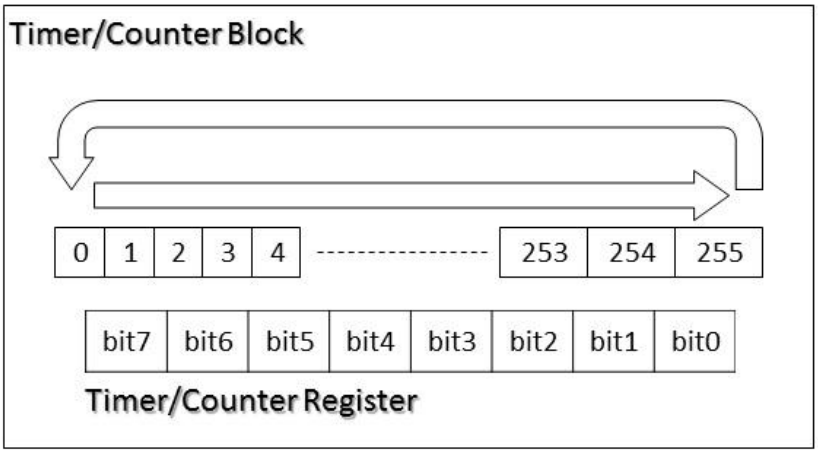
مبدأ عمل (المؤقت / العداد) :

ببساطة يتألف كيان المؤقت/العداد من مسجل تحكّم (بطول (8-bits) أو بطول (16-bits)) يُسمّى مسجل (المؤقت/العداد) تزداد قيمته بمقدار عدّة واحدة كلما وردت على هذا الكيان نبضة عدّ واحدة . هذه النبضات يمكن أن تكون نبضات داخلية (من الكريستالة الموصولة مع المتحكم الصغري) و منتظمة فيسمى عندها مؤقتاً . أو نبضات خارجية غير منتظمة ويسمى عندها عدّاداً . وفي كلا الحالتين عندما يصل مسجل المؤقت/العداد إلى القيمة العُظمى ((255) في حالة مسجل (8-bits) و (65535) في حالة مسجل (16-bits)) يعود إلى الصفر بعد أن يُعلم المتحكم بأنه قد وصل إلى حالة الطفحان .

نبضات مُنتظمة داخليّة (واردة من كريستالة المتحكم أو من مقسم ترددات الكريستالة)
(نمط المؤقت)



نبضات غير مُنتظمة خارجية (واردة من أحد الأقطاب الخارجية الخاصة بالعدادات)
(نمط العداد)



المؤقت/العداد (0) (Timer/Counter 0)

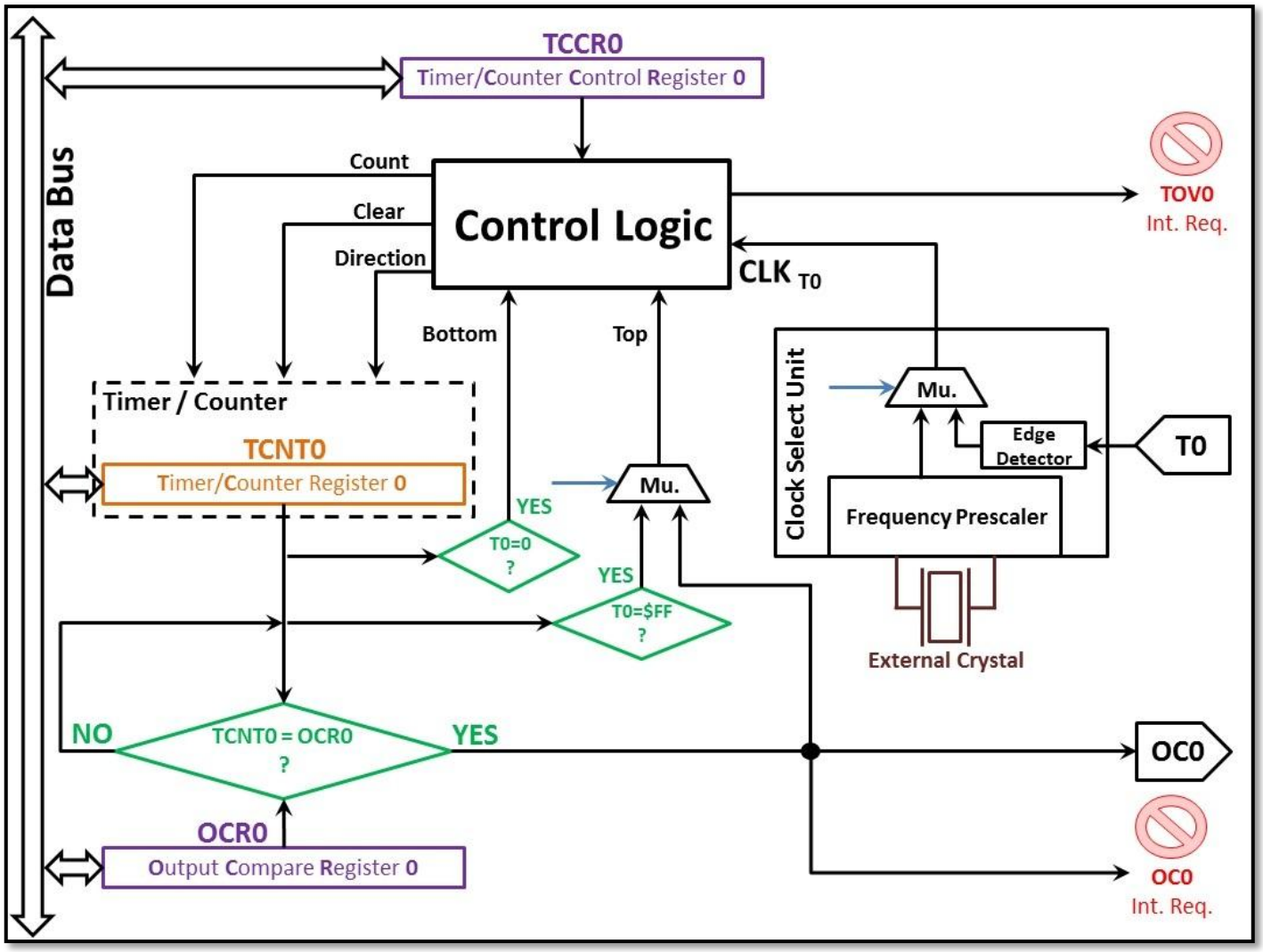
تحتوي معظم متحكمات (ATMEL AVR) بداخلها على أكثر من كيان داخلي يعمل كمؤقت/عداد . بعضها بطول (8-bits) وبعضها الآخر بطول (16-bits) . بالنسبة للمتحكم (Atmega16) يحتوي بداخله على ثلاثة كيانات تعمل كمؤقت/عداد هي :

- المؤقت / العداد (0) (Timer / Counter 0) : وهو بطول (8-bits) .
- المؤقت / العداد (1) (Timer / Counter 1) : وهو بطول (16-bits) .
- المؤقت / العداد (2) (Timer / Counter 2) : وهو بطول (8-bits) .

لا تختلف المؤقتات الثلاثة السابقة بعرض مسجل المؤقت/العداد فحسب وإنما تختلف أيضاً ببعض الميزات والخصائص التي زوِّدت بها شركة (ATMEL) مؤقّتات دون أخرى . إلا أنها وعلى اختلاف ميزاتها تمتلك طريقة واحدة في التعامل سندرسها في هذه الجلسة حيث نبدأ بدراسة المؤقت/العداد (0) (Timer/Counter 0) من المتحكم (Atmega16) ونترك دراسة باقي المؤقتات إلى الجلسات القادمة بمشيئة الله تعالى .

بنية المؤقت / العداد (0) :

يُبيّن الشكل التالي البنية المنطقيّة للمؤقت/العداد (0) :



نلاحظ من الشكل السابق أن المؤقت/العداد (0) يمتلك ثلاثة مسجلات تحكم خاصة به يتم من خلالها التعامل معه هي :

- ❖ مسجل التحكم بالمؤقت/العداد (0) (TCCR0) .
- ❖ مسجل المؤقت/العداد (0) (TCNT0) .
- ❖ مسجل نظير المقارنة للمؤقت/العداد (0) (OCR0) .

كما أنه يمتلك مقاطعتين ضمن أشعة المقاطعة الخاصة بالمتحكم (ATmega16) هما :

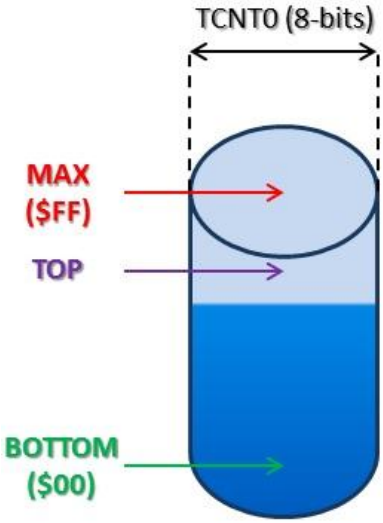
- مقاطعة طفحان المؤقت/العداد (0) (TOV0) .
- مقاطعة تطابق نظير المقارنة في المؤقت/العداد (0) (OC0) .

وحدة منطق التحكم (Control Logic) :

تتحكّم وحدة منطق التحكم (Control Logic) بعمل المؤقت/العداد (0) حيث أنها تقوم بتهيئة المؤقت/العداد (0) للعمل وفق شيفرة الإعدادات التي يكتبها المبرمج ضمن مسجل التحكم (TCCR0) . ويخرج منها ثلاث إشارات أوامر داخلية تقوم بالتحكّم بالقيمة المُخزّنة في مسجل المؤقت/العداد (0) (TCNT0) :

- ✓ الإشارة (Count) تُؤدي إلى زيادة أو إنقاص قيمة المسجل (TCNT0) بمقدار (1) .
- ✓ الإشارة (Clear) تُؤدي إلى تصفير قيمة المسجل (TCNT0) .
- ✓ الإشارة (Direction) تُحدّد هذه الإشارة فيما إذا كانت قيمة المسجل (TCNT0) ستتغيّر بالزيادة (Increment) أو بالنقصان (Decrement) .

تقوم وحدة منطق التحكم (Control Logic) بتخريج إشارات الأوامر بناءاً على إشارات الدخل القادمة إليها . حيث تُعبّر هذه الإشارات عن حالة المؤقت/العداد (0) الآنيّة . فمثلاً عند وصول قيمة المسجل (TCNT0) إلى أعلى قيمة له تُفعّل نتيجةً لذلك إشارة الدخل (Top) فتقوم وحدة منطق التحكم بإرسال إشارة الأمر (Clear) إلى المسجل (TCNT0) لإعادته إلى قيمته الأوليّة ($00) , وعندما تصل قيمته إلى الصفر (في حالة العد التنازلي) تُفعّل نتيجةً لذلك إشارة الدخل (Bottom) لتقوم وحدة منطق التحكم بتحميل القيمة ($FF) ضمن المسجل (TCNT0) . وهكذا تقوم وحدة منطق التحكم بإعطاء إشارات الأوامر لدى استقبالها لإشارات الدخل المعبّرة عن حالة المؤقت/العداد (0) .



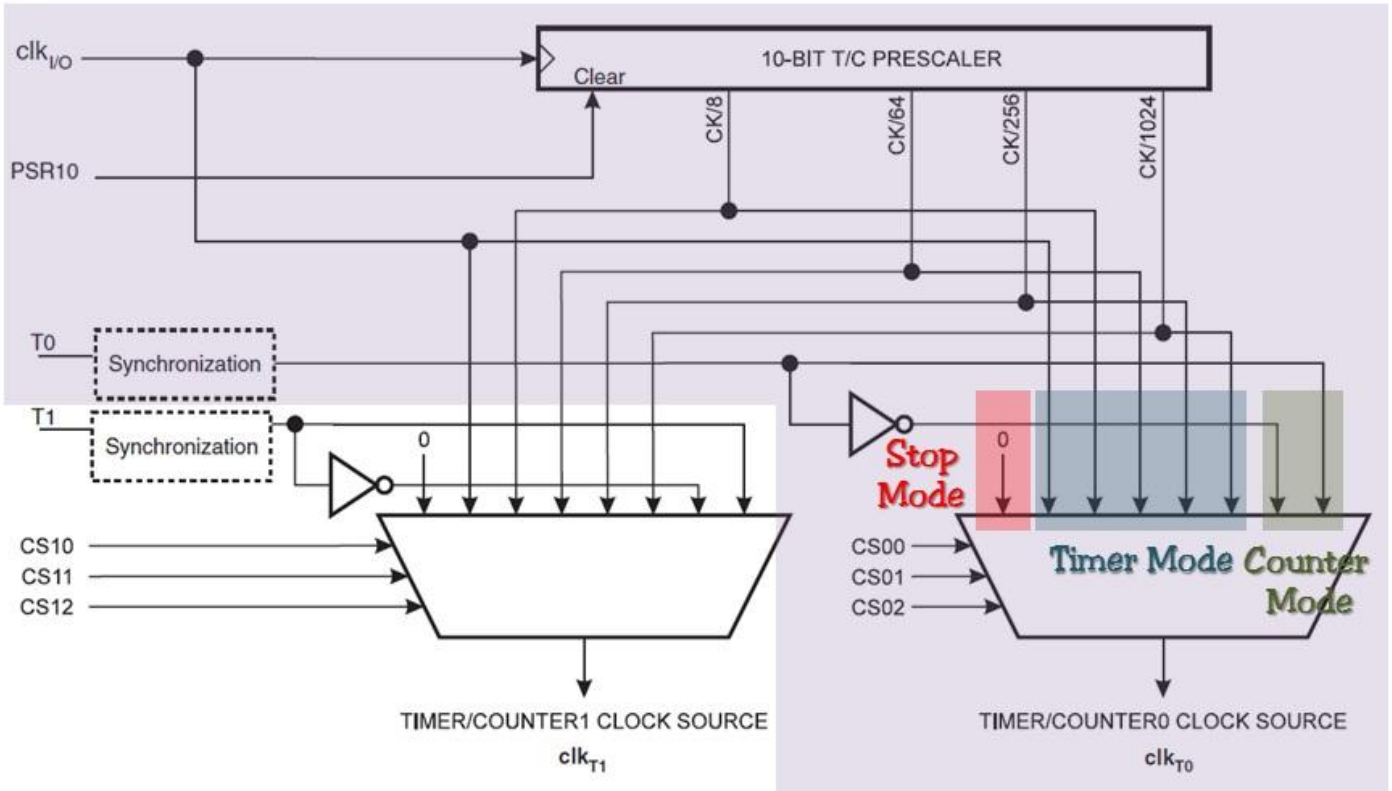
ويُمكن أن نُميّز هنا ثلاث قيم يبلغها مسجل المؤقت/العداد (0) (TCNT0) في عملية العدّ :

- ❖ القيمة (BOTTOM) .
- ❖ القيمة (TOP) .
- ❖ القيمة (MAX) .

BOTTOM	نقول أن المؤقت/العداد (0) وصل إلى القيمة (BOTTOM) عندما تكون قيمة المسجل (TCNT0=$00)
TOP	نقول أن المؤقت/العداد (0) وصل إلى القيمة (TOP) عندما تكون قيمة العدّ في المسجل (TCNT0) وصلت إلى أعلى قيمة في سلسلة العدّ . هذه القيمة يمكن أن تكون (MAX) أي ($FF) ويمكن أن تكون القيمة المُخزّنة في مسجل نظير المقارنة (OCR0) . وهذا يعتمد على نمط عمل المؤقت/العداد (0) المُبرمج للعمل به
MAX	نقول أن المؤقت/العداد (0) وصل إلى القيمة (MAX) عندما تكون قيمة المسجل (TCNT0=$FF)

وحدة اختيار مصدر نبضات الساعة (Clock Select Unit) :

تقوم هذه الوحدة بتحديد مصدر نبضات الساعة (CLK) التي سيقوم المؤقت/العداد (0) بعدّها . حيث يمكن أن يكون هذا المصدر داخليّاً (من مقسّم التردد الداخلي) أو خارجيّاً (من قطب المتحكم (T0)) . ويتم الاختيار بين مصادر نبضات الساعة للمؤقت/العداد (0) عبر ناخب (Multiplexer) وذلك بحسب نمط العمل المُبرمج عليه هذا الكيان . ويُبيّن الشكل التالي بنية وحدة اختيار مصدر نبضات الساعة عند المؤقت/العداد (0) وهي مُشتركة بينه وبين المؤقت/العداد (1) :



نلاحظ من الشكل السابق وجود مقسم تردد بعرض (10 bit) (10-bit Frequency Prescaler) يسمح بتقسيم إشارة التردد الواردة من الكريستالة الخارجيّة إلى إشارات ذات ترددات أصغر ليتم إدخالها إلى كيان المؤقت/العداد (0) فيقوم بعدّ نبضاتها في حال كونه يعمل بنمط "المؤقت" .

يمكن لكيان المؤقت/العداد (0) وفقاً لمصدر نبضات الساعة (CLK_{TO}) الداخلة إليه أن يكون ضمن أحد الأنماط الثلاثة التالية :

نمط التوقّف (Stop Mode) .

نمط المؤقّت (Timer Mode) .

نمط العدّاد (Counter Mode) .

Work Mode	Clock Source	CLK_{TO}			
		CLK= 1 MHz	CLK= 2 MHz	CLK= 4 MHz	CLK= 8 MHz
Stop Mode	0	---	---	---	---
Timer Mode	CLK	1 MHz	2 MHz	4 MHz	8 MHz
	CLK/8	125 KHz	250 KHz	500 KHz	1 MHz
	CLK/64	15625 Hz	31250 Hz	62500 Hz	125 KHz
	CLK/256	3906.25 Hz	7812.5 Hz	15625 Hz	31250 Hz
	CLK/1024	976.56 Hz	1953.1 Hz	3906.25 Hz	7812.5 Hz
Counter Mode	T0	---	---	---	---
	T0	---	---	---	---

مسجلات التحكم الخاصة بالمؤقت / العداد (0) :

على الرغم من أن لغات برمجة متحكمات (AVR) عالية المستوى قد وفّرت بشكل عام تعليمات خاصة للتعامل مع كيان المؤقت / العداد (0) في أي متحكم منها . إلا أنّ الاطلاع على مسجلات التحكم الخاصة بهذا الكيان يفيد في تجريد طريقة عمله وبالتالي زيادة القدرة على تحديد أبسط الطرق في كتابة البرامج التي تستثمر هذا الكيان من قِبل المبرمجين, كما أن كتابة برامج بلغة التجميع (Assembly Language) تستلزم معرفة دقيقة بمسجلات التحكم الخاصة بهذا الكيان .

مسجل المؤقت / العداد (0) (TCNT0) Timer/Counter (0) Register

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

يحتوي هذا المسجل على القيمة الآنيّة للعد حيث تزداد قيمته (أو تنقص) بمقدار واحد عند كل نبضة ترد على كيان المؤقت / العداد (0) . وهو مسجل بعرض (8-bits) تتراوح قيمته بين (0 → 255) عشرياً أو بين (00 → FF) ست عشرياً . وتُعطي متحكمات (AVR) إمكانية الوصول المباشر لهذا المسجل (سواءاً بالقراءة أو بالكتابة) حيث يمكن تهيئة المؤقت / العداد (0) بقيمة ابتدائيّة ما عبر كتابة هذه القيمة ضمن هذا المسجل في بداية عملية العد .

مسجل نظير المقارنة (OCR0) Output Compare Register

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
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	

يُستخدم هذا المُسجّل في حالة تشغيل المؤقت/ العداد (0) وفق نمط "مقاطعة تطابق نظير المقارنة" حيث يقوم المبرمج في هذا النمط بوضع قيمة ما في هذا المسجل (تتراوح ما بين (0 → 255)) . ويقوم كيان المؤقت / العداد (0) عند كل نبضة بمقارنة القيمة الآنية للعد (المسجل (TCNT0)) مع قيمة المسجل (OCR0) وعند تطابق القيمتين يذهب إلى مقاطعة نظير المقارنة (Output Compare Interrupt). كما ويمكن أيضاً تخريج قيمة منطقيّة معينة على القطب (OC0) عند حدوث التطابق بين المسجلين .

مسجل التحكم بالمؤقت / العدّاد (0) (TCCR0)

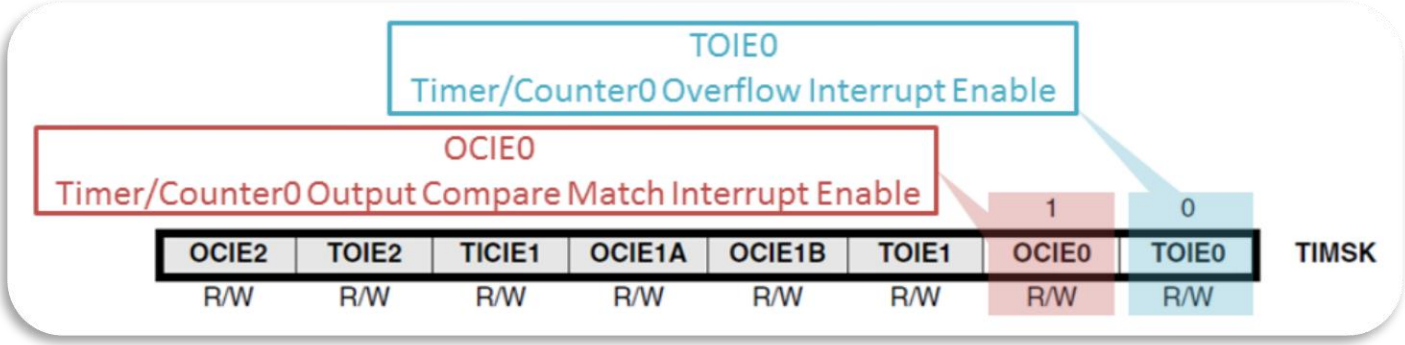
Timer/Counter (0) Control Register

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	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	

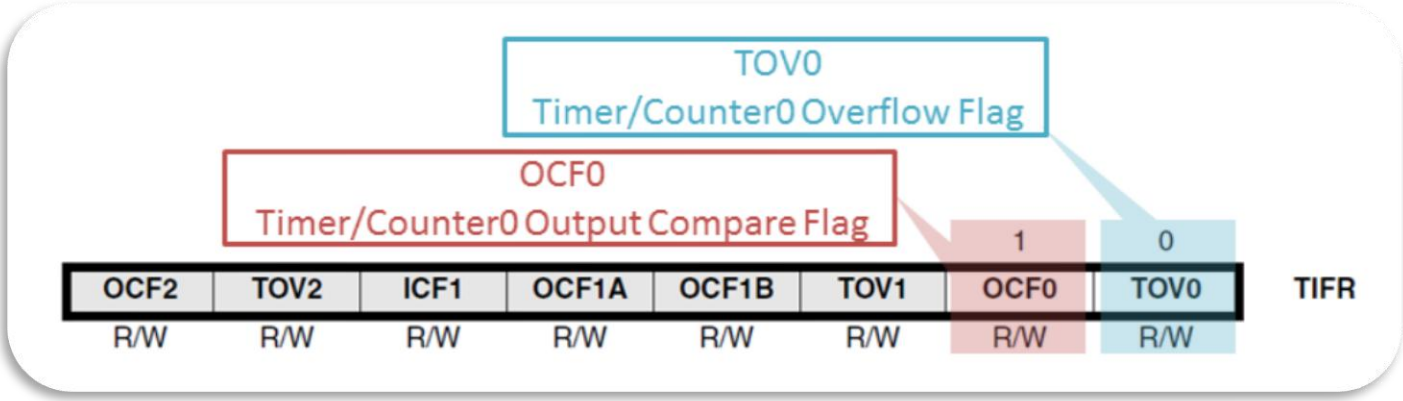
يتم عن طريق بتات هذا المسجل التحكم بعمل وسلوك كيان المؤقت / العداد (0) ضمن المتحكم. بحيث يقوم المبرمج بكتابة بايت تحكم مناسب فيه في البداية لتحديد طريقة عمل هذا الكيان ضمن البرنامج .
يمكن مراجعة وثيقة الشريحة (ATmega16) للحصول على معلومات تفصيليّة عن هذا المسجل .

وعلى اعتبار أن كيان المؤقت / العداد (0) يمتلك مقاطعتين فقط (مقاطعة طفحان المؤقت / العداد (0) و مقاطعة تطابق نظير المقارنة) فإن ذلك يستلزم وجود بتّين لكل مقاطعة : بت تفعيل وبت علم مقاطعة.
تتواجد هذه البتات ضمن المسجلين التاليين :

مسجل قناع مقاطعة المؤقت/العداد (TIMSK)



مسجل أعلام مقاطعة المؤقت/العداد (TIRF)



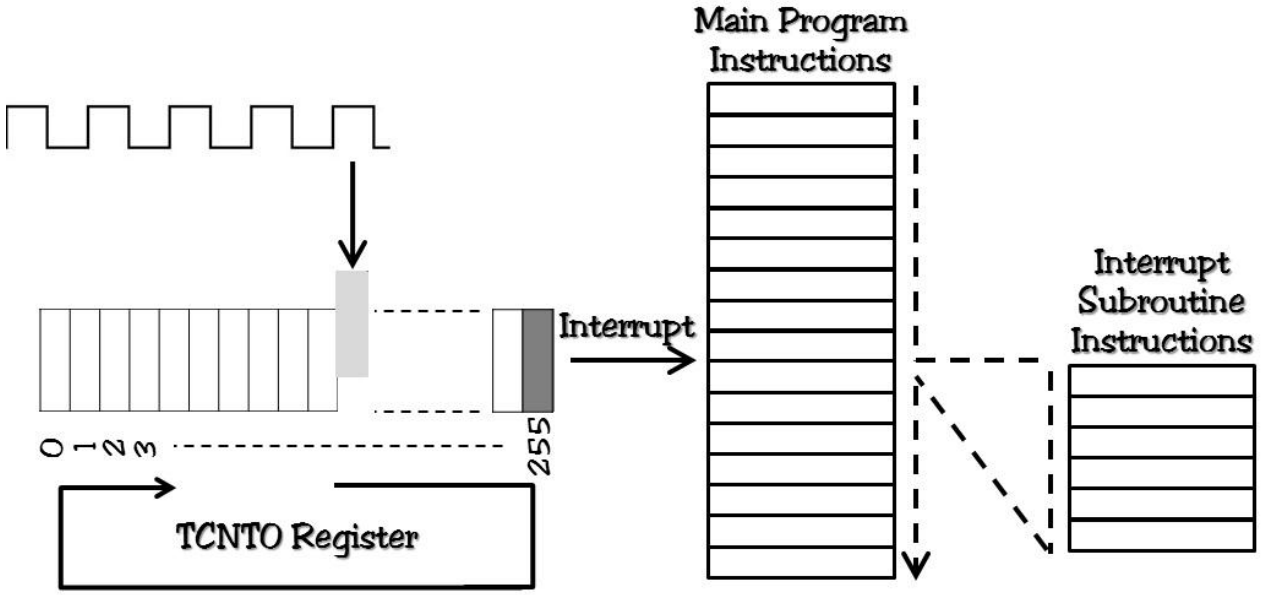
أنماط عمل المؤقت/العداد (0)
(Timer/Counter (0) Operation Modes)

أولاً : أنماط عمله كمؤقت (Timer Modes) :

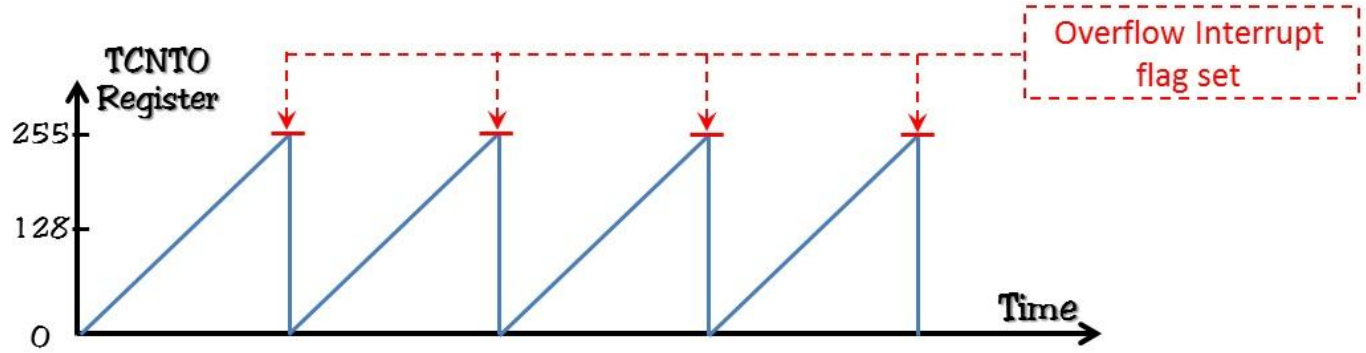
حتى نستطيع استثمار كيان المؤقت / العداد (0) داخل المتحكم الصغري كمؤقت يجب أن نُفعّله ليعمل ضمن أحد أنماط العمل التالية :

نمط العمل العادي (Normal Mode) :

وهو أبسط نمط عمل للمؤقت , حيث يتم فيه تفعيل هذا الكيان ليعمل كمؤقّت فيقوم بزيادة محتوى المسجل (TCNT0) بمقدار واحد عند كل نبضة واردة إلى كيان المؤقت (0) , وعند الوصول إلى القيمة العظمى للمسجل (TCNT0) (MAX=255) يقوم الكيان بتصفير المسجل (TCNT0) والذهاب إلى برنامج مقاطعة طفحان المؤقت / العدّاد (0) (هذا إن كانت المقاطعة مُفعّلة) ومن ثم يتابع العد التصاعدي طالما أن كيان المؤقت / العدّاد (0) مُفعّلاً . ويُمكن تهيئة المسجل (TCNT0) بقيمة ابتدائية ليبدأ العد منها ويُفضّل أن تُكتب التعليمة التي تقوم بذلك ضمن برنامج مقاطعة طفحان المؤقت / العدّاد (0) .



ويُبيّن الشكل التالي المخطط الزمني للمؤقت (0) في نمط العمل العادي (Normal Mode) :

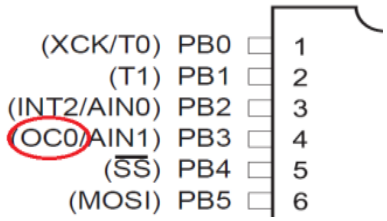


نمط تصفير المؤقت عند تطابق نظير المقارنة (CTC Mode) :

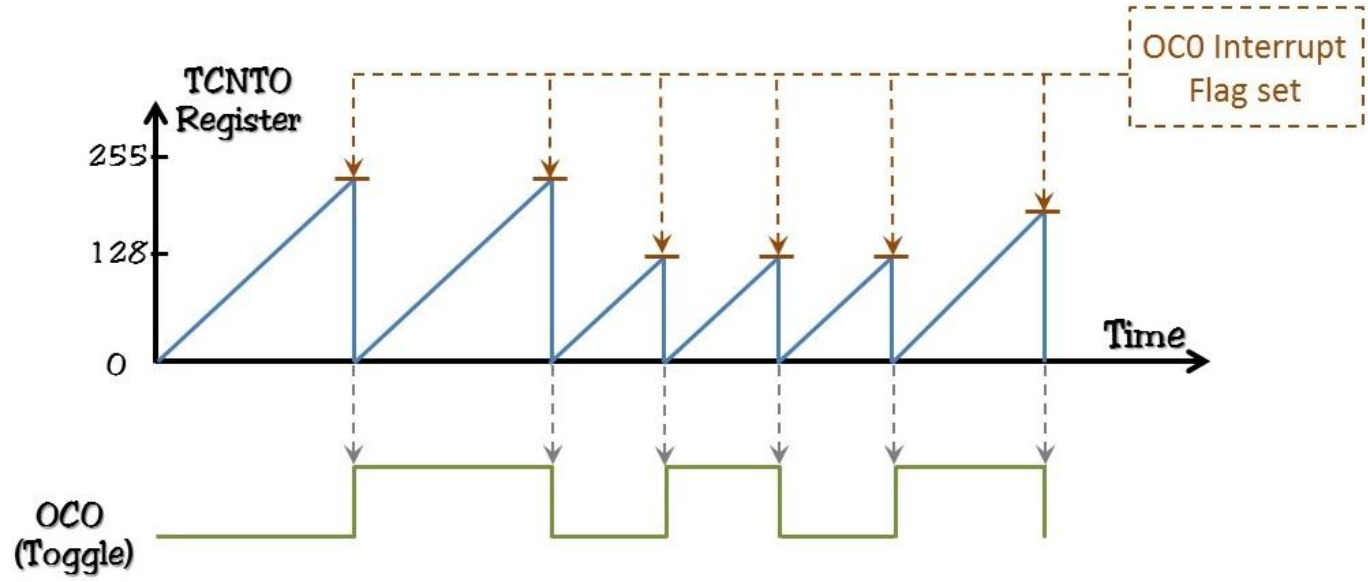
(Clear Timer on Compare Match Mode)

في هذا النمط يقوم كيان المؤقت (0) بزيادة قيمة المسجل (TCNT0) عند كل نبضة ترد إليه. وفي كل مرة يقوم بمقارنة قيمة المسجل (TCNT0) مع القيمة الموجودة ضمن مسجل نظير المقارنة (OC0). وعند تطابق القيمتين ضمن المسجلين يقوم المتحكم الصغري بتصفير قيمة المؤقت/العداد (0) (TCNT0 = 0) ومن ثم يقوم بما يلي : (تحتاج كل وظيفة إلى تفعيل على حدى)

- ❖ يذهب إلى برنامج خدمة مقاطعة "تطابق نظير المقارنة" للمؤقت / العداد (0) لينفذ تعليماته ويعود بعد انتهائه إلى البرنامج الرئيسي .
- ❖ يقوم بتخريج قيمة منطقيّة مُعيّنة (مُحدّدة مسبقاً) على القطب (OC0) .



وبالتالي يمكننا من خلال تشغيل كيان المؤقت / العداد (0) في هذا النمط توليد قطار من النبضات على القطب (OC0) وذلك لدى تهيئته لعكس المنطق المتوضّع عليه عند كل حدوث لمقاطعة تطابق نظير المقارنة كما هو مُبيّن في المخطط الزمني التالي :

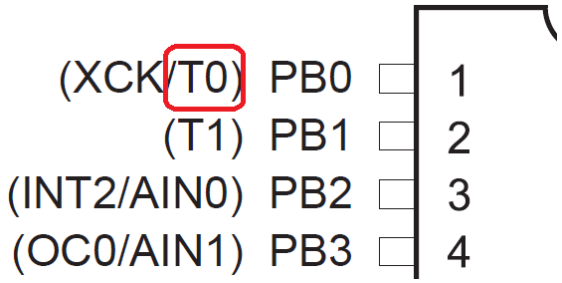


ونترك دراسة أنماط عمل (PWM Modes) للمؤقت / العداد (0) للدرس القادم بمشيئة اللّه تعالى حيث سندرس أنماط عمل (PWM) للمؤقت / العداد (1) وما ينطبق عليه ينطبق على المؤقت / العداد (0) .

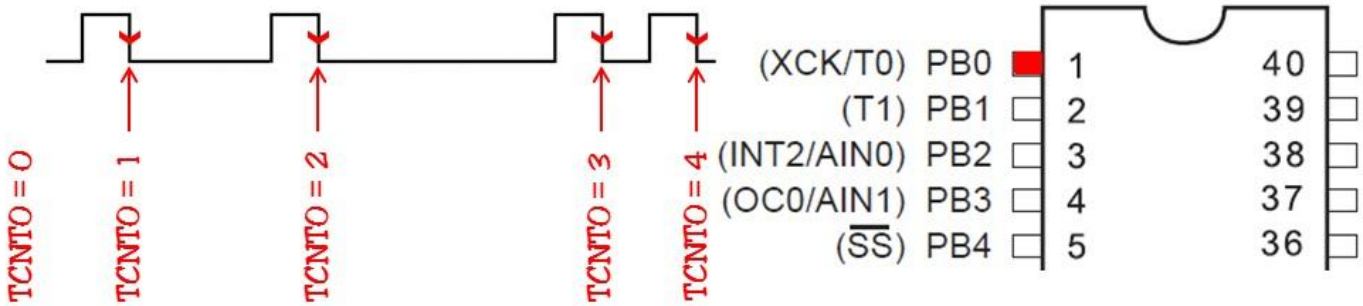
ثانياً : أنماط عمله كعدّاد (Counter Modes) :

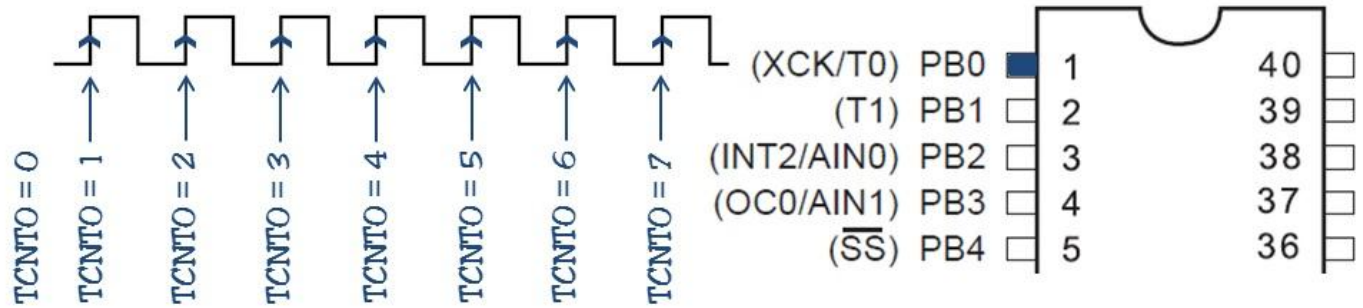
يُمكن لكيان المؤقت / العداد (0) أن يعمل كعدّاد بأحد الأنماط السابقة مع ملاحظة اختلاف مصدر النبضات التي يعدّها. ففي حالة المؤقت (Timer) يقوم بعدّ نبضات داخلية منتظمة قادمة من مقسّم التردد الخاص بهذا الكيان وبالتالي مصدر النبضات هنا هو الكريستالة الموصولة مع المتحكم.

أمّا في حالة العدّاد (Counter) فهو يعد النبضات الواردة إلى القطب الخارجي (T0) من المتحكم. وبالتالي يعدّ الكيان هنا نبضات خارجيّة يمكن أن تكون منتظمة أو غير منتظمة .



يمكن لكيان المؤقت / العداد (0) أن يعد الجبهات الهابطة للنبضات الخارجيّة الواردة على القطب (T0) أو يعد الجبهات الصاعدة لها وذلك بحسب تهيئة المبرمج له في بداية عمله .





ملاحظة :

لا يتغير أي شيء في خصائص كيان المؤقت / العدّاد (0) في حالة عمله كعدّاد وإنما يختلف فقط مصدر النبضات التي يعدّها. حيث يمكن استخدام أي مقاطعة من مقاطعات هذا الكيان ضمن البرنامج كمقاطعة طفحان المؤقت / العدّاد (0) ومقاطعة تطابق نظير المقارنة .

تعليمات التعامل مع المؤقت / العداد (0) في لغة (BASCOM AVR) :

تُوفّر بيئة (BASCOM AVR) توابع وإجرائيات جاهزة للتعامل مع المؤقتات / العدادات في متحكمات (AVR) . بالإضافة إلى إمكانيّة الوصول إلى مسجلات التحكم الخاصة بهذا الكيان عن طريق كتابة اسم مسجل التحكم مباشرة . ويمكن في بعض المسجلات استخدام اسم مكافئ لتبسيط كتابة البرامج . فمثلاً يمكن الوصول إلى المسجل (TCNT0) عن طريق كتابة اسمه مباشرةً أو عن طريق كتابة (Timer0) .

يُبيّن الجدول التالي كافّة تعليمات التحكم بالمؤقت/العداد (0) في بيئة البرمجة (BASCOM AVR) :

`Config Timer0 = Timer , Prescale = 1|8|64|256|1024 [,Clear_timer = 1|0]`	• تهيئة كيان المؤقت / العداد (0) ليعمل كمؤقت بتقسيمة تردد (1 or 8 or 64 or 256 or 1024). • تحديد تصفير المؤقت (Clear_timer = 1) أو عدم تصفيره (Clear_timer = 0) عند حدوث مقاطعة تطابق نظير المقارنة أي عند تحقق المساواة (TCNT0 = OCR0). • يبدأ المؤقت (0) بالعمل بعد كتابة هذه التعليمة فوراً.
`Config Timer0 = Counter , Edge = Rising | Falling [,Clear_timer = 1|0]`	• تهيئة كيان المؤقت / العداد (0) ليعمل كعدّاد يعد عند الجبهة الصاعدة (Rising) أو الهابطة (Falling). • تحديد تصفير العدّاد (Clear_timer = 1) أو عدم تصفيره (Clear_timer = 0) عند حدوث مقاطعة تطابق نظير المقارنة أي عند تحقق المساواة (TCNT0 = OCR0). • يبدأ العدّاد (0) بالعمل بعد كتابة هذه التعليمة فوراً.
`Stop Timer0`	إيقاف المؤقت / العدّاد (0) عن العمل
`Start Timer0`	إعادة تشغيل المؤقت / العدّاد (0)
`Enable Interrupts`	تفعيل علم المقاطعة العام
`Enable Timer0` or `Enable Ovf0`	تفعيل مقاطعة طفحان المؤقت / العدّاد (0)
`Enable Compare0` or `Enable Oc0`	تفعيل مقاطعة تطابق نظير المقارنة للمؤقت / العدّاد (0)
`On Ovf0 Label`	عند تحقق مقاطعة طفحان المؤقت / العدّاد (0) يذهب المتحكم لتنفيذ تعليمات برنامج المقاطعة ابتداءاً من اللافتة (Label)
`On Compare0 Label`	عند تحقق مقاطعة تطابق نظير المقارنة للمؤقت / العدّاد (0) يذهب المتحكم لتنفيذ برنامج المقاطعة ابتداءاً من اللافتة (Label)
`Timer0=value or TCNT0=value`	إسناد قيمة أوّليّة (Value) إلى المسجل (TCNT0)
`Var=Timer0 or Var=TCNT0`	قراءة قيمة المسجل (TCNT0) ووضعها ضمن المتحول (Var)
`Compare0 = value`	إسناد قيمة أوّليّة (Value) إلى المسجل (OCR0)
`Var = Compare0`	قراءة قيمة المسجل (OCR0) ووضعها ضمن المتحول (Var)

التطبيق العملي :

سنقوم بتصميم ساعة الكترونية اعتماداً على كيان المُؤقت / العدّاد (0) . إلا أنّه وقبل البداية علينا فهم فكرة توليد الثانية ضمن المتحكم الصغري انطلاقاً من الهزاز الكريستالي الموصول معه .

فكرة توليد الثانية :

تقوم فكرة الساعة الالكترونية على زيادة رقم مُعيّن (عدّاد الثواني) بمقدار واحد كل فترة زمنية محددة (ثانية). وفي كل مرة نفحص هذا الرقم فإذا وصل إلى (60) نقوم بزيادة رقم آخر بمقدار واحد (عدّاد الدقائق) ومن ثم إذا وصل عدّاد الدقائق إلى (60) نزيد رقم آخر بمقدار واحد (عدّاد الساعات) وهكذا يتم احتساب اليوم والشهر والسنة .

يقودنا الكلام السابق إلى أن أساس احتساب الوقت والتاريخ هي الثانية فإذا استطعنا توليد الفترة الزمنيّة التي يُعبّر عنها بالثانية نستطيع بعدها احتساب الوقت والتاريخ بكل سهولة .

من المعروف أن العنصر الوحيد المتصل مع المتحكم الصغري والذي يمكن تعريف فترة الثانية من خلاله هو الكريستالة (الهزّاز الكريستالي) . فإذا ما ربطنا المتحكم بكريستالة ذات قيمة (4 MHz) فهذا يعني أنه عند استقبالنا من الكريستالة (4000000) نبضة تكون قد تحققت ثانية واحدة . ووفقاً لذلك نستطيع استخدام كيان المؤقت / العداد (0) ليعدّ النبضات الواردة من الكريستالة وعند وصولها إلى الرقم (4000000) تكون قد مرت ثانية كاملة .

على اعتبار أن المؤقت / العداد (0) مؤقت بعرض (8-bits) فإن أكبر رقم يمكن أن يصل إليه هو (FF=255) وبالتالي لا يمكن إدخال نبضات الكريستالة مباشرةً إلى المؤقت وإنما يتم ذلك عن طريق مُقسّم تردد . فلنختر مثلاً تقسيمة تردد (CLK/1024) . عندها تصبح قيمة الثانية من النبضات :

$$1\ \text{Sec} = \frac{4000000}{1024} = 3906.25 \approx \mathbf{3906\ \ pulses}$$

بذلك وبعد التقسيمة الجديدة نستطيع معرفة مرور ثانية من الوقت كلّما انتهى المؤقت / العداد (0) من عد (3906) نبضة . إلا أن هذا الرقم لايزال أكبر بكثير من أعلى رقم يستطيع المؤقت (0) عدّه (وهو 255) . لذلك سنلجأ هنا إلى الطريقة التالية :

نقوم بتشغيل المؤقت / العداد (0) ليعمل كمؤقت و وفق تقسيمة معينة عندها سيقوم بعد النبضات الواردة إليه من مقسّم التردد إلى أن يطفح ليعود مرة أخرى إلى الصفر ويُعاود عملية العدّ . نستطيع هنا استخدام مقاطعة طفحان المؤقت / العداد (0) لاحتساب العدد اللازم من الطفحانات للمؤقت (0) حتى تمر ثانية واحدة . ولتوضيح ذلك نأخذ المثال الرقمي التالي :

External Crystal = 4 MHz = 4000000 Pulses

Prescaler = CLK/256 $\Longrightarrow$ CLK_{TO} = 1 Sec = 4000000/256 = 15625 Pulses

وبتقسيم العدد (15625) على (256) ينتج لدينا عدد مرات الطفحان التي ينفذها المؤقت / العداد (0) حتى يُحقق زمن ثانية واحدة : $15625/256 = 61.035 \approx 61$

وبالتالي ووفق المعطيات السابقة نقوم باحتساب عدد مرات طفحان المؤقت / العدّاد (0) وعند بلوغ هذا العدد الرقم (61) نكون قد حصّلنا زمن الثانية الواحدة .

الدارة العمليّة :

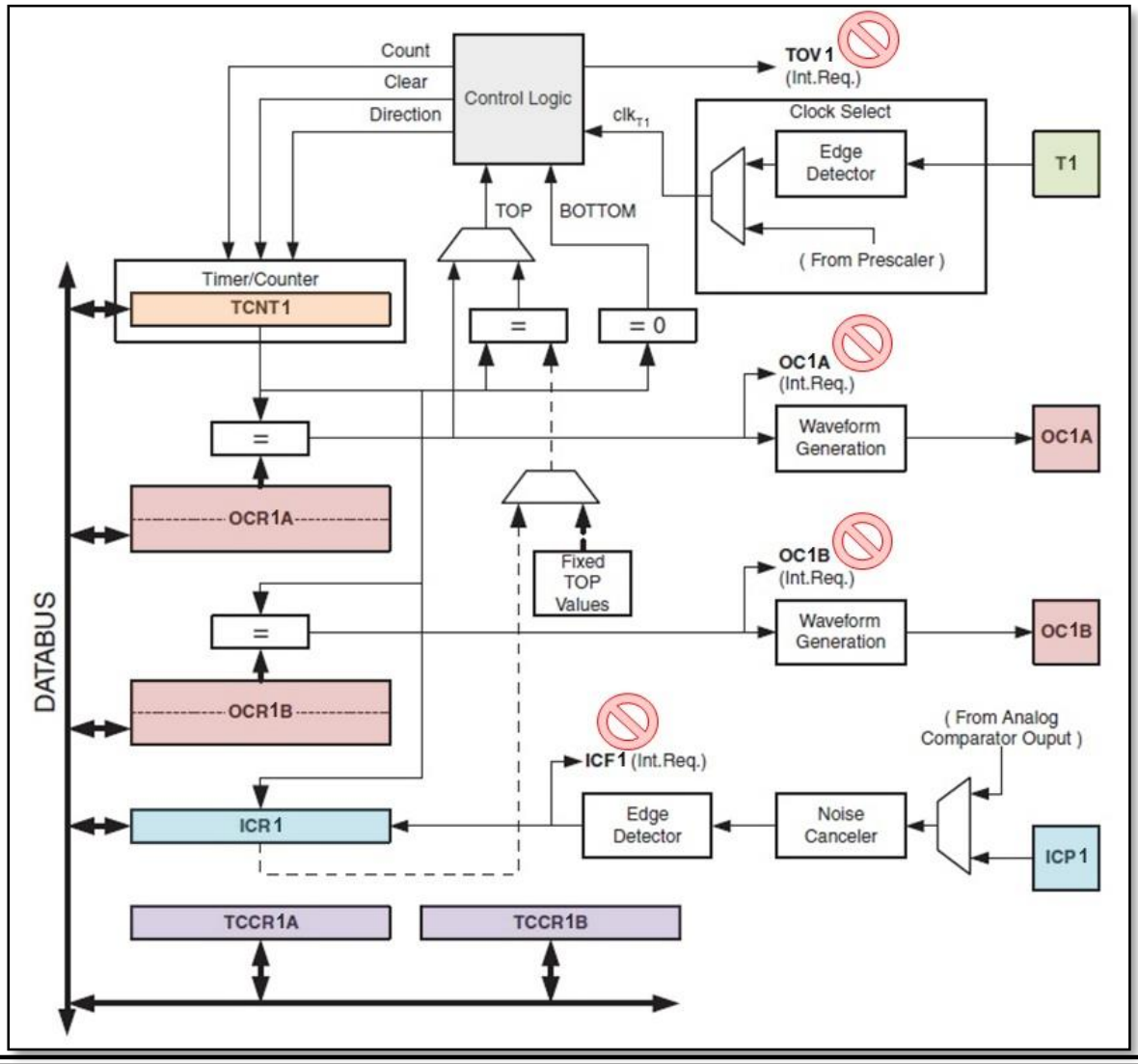
البرنامج :

الجلسة التاسعة

المؤقت/العداد (1) (Timer/Counter 1)

- كيان مؤقت/عداد بعرض (16 - bits) وبالتالي مجال العد فيه (0 → 65535) .
- يمتلك مسجلين لنظير المقارنة (A,B) يمكن استخدامهما بشكل منفصل . يقابلهما قطبين خارجيين لتوليد نبضات (PWM) هما (OC1A , OC1B) وكذلك مقاطعتين لتطابق نظير المقارنة .
- يمتلك مقاطعة إضافية تُسمى مقاطعة "حادثة المسك" ترتبط بالقطب (ICP1) من المتحكم .
- يمكن أن يعمل كمولد تردد (Frequency Generator) في أحد أنماط عمله .
- عدّاد يعدّ الحوادث الخارجية عند الجبهة الهابطة أو الصاعدة .
- يمتلك أربعة مصادر مقاطعة منفصلة (TOV1 ,OC1A ,OC1B ,ICF1) .

البنية المنطقيّة للمؤقت/العداد (1) :



نلاحظ من الشكل السابق أن المؤقت/العداد (1) يمتلك عشرة مسجلات تحكم خاصة به يتم من خلالها التعامل معه هي:

- مسجل التحكم (A) بالمؤقت/العداد (1) (TCCR1A) .
- مسجل التحكم (B) بالمؤقت/العداد (1) (TCCR1B) .
- مسجل المؤقت/العداد (1) (TCNT1) وهو بطول (16-bits) (TCNT1H --- TCNT1L) .
- مسجل نظير المقارنة (A) للمؤقت/العداد (1) (OCR1A) وهو بطول (16-bits) أيضاً مؤلف من اجتماع المسجلين (OCR1AH --- OCR1AL) .
- مسجل نظير المقارنة (B) للمؤقت/العداد (1) (OCR1B) وهو بطول (16-bits) أيضاً مؤلف من اجتماع المسجلين (OCR1BH --- OCR1BL) .
- مسجل حادثة المسك للمؤقت/العداد (1) (ICR1) وهو بطول (16-bits) (ICR1H --- ICR1L).

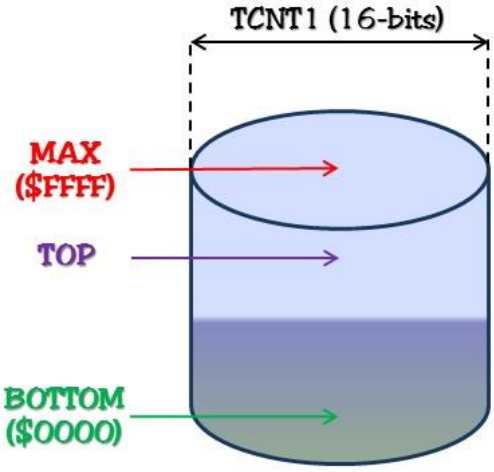
كما أنه يمتلك أربع مقاطعات ضمن أشعة المقاطعة الخاصة بالمتحكم (ATmega16) هم :

- مقاطعة طفحان المؤقت/العداد (1) (TOV1) .
- مقاطعة تطابق نظير المقارنة (A) في المؤقت/العداد (1) (OC1A) .
- مقاطعة تطابق نظير المقارنة (B) في المؤقت/العداد (1) (OC1B) .
- مقاطعة حادثة المسك (ICF1) .

وحدة منطق التحكم (Control Logic) :

تتحكّم وحدة منطق التحكم (Control Logic) بعمل المؤقت/العداد (1) حيث أنها تقوم بتهيئة المؤقت/العداد (1) للعمل وفق شيفرة الإعدادات التي يكتبها المبرمج ضمن مسجلي التحكم الخاصين به (TCCR1A , TCCR1B) . ويخرج منها ثلاث إشارات أوامر داخلية تقوم بالتحكّم بالقيمة المُخزّنة في مسجل المؤقت/العداد (1) (TCNT1) :

- ✓ الإشارة (Count) تُؤدي إلى زيادة أو إنقاص قيمة المسجل (TCNT1) بمقدار (1) .
- ✓ الإشارة (Clear) تُؤدي إلى تصفير قيمة المسجل (TCNT1) .
- ✓ الإشارة (Direction) تُحدّد هذه الإشارة فيما إذا كانت قيمة المسجل (TCNT1) ستتغيّر بالزيادة (Increment) أو بالنقصان (Decrement) .



تقوم وحدة منطق التحكم (Control Logic) بتخريج إشارات الأوامر بناءاً على إشارات الدخل القادمة إليها. و تُعبّر هذه الإشارات عن حالة المؤقت/العداد (1) الآنيّة . فمثلاً عند وصول قيمة المسجل (TCNT1) إلى أعلى قيمة له تُفعّل نتيجةً لذلك إشارة الدخل (Top) فتقوم وحدة منطق التحكم بإرسال إشارة الأمر (Clear) إلى المسجل (TCNT1) لإعادته إلى قيمته الأوليّة ($0000), وعندما تصل قيمته إلى الصفر تُفعّل نتيجةً لذلك إشارة الدخل (Bottom) لتقوم وحدة منطق التحكم بتحميل القيمة ($FFFF) ضمن المسجل (TCNT1).

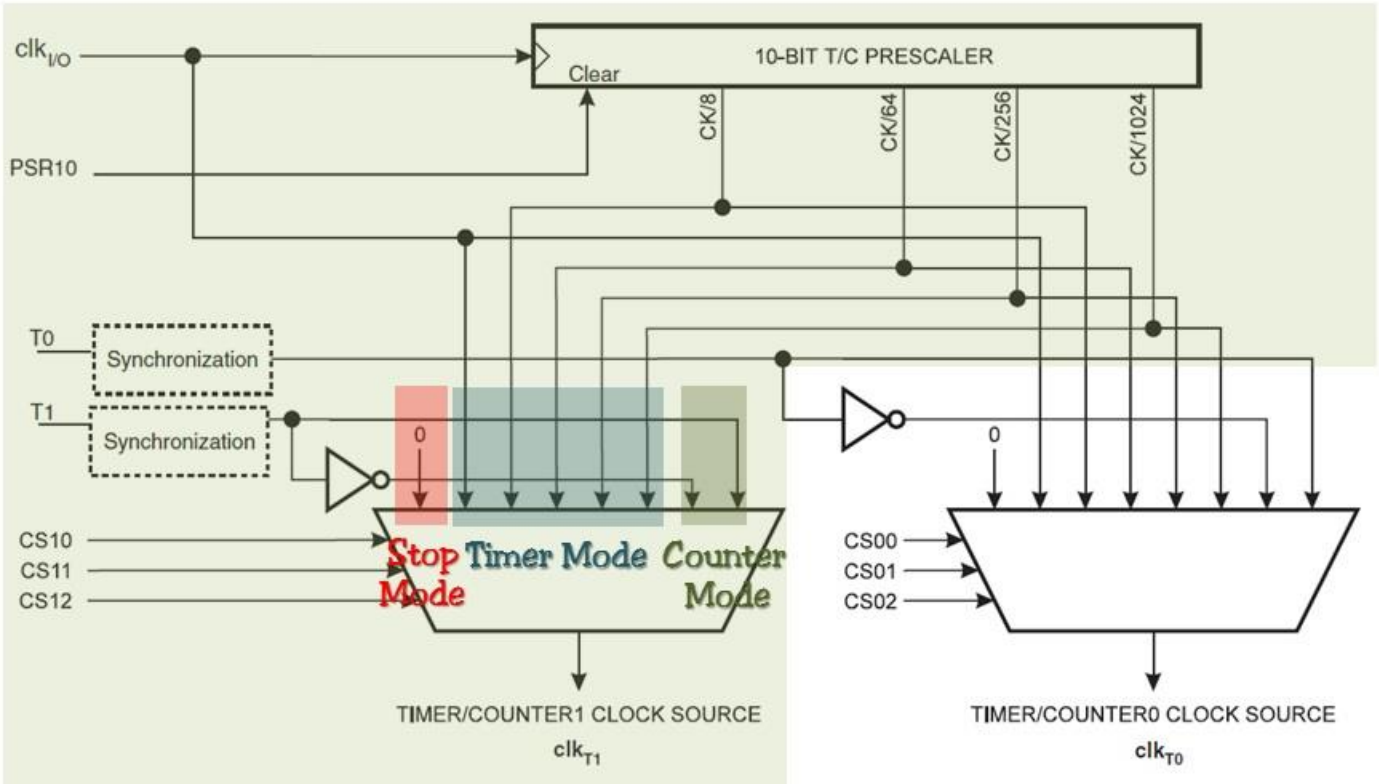
ويُمكن أن نُميّز هنا ثلاث قيم يبلغها مسجل المؤقت/العداد (1) (TCNT1) في عملية العدّ :

- القيمة (BOTTOM) .
- القيمة (TOP) .
- القيمة (MAX) .

BOTTOM	نقول أن المؤقت/العداد (1) وصل إلى القيمة (BOTTOM) عندما تكون قيمة المسجل (TCNT1 = $0000)
TOP	نقول أن المؤقت/العداد (1) وصل إلى القيمة (TOP) عندما تكون قيمة العدّ في المسجل (TCNT1) وصلت إلى أعلى قيمة في سلسلة العدّ . هذه القيمة يمكن أن تكون (MAX) أي ($FFFF) و يمكن أن تكون القيمة المُخزّنة في أحد مسجلي نظير المقارنة (OCR1A ,OCR1B) وذلك يعتمد على نمط عمل المؤقت/العداد (1) المُبرمج للعمل به
MAX	نقول أن المؤقت/العداد (1) وصل إلى القيمة (MAX) عندما تكون قيمة المسجل (TCNT1 = $FFFF)

وحدة اختيار مصدر نبضات الساعة (Clock Select Unit) :

تقوم هذه الوحدة بتحديد مصدر نبضات الساعة (CLK) التي سيقوم المؤقت/العداد (1) بعدّها . حيث يمكن أن يكون هذا المصدر داخليّاً (من مقسّم التردد الداخلي) أو خارجيّاً (من قطب المتحكم (T1)) . ويتم الاختيار بين مصادر نبضات الساعة للمؤقت/العداد (1) عبر ناخب (Multiplexer) وذلك بحسب نمط العمل المُبرمج عليه هذا الكيان . ويُبيّن الشكل التالي بنية وحدة اختيار مصدر نبضات الساعة عند المؤقت/العداد (1) وهي مُشتركة بينه وبين المؤقت/العداد (0) :



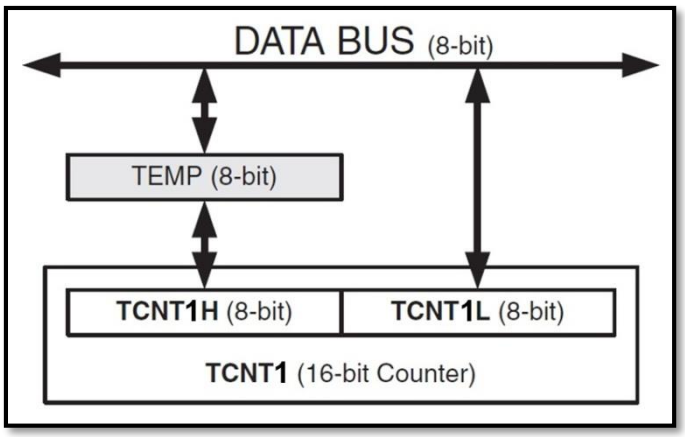
نلاحظ من الشكل السابق وجود مُقسّم تردد بعرض (10 bit) (10-bit Frequency Prescaler) يسمح بتقسيم إشارة التردد الواردة من الكريستالة الخارجيّة إلى إشارات ذات ترددات أصغر ليتم إدخالها إلى كيان المؤقت/العداد (1) فيقوم بعدّ نبضاتها في حال كونه يعمل بنمط "المؤقت" .

يمكن لكيان المؤقت/العداد (1) وفقاً لمصدر نبضات الساعة (CLK_{T1}) الداخلة إليه أن يكون ضمن أحد الأنماط الثلاثة المُبيّنة في الجدول التالي :

Work Mode	Clock Source	CLK_{T1}			
		CLK= 1 MHz	CLK= 2 MHz	CLK= 4 MHz	CLK= 8 MHz
Stop Mode	0	---	---	---	---
Timer Mode	CLK	1 MHz	2 MHz	4 MHz	8 MHz
	CLK/8	125 KHz	250 KHz	500 KHz	1 MHz
	CLK/64	15625 Hz	31250 Hz	62500 Hz	125 KHz
	CLK/256	3906.25 Hz	7812.5 Hz	15625 Hz	31250 Hz
	CLK/1024	976.56 Hz	1953.1 Hz	3906.25 Hz	7812.5 Hz
Counter Mode	T1	---	---	---	---
	T1	---	---	---	---

التعامل مع مسجلات بعرض (16-bits) بواسطة ممر معطيات بعرض (8-bits) :

يمتلك كيان المؤقت/العداد (1) عدة مسجلات بعرض (16-bits) وبالتالي هو بحاجة إلى آليّة خاصة للتعامل معها (القراءة منها والكتابة عليها) . حيث أنّ عرض ممر المعطيات في متحكمات (AVR) هو (8-bits) فهو غير قادر أن يُمرر (16) بت من المعطيات دفعةً واحدة . ولتحقيق هذا الغرض يقوم المتحكم بكتابة بيانات البايت العلوي ضمن مسجل مُؤقّت (TEMP Register) ومن ثم و عند كتابة بيانات البايت السفلي من المسجل يقوم بنقل محتويات المسجل المؤقّت إلى البايت العلوي لتتم الكتابة على المسجل ذو (16) بت في نفس اللحظة الزمنية . والشكل المجاور يُبيّن طريقة اتصال المسجل (TCNT1) مع ممر المعطيات في المتحكم. حيث يتصل البايت السفلي منه مع الممر بشكل مباشر. في حين أن البايت العلوي منه يتصل مع الممر عبر مسجل وسيط (TEMP) (المسجل المؤقت).



يقودنا ذلك إلى أنه في أي عملية كتابة على أحد مسجلات المؤقت/العداد (1) ذات (16-bits) يجب أن نبدأ بكتابة معطيات البايت العلوي من المسجل ومن ثم نكتب معطيات البايت السفلي منه . أما عملية القراءة من أحد مسجلات المؤقت/العداد (1) ذات (16-bits) فتتم بالعكس حيث نقرأ في البداية محتويات البايت السفلي من المسجل ثم محتويات البايت العلوي منه.

مسجلات التحكم الخاصة بالمؤقت/العدّاد (1) :

مسجل المؤقت/العداد (1) (TCNT1) Timer/Counter (1) Register

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
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	

يحتوي هذا المسجل على القيمة الآنيّة للعد حيث تزداد قيمته (أو تنقص) بمقدار واحد عند كل نبضة ترد على كيان المؤقت / العداد (1) . وهو مسجل بعرض (16-bits) تتراوح قيمته بين (0 → 65535) عشرياً أو بين (0000 → FFFF) ست عشرياً . وهو مؤلّف من اجتماع المسجلين (TCNT1H --- TCNT1L).
وتُعطي متحكمات (AVR) إمكانية الوصول المباشر لهذا المسجل (سواءاً بالقراءة أو بالكتابة) حيث يمكن تهيئة المؤقت/العداد (1) بقيمة ابتدائيّة ما عبر كتابة هذه القيمة ضمن هذا المسجل في بداية عملية العد. مع مراعاة طريقة الكتابة عليه وطريقة القراءة منه على اعتباره مسجلاً ذو (16-bits) .

مسجل نظير المقارنة (A) (OCR1A) Output Compare Register (1) A

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
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	

يُستخدم هذا المُسجّل في حالة تشغيل المؤقت/العداد (1) وفق نمط "مقاطعة تطابق نظير المقارنة (A)". وهو مؤلف من اجتماع المسجلين (OCR1AH --- OCR1AL) . حيث يقوم المبرمج في هذا النمط بوضع قيمة ما في هذا المسجل (تتراوح ما بين (0 → 65535)) . ويقوم كيان المؤقت/العداد (1) عند كل نبضة بمقارنة القيمة الآنية للعد (المسجل (TCNT1)) مع قيمة المسجل (OCR1A) وعند تطابق القيمتين يذهب إلى مقاطعة نظير المقارنة (A) (Output Compare Interrupt A). كما ويمكن أيضاً تخريج قيمة منطقيّة معينة على القطب (OC1A) عند حدوث التطابق بين المسجلين .
يُستخدم هذا المسجل أيضاً عند تشغيل المؤقت/العداد (1) في نمط (PWM) حيث يتم وضع القيمة التي تُعبّر عن عرض النبضة المُراد التحكم بها داخل هذا المسجل . ويتم تخريج النبضات المطلوبة في هذا النمط عبر القطب (OC1A) . و باعتباره مسجل بعرض (16-bits) أيضاً يجب مراعاة طريقة الكتابة عليه و القراءة منه .

مسجل نظير المقارنة (B) (OCR1B) Output Compare Register (1) B

Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
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	

لا يختلف هذا المسجل عن سابقه في شيء . حيث أنه يُستخدم في حالة تشغيل المؤقت/العداد (1) وفق نمط "مقاطعة تطابق نظير المقارنة (B)". وهو مؤلف من اجتماع المسجلين (OCR1BH --- OCR1BL) . حيث يقوم المبرمج في هذا النمط بوضع قيمة ما في هذا المسجل (تتراوح ما بين (0 → 65535)) . ويقوم كيان المؤقت/العداد (1) عند كل نبضة بمقارنة القيمة الآنية للعد (المسجل (TCNT1)) مع قيمة المسجل (OCR1B) وعند تطابق القيمتين يذهب إلى مقاطعة نظير المقارنة (B) (Output Compare Interrupt B). كما ويمكن أيضاً تخريج قيمة منطقيّة معينة على القطب (OC1B) عند حدوث التطابق بين المسجلين .

يُستخدم هذا المسجل أيضاً عند تشغيل المؤقت/العداد (1) في نمط (PWM) حيث يتم وضع القيمة التي تُعبّر عن عرض النبضة المُراد التحكم بها داخل هذا المسجل . ويتم تخريج النبضات المطلوبة في هذا النمط عبر القطب (OC1B) . و باعتباره مسجل بعرض (16-bits) أيضاً يجب مراعاة طريقة الكتابة عليه و القراءة منه .

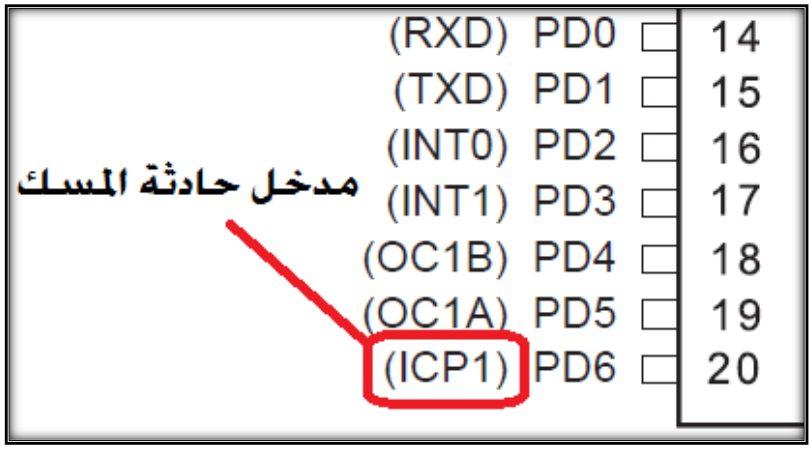
يُزوّدنا المؤقت/العداد (1) بإمكانية استخدام مسجلي نظير المقارنة (A,B) مع بعضهما البعض في نمط عمل واحد للمؤقت/العداد (1) . حيث يمكن وضع قيمة ما داخل المسجل (OCR1A) وقيمة أخرى داخل المسجل (OCR1B). ولدى وصول المؤقت/العداد (1) في العد إلى القيمة الموجودة ضمن المسجل (OCR1A) يذهب إلى مقاطعة نظير المقارنة (A) ويتابع العد حتى وصوله إلى القيمة الموجودة ضمن المسجل (OCR1B) ليذهب إلى مقاطعة نظير المقارنة (B) .

مسجل حادثة المسك (ICR1) Input Capture Register (1)

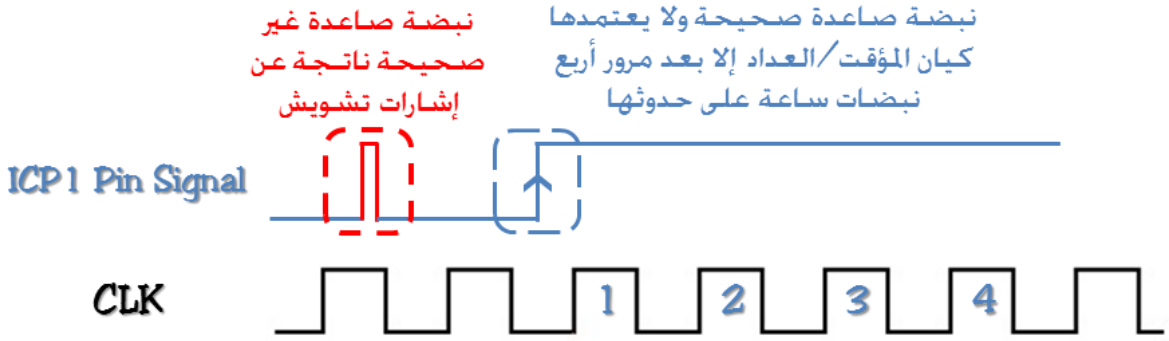
Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

يُستخدم هذا المُسجل في حال أراد المبرمج الاستفادة من ميزة حادثة المسك التي زُوّد بها المؤقت / العداد (1) دون غيره من المؤقتات . وهو مسجل بعرض (16-bits) مُؤلّف من اجتماع مسجلين (ICR1H --- ICR1L). و يجب مراعاة طريقة الكتابة عليه و القراءة منه عند التعامل معه .

ترتبط حادثة المسك (Input Capture Event) بقطب المتحكم الخارجي (ICP1) . وعند تفعيل هذه الميزة في عمل المؤقت/العدّاد (1) يبدأ بالعدّ وفق تقسيمة تردد معينة (يعمل كمؤقت) مُهيّئة مُسبقاً , وينتظر في هذا النمط من العمل ورود جبهة المسك على القطب الخارجي (ICP1) (جبهة صاعدة أو جبهة هابطة) . وعند ورودها على هذا القطب يقوم كيان المؤقت/العدّاد (1) مباشرةً بأخذ نسخة عن قيمة مسجل المؤقت/العدّاد (1) (TCNT1) ويضعها في مسجل حادثة المسك (ICR1) ثم يذهب إلى برنامج خدمة مقاطعة حادثة المسك (Input Capture Interrupt) إذا كانت هذه المقاطعة مُفعّلة . و يستمر كيان المؤقت/العدّاد (1) بالعدّ أو يُصفّر المسجل (TCNT1) بحسب تهيئة المُبرمج المُسبقة له .



زُوّدت متحكمات (AVR) المؤقت/العدّاد (1) بإمكانية تفعيل ميزة مانع الضجيج (Noise Canceler) على قطب حادثة المسك (ICP1) . في هذه الميزة ينتظر المتحكم بمقدار أربع نبضات ساعة حتى يقرر تحقق جبهة النبضة المُسبّبة لحادثة المسك من عدمه . وتهدف هذه الميزة إلى إلغاء حالات الجبهات الكاذبة الممكن حدوثها نتيجة إشارات الضجيج والتشويش على قطب مدخل حادثة المسك (ICP1) .



آليّة عمل مانع الضجيج (Noise Canceler)

مسجل التحكم A بالمؤقت / العدّاد (1) (TCCR1A)

Timer/Counter (1) A Control Register

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

يشترك هذا المسجل مع المسجل (TCCR1B) بالتحكم بعمل وسلوك كيان المؤقت/العداد (1). بحيث يقوم المُبرمج بكتابة بايت تحكم مناسب فيه في البداية لتحديد طريقة عمل هذا الكيان ضمن البرنامج .

البتين (COM1A1 , COM1A0) (bit7 , bit6) :

يُعتبر هذين البتين مسؤولين عن تحديد سلوك القطب (OCR1A) عند تحقق مقاطعة نظير المقارنة (A) للمؤقت / العداد (1) . وذلك وفقاً للجدول المبين في الفقرة التالية .

البتين (COM1B1 , COM1B0) (bit5 , bit4) :

يُعتبر هذين البتين مسؤولين عن تحديد سلوك القطب (OCR1B) عند تحقق مقاطعة نظير المقارنة (B) للمؤقت / العداد (1) . وذلك وفقاً للجدول المبين التالي :

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on compare match
1	0	Clear OC1A/OC1B on compare match (Set output to low level)
1	1	Set OC1A/OC1B on compare match (Set output to high level)

إن الجدول السابق خاص بأنماط عمل المؤقت/العدّاد (1) عدا أنماط عمل (PWM) . عندها تصبح للبتات السابقة وظيفة إضافيّة مُبيّنة ضمن الجدولين التاليين (الجدول الأول خاص بنمط (Fast PWM) والثاني لبقيّة أنماط (PWM)) :

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OCnA/OCnB disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at TOP
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at TOP

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OCnA on Compare Match, OCnB disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when downcounting.

البتين (FOC1A , FOC1B) (bit3 , bit2) :

يمكن مراجعة وثيقة المتحكم (ATmega16) (ATmega16 Datasheet) للحصول على معلومات عن هذين البتين .

البتين (WGM11 , WGM10) (bit1 , bit0) :

يشترك هذين البتين مع البتين (WGM13 , WGM12) من مسجل التحكم (TCCR1B) في تحديد نمط عمل المؤقت/العداد (1) وبالتالي تحديد بداية سلسلة العد ونهايتها. وذلك وفقاً للجدول التالي :

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

مسجل التحكم B بالمؤقت / العدّاد (1) (TCCR1B)

Timer/Counter (1) B Control Register

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

يشترك هذا المسجل مع المسجل (TCCR1A) بالتحكم بعمل وسلوك كيان المؤقت/العداد (1). بحيث يقوم المُبرمج بكتابة بايت تحكم مناسب فيه في البداية لتحديد طريقة عمل هذا الكيان ضمن البرنامج .

البت (ICNC1) (bit7) :

وهو البت المسؤول عن تفعيل ميزة مانع الضجيج على قطب مدخل حادثة المسك (ICP1) . عند كتابة (1) منطقي في هذا البت تُفعّل هذه الخاصيّة على القطب ICP1 ويُلغى تفعيلها عند كتابة (0) منطقي فيه .

البت (ICES1) (bit6) :

وهو البت المسؤول عن تحديد جبهة النبضة التي تؤدي إلى توليد حادثة المسك على قطب مدخل حادثة المسك (ICP1) . وذلك وفقاً للقاعدة التالية :

عند وضع القيمة (0) منطقي في هذا البت فإن حادثة المسك سوف تُقدح لدى ورود نبضة هابطة على القطب (ICP1) .

و عند وضع القيمة (1) منطقي في هذا البت فإن حادثة المسك سوف تُقدح لدى ورود نبضة صاعدة على القطب (ICP1) .

البتين (WGM13 , WGM12) (bit4 , bit3) :

يشترك هذين البتين مع البتين السابقين (WGM11 , WGM10) من مسجل التحكم السابق (TCCR1A) في تحديد نمط عمل المؤقت/العداد (1) وذلك وفقاً للجدول المُبيّن مُسبقاً .

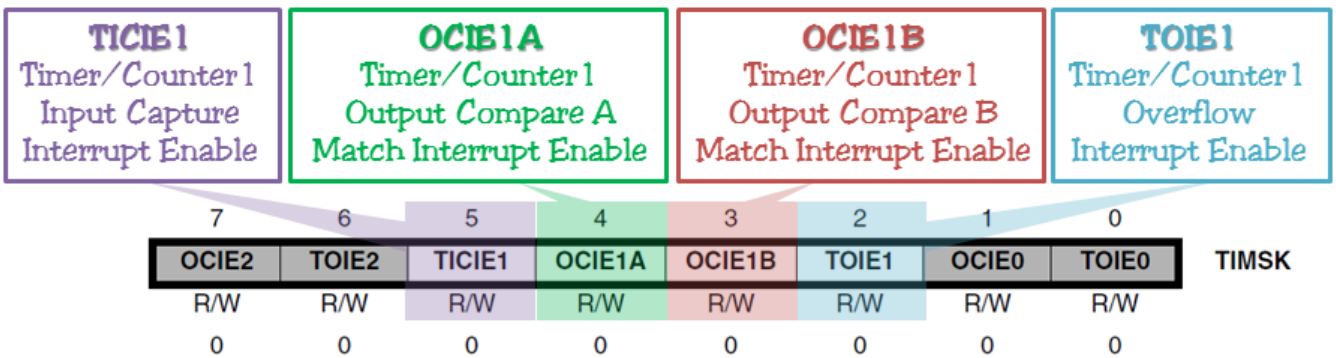
البتات (CS12 , CS11 , CS10) (bit2 , bit1 , bit0) :

تُحدد هذه البتات الثلاثة مصدر نبضات الساعة (CLK_{T1}) الذي سيقوم المؤقت/العداد (1) بعدّها وذلك وفقاً للجدول التالي :

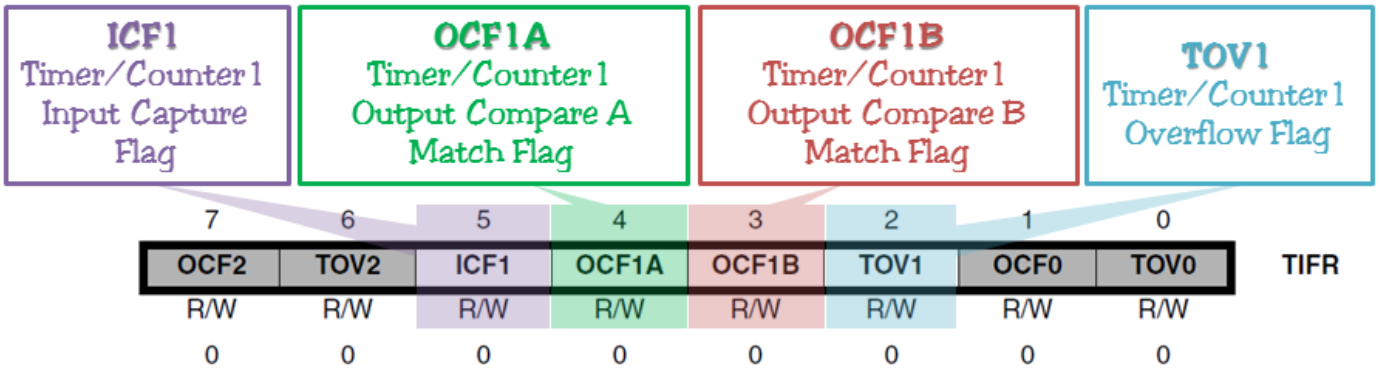
CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

وعلى اعتبار أن كيان المؤقت/العداد(1) يمتلك أربع مقاطعات (مقاطعة طفحان المؤقت/العداد(1) و مقاطعة تطابق نظير المقارنة (A) و مقاطعة تطابق نظير المقارنة (B) و مقاطعة حادثة المسك) فإن ذلك يستلزم وجود بتّين لكل مقاطعة : بت تفعيل وبت علم مقاطعة. تتواجد هذه البتات ضمن المسجلين التاليين :

مسجل قناع مقاطعة المؤقت/العداد (TIMSK)



مسجل أعلام مقاطعة المؤقت/العداد (TIRF)



أنماط عمل المؤقت/العداد (1)

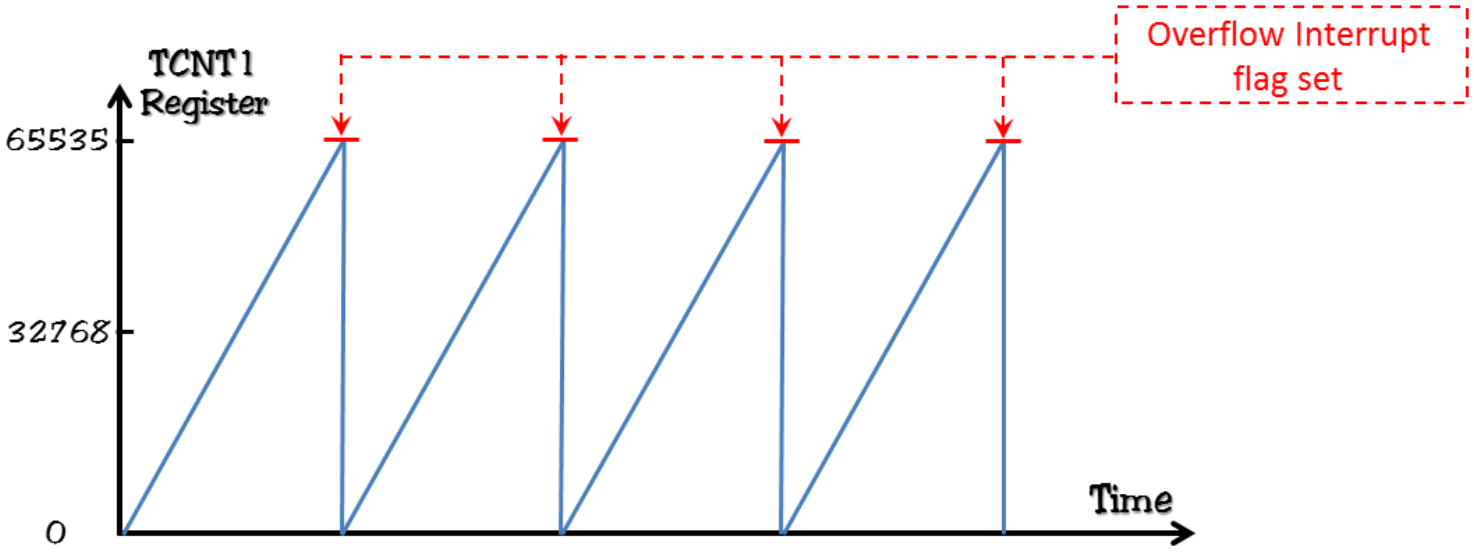
(Timer/Counter (1) Operation Modes)

أولاً : أنماط عمله كمؤقت (Timer Modes) :

نمط العمل العادي (Normal Mode) :

وهو أبسط نمط عمل لكيان المؤقت / العداد (1), حيث يتم فيه تفعيل هذا الكيان ليعمل كمؤقّت فيقوم بزيادة محتوى المسجل (TCNT1) بمقدار واحد عند كل نبضة واردة إلى كيان المؤقت / العداد (1) , وعند الوصول إلى القيمة العظمى للمسجل (TCNT1) (MAX=65535) يقوم الكيان بتصفير المسجل (TCNT1) والذهاب إلى برنامج مقاطعة طفحان المؤقت / العدّاد (1) (هذا إن كانت المقاطعة مُفعّلة) ومن ثم يتابع العد التصاعدي طالما أن كيان المؤقت / العدّاد (1) مُفعّلاً . ويُمكن تهيئة المسجل (TCNT1) بقيمة ابتدائية ليبدأ العد منها ويُفضّل أن تُكتب التعليمة التي تقوم بذلك ضمن برنامج مقاطعة طفحان المؤقت / العدّاد (1) .

ويُبيّن الشكل التالي المخطط الزمني للمؤقت (1) في نمط العمل العادي (Normal Mode) :



نمط تصفير المؤقت عند تطابق المقارنة (CTC Mode) :

(Clear Timer on Compare Match Mode)

يمكن للمؤقت/العداد (1) أن يعمل في هذا النمط وفق حالتين اثنتين :

الحالة الأولى : تفعيل مقاطعة نظير المقارنة (A) (OC1A) :

يقوم المُبرمج هنا بوضع قيمة ما تتراوح بين (0 --> 65535) وبالتمثيل الست عشري تتراوح بين ($0000 --> $FFFF) داخل مسجل نظير المقارنة (A) (OCR1A) . ومن ثم يقوم بتشغيل كيان المؤقت/العداد (1) ليعمل كمؤقت يعدّ نبضات الساعة القادمة إليه من مقسم التردد (CLK_{T1}) وعند كل عدّة يقوم بمقارنة القيمة الآنيّة لمسجل المؤقت/العداد (1) (TCNT1) مع القيمة الموجودة في مسجل نظير المقارنة (A) (OCR1A) وعند تطابق القيمتين يقوم المتحكم بتصفير قيمة مسجل المؤقت/العداد (1) (TCNT1) ومن ثم يقوم بما يلي : (تحتاج كل وظيفة إلى تفعيل على حدى)

- يذهب إلى برنامج خدمة مقاطعة "تطابق نظير المقارنة (A)" للمؤقت / العداد (1) لينفذ تعليماته ويعود بعد انتهائه إلى البرنامج الرئيسي .
- يقوم بتخريج قيمة منطقيّة مُعيّنة (مُحدّدة مسبقاً) على القطب (OC1A) .

وبالتالي يمكننا من خلال تشغيل كيان المؤقت/العداد (1) في هذا النمط توليد قطار من النبضات على القطب (OC1A) وذلك بتهيئته لعكس المنطق (Toggle) المتوضّع عليه عند حدوث مقاطعة تطابق نظير المقارنة (A) . وبذلك يتولد لدينا قطار من النبضات يُعطى ترددها بالعلاقة التالية :

$$F_{OC1A\ Pin} = \frac{F_{CLK}}{2 * N * (1 + OCR1A)}$$

حيث أنّ :

$F_{OC1A\ Pin}$: تردد إشارة النبضة المُربعة المولّدة عبر قطب نظير المقارنة (OC1A) .

F_{CLK} : تردد الهزاز الكريستالي الخارجي الموصول مع المتحكم الصغري .

N : معامل مقسّم التردد الذي يعمل عنده المؤقت/العدّاد (1) (1 or 8 or 64 or 256 or 1024) .

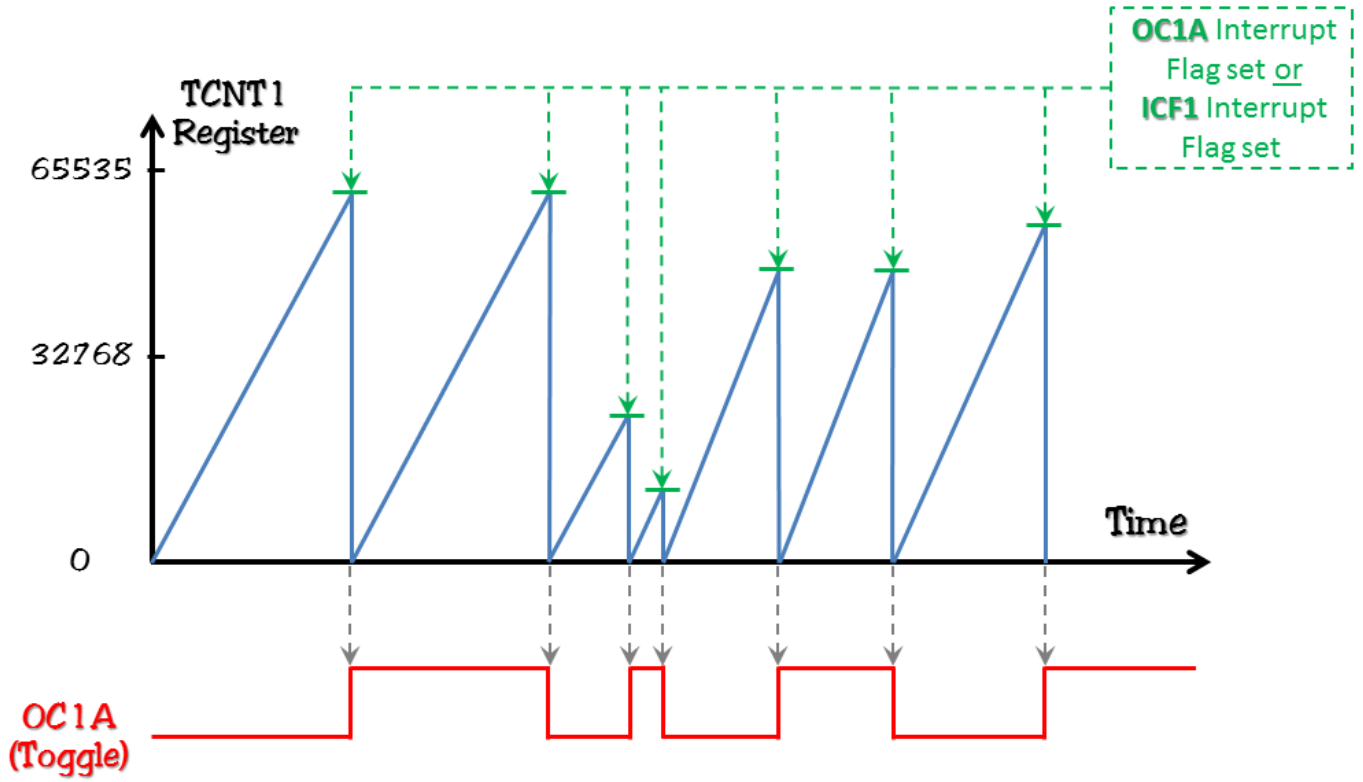
OCR1A : القيمة العشرية لمحتوى مسجل نظير المقارنة (OCR1A) .

ملاحظة : ما تمّ شرحه عن مقاطعة نظير المقارنة (A) للمؤقت/العدّاد (1) ينطبق على مقاطعة نظير المقارنة (B) باستبدال الحرف (A) بالحرف (B) فيما ورد من الشرح السابق .

الحالة الثانية : تفعيل مقاطعة حادثة المسك (ICF1) :

لاحظنا أنّ المبرمج في مقاطعة تطابق نظير المقارنة هو من يُحدد نهاية سلسلة العدّ عند المؤقت/العدّاد (1). وذلك بكتابة تلك القيمة داخل مسجل نظير المقارنة (OCR1A) . أما في حالة حادثة المسك فينتظر كيان المؤقت/العدّاد (1) حتى تحقق حادثة المسك على القطب (ICP1) عندها يقوم بنسخ قيمة المسجل (TCNT1) إلى المسجل (ICR1) ومن ثم يُصفّر قيمة المسجل (TCNT1) ليبدأ العد من جديد . وبالتالي نحن هنا لا نعلم القيمة الأعلى التي سيقف عندها العدّ حيث يتم ذلك عند ورود النبضة التي تُؤدّي إلى تحقيق حادثة المسك على قطب مدخل حادثة المسك (ICP1) .

ويُبيّن الشكل التالي المخطط الزمني للمؤقت/العداد (1) في نمط العمل (CTC Mode) :



نمط تعديل عرض النبضة السريع (Fast PWM Mode) :

(Fast Pulse Width Modulator Mode)

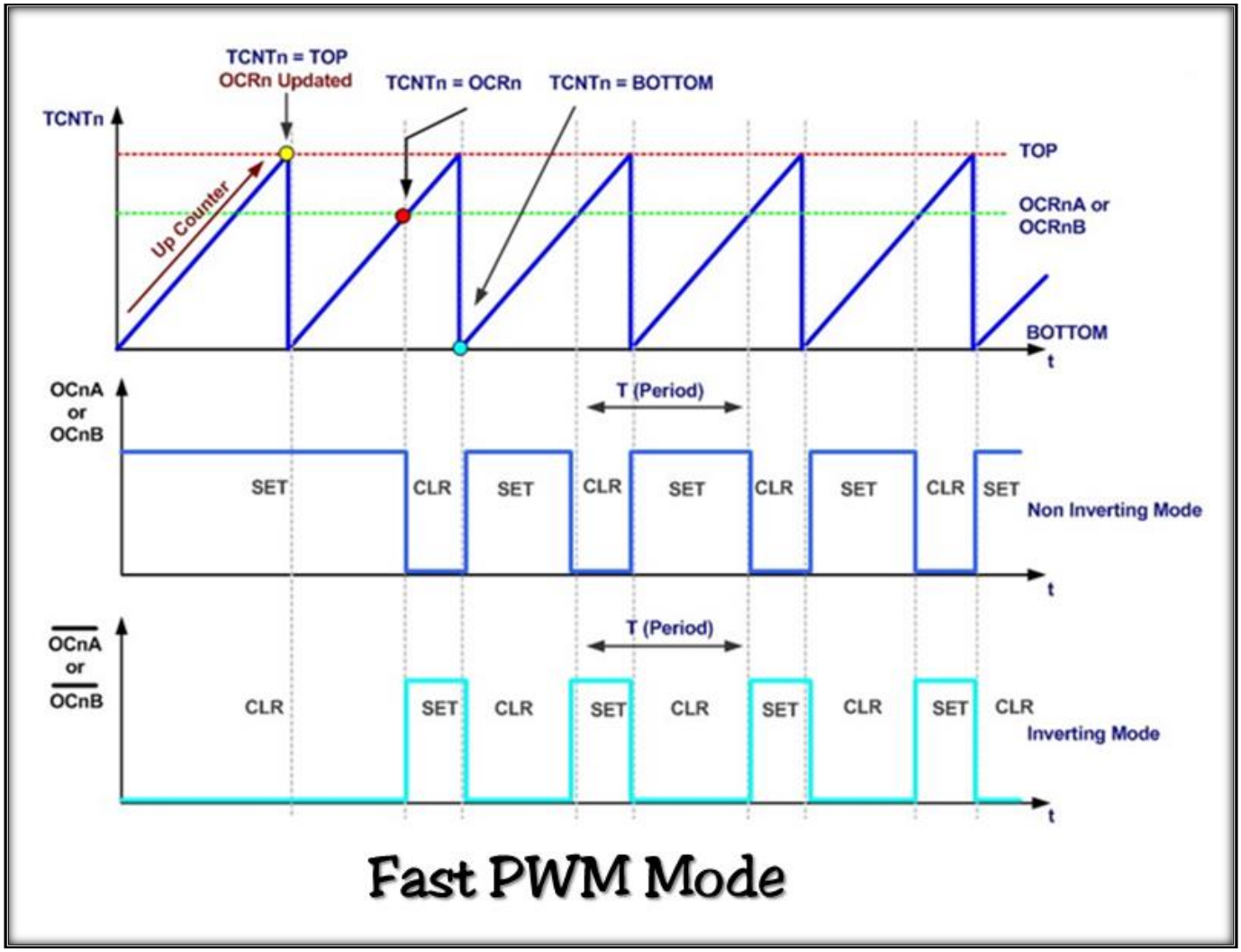
يمكن للمُتحكم الصغري (ATmega16) توليد نبضات مربعة عبر أحد القطبين (OC1A or OC1B) كما يمكن التحكم بعرض النبضة في تلك الإشارات المولّدة عبر هذين القطبين . ويتم ذلك بتشغيل المؤقت/العداد (1) في أحد أنماط (PWM) الخاصة به. ويُستفاد من ميزة تعديل عرض النبضة في توليد إشارات مختلفة التردد و التحكم بسرعة المحركات وكذلك توليد إشارات تشابهية عبر قيم رقمية ..... الخ .

يعتمد مبدأ عمل نمط تعديل النبضة (PWM) السريع في المؤقت/العداد (1) بشكل أساسي على مسجل نظير المقارنة (OCR1A) وقطب نظير المقارنة (A) (OC1A) . ويتلخص عمله في النقاط التالية :

- عند تفعيل المؤقت/العداد (1) في نمط (PWM) السريع يقوم مباشرةً بالعد تصاعديّاً من أصغر قيمة له (Bottom=$0000) وحتى أعلى قيمة له (MAX=$FFFF) و من ثم يعود للعد من القيمة الصغرى له (Bottom) .
- خلال العد التصاعدي للمؤقت/العداد (1) عندما تتحقق العلاقة (TCNT1=OCR1A) يقوم المتحكم بتخريج القيمة (0) منطقي على القطب (OC1A) في حالة نمط (Fast PWM) غير المعكوس (Non-inverting Fast PWM Mode) و تخريج (1) منطقي على القطب (OC1A) في حالة نمط (Fast PWM) المعكوس (Inverting Fast PWM Mode) .
- عند وصول المؤقت/العداد (1) إلى القيمة الصُغرى له (TCNT1=Bottom) يقوم المتحكم بتخريج القيمة (1) منطقي على القطب (OC1A) في حالة نمط (Fast PWM) غير المعكوس

(Non-inverting Fast PWM Mode) و تخريج (0) منطقي على القطب (OC1A) في حالة نمط (Fast PWM) المعكوس (Inverting Fast PWM Mode) (قلب الحالة السابقة للقطب (OC1A))

❖ إذا أراد المُبرمج تحديث قيمة مسجل نظير المقارنة (OCR1A) (تغيير عرض نبضة الإشارة المولّدة) فإنه يقوم بذلك عند وصول المؤقت/العداد (1) إلى القيمة العُليا من العد (TCNT1=TOP)، ويُفضّل أن تُكتب تعليمة إسناد القيمة الجديدة إلى مسجل نظير المقارنة (OCR1A) ضمن برنامج مقاطعة طفحان المؤقت/العداد (1).



❖ عند تساوي قيمة مسجل المؤقت/العدّاد (1) (TCNT1) مع قيمة مسجل نظير المقارنة (OCR1A) تتولّد إشارة مقاطعة نظير المقارنة (A) ويذهب المتحكم لتنفيذ برنامج مقاطعة نظير المقارنة (A) إذا كان ذلك مُفعّلاً. وكذلك عند وصول قيمة مسجل المؤقت/العدّاد (1) (TCNT1) إلى القيمة العُظمى من العد (TCNT1=$FFFF) تتولّد إشارة مقاطعة طفحان المؤقت/العدّاد (1) ويذهب المُتحكّم لتنفيذ برنامج مقاطعة طفحان المؤقت/العدّاد (1) إذا كان ذلك مُفعّلاً أيضاً.

❖ يُعطى تردد الإشارة المُربّعة المُولّدة عبر القطب (OC1A) في نمط (Fast PWM) بالعلاقة التالية:

$$F_{\substack{OC1A\ Pin \\ Fast\ PWM\ Mode}} = \frac{F_{CLK}}{N * 65536}$$

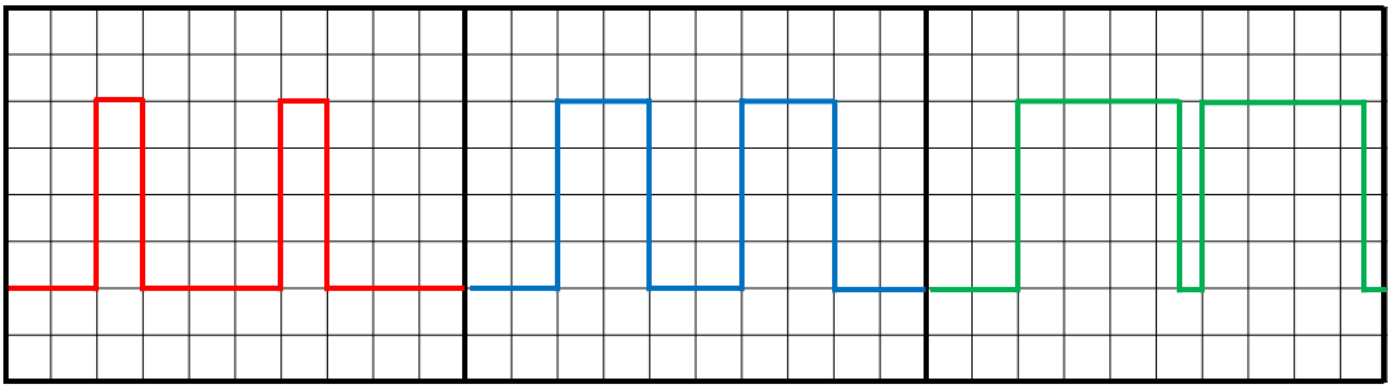
حيث أنّ :

$F_{OC1A\ Pin}$: تردد إشارة النبضة المُربّعة المولّدة عبر قطب نظير المقارنة (OC1A) .

F_{CLK} : تردد الهزاز الكريستالي الخارجي الموصول مع المتحكم الصغري .

N : معامل مقسّم التردد الذي يعمل عنده المؤقت/العدّاد (1) (1 or 8 or 64 or 256 or 1024) .

الجدير بالذكر هنا أنّ تردد الإشارة المُولّدة لا يتعلق بقيمة مسجل نظير المقارنة (OCR1A) حيث أن تغيّر هذه القيمة يُغيّر في عرض النبضة الموجبة ضمن الإشارة المربعة ولا يُغيّر ترددها . ويُطلق على نسبة النبضة الموجبة في الدور الواحد ضمن الإشارة المُربّعة مصطلح (Duty Cycle) .



Duty Cycle : 25% Duty Cycle : 50% Duty Cycle : 88%

❖ يُتيح المتحكم (ATmega16) تشغيل المؤقت/العدّاد (1) وفق نمط (Fast PWM) بعرض مسجل (8-bits or 9-bits or 10-bits) بالإضافة طبعاً لإمكانية تشغيله بكامل عرض مسجل المؤقت/العدّاد (1) وهو (16-bits) والاختلاف هنا يكون في قيمة نهاية سلسلة العد في كل نمط (TOP Value) . كما ويمكن تشغيله وفق هذا النمط بحيث تكون نهاية سلسلة العد مساوية لمسجل حادثة المسك (Top = ICR1) . نُبيّن في الجدول التالي أنماط عمل المؤقت/العدّاد (1) ضمن النمط (Fast PWM) (خمسة أنماط) :

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1X	TOV1 Flag Set on
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

نمط تعديل عرض النبضة بتصحيح الطور (Phase Correct PWM Mode) :

(Phase Correct Pulse Width Modulator Mode)

هناك أوجه تشابه كثيرة بين الأنماط المختلفة لتعديل عرض النبضة (PWM) في المؤقت/العدّاد (1) . فالغاية الرئيسية من جميع تلك الأنماط هي توليد إشارة مربعة على القطب (OC1A) مع إمكانية تعديل عرض النبضة الموجبة فيها وبشكل رقمي حيث يتغير عرض النبضة بتغيير القيمة الموضوعة في مسجل نظير المقارنة (OCR1A) .

يمكن شرح عمل هذا النمط (Phase Correct PWM Mode) من خلال النقاط التالية :

- يعتمد هذا النمط كما في النمط الأول على تشغيل المؤقت/العدّاد (1) ليعمل كمؤقت يعد النبضات الداخلية القادمة من مقسم التردد الخاص بذاك الكيان . إلا أنّه وبخلاف النمط الأول من (PWM) يعدّ النبضات على مرحلتين : مرحلة عد تصاعدي ومرحلة عد تنازلي ,