برمجة المتحكمات المصغرة

التجارب العملية

الملحق

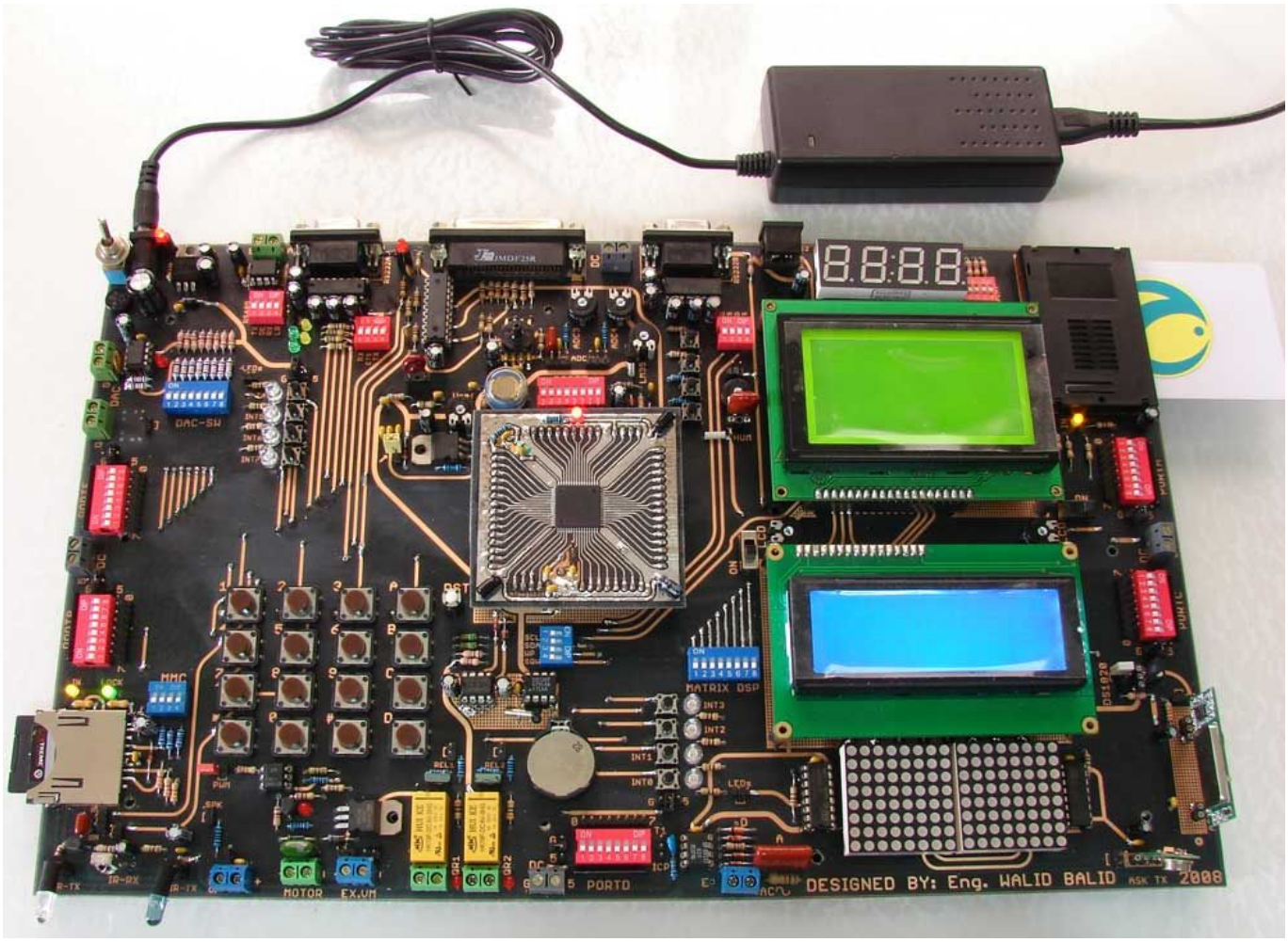
Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

WALID BALID, Tuesday, December 15, 2009

Open-Source multipurpose interactive embedded systems Microcontroller kit for laboratory education



WALID BALID

Faculty of EEE, University of Aleppo

Aleppo – Syria

2009

Acknowledgment :

I would thanks Mr. Mark ALBERTS the programmer of BASCOM-AVR, this kit has been designed and tested with BASCOM.

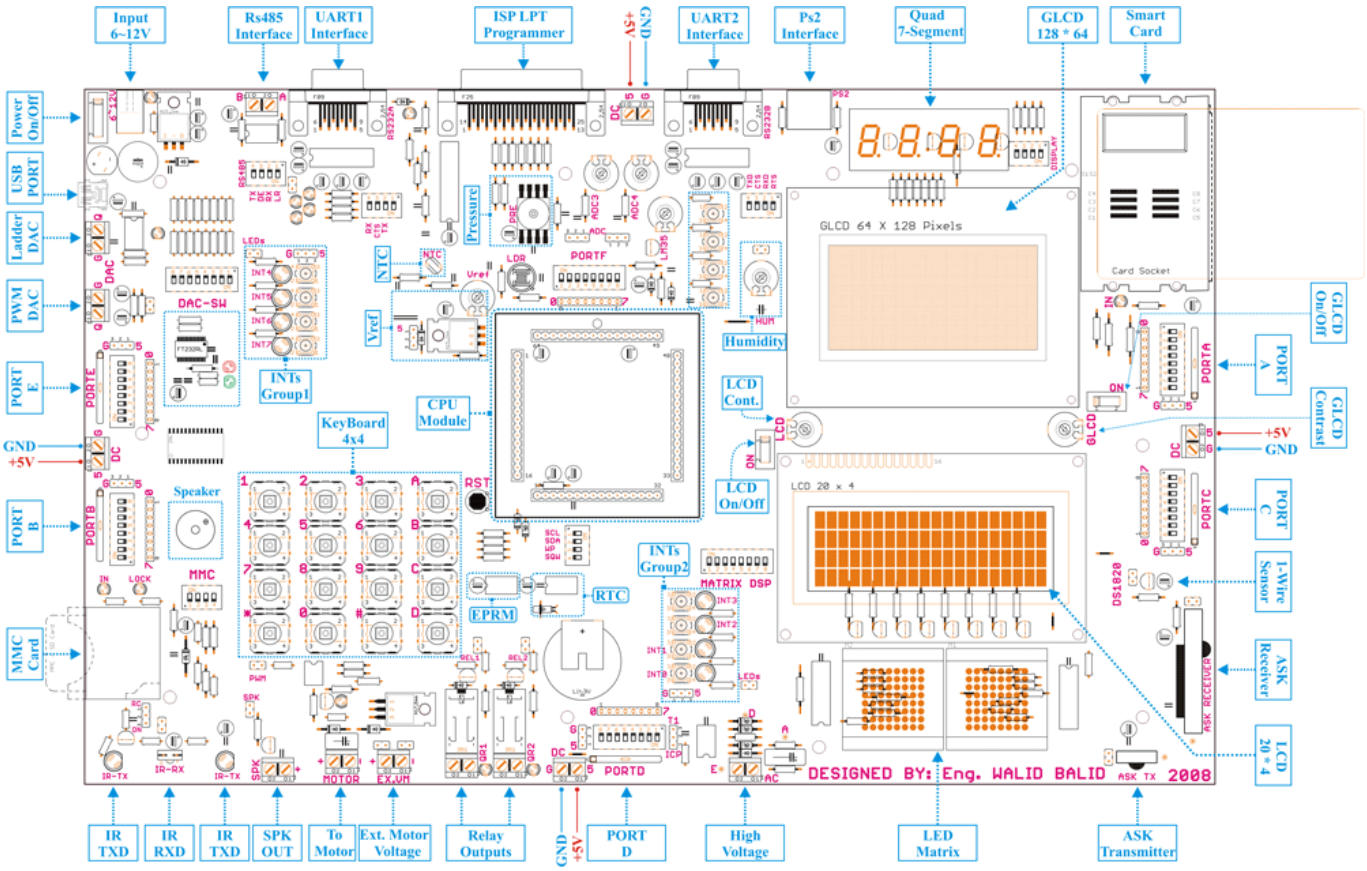
Introduction:

The kit development aimed at designing a universal training board that can cover a wide range of experiments for the electronics, communication, control, and power departments of the electrical and electronics engineering.

The kit has followed in the designing process all standards, so it can utilize any AVR MCU or any SPI protocol compatible one. The MCU programming can be done by all GNU based compilers with both low-level (ASM) and high-level (Basic, Pascal, C, or C++.) MCUs programming languages.

Many peripherals have been placed on the board to enrich its universality. About 70 different experiments (on basic, intermediate, and advanced level) have been developed.

There has been 47 peripherals unit selected and the units were placed on the board in a way that saves a comfortable working space and separates the elements of each unit from the other units.



Since a good planned student-centric approach has proven effectiveness in engaging students, the experiments manual was designed in a student-centred manner, so that the students can perform proceed with the experiment and develop the aimed skills without a need of teacher supervision.

A detailed description including schematics, experiments, and How to Build recipe have been made available for the students as an open source to enable them to build their own board, and give them even more deeper centric role in this experiential part of engineering.

It is planned to put these resources on the web making the prototype available to implement for any practicing engineering student. It is hoped that the open source approach will attract academic staff and/or students to develop more experiments based on the rich peripheral components of the kit.

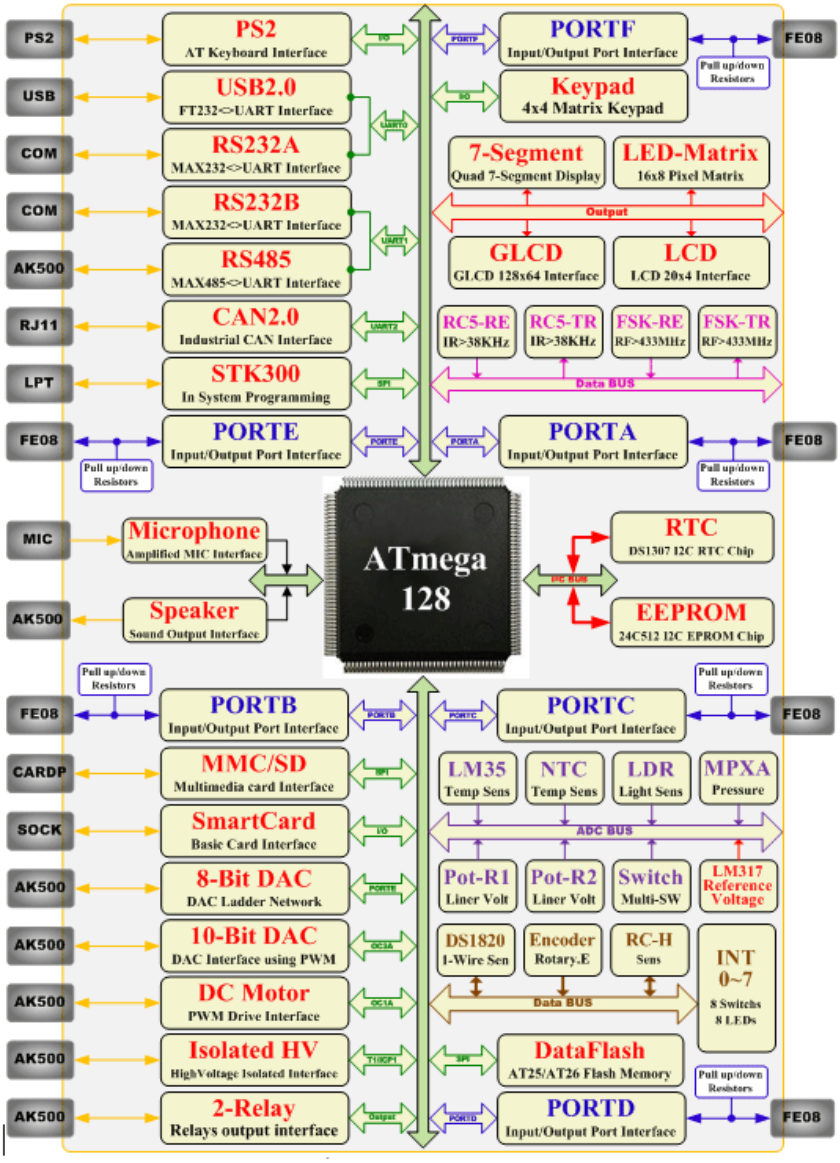
The average cost of building our board is about 120$, while any similar commercial kit will cost more than 700$.

Selection of the designed experiments has been taught in Spring 2008/2009 for about 65 students in the Electrical Engineering faculty at Aleppo University.

So, here we go. I will put the whole system between your hands.

The Kit Block Diagram:

The MCU ATmega128 is the kit core and it is linked with the kit peripherals such as a Liquid Crystal Display LCD through I/O ports. It is linked with the kit serial communication units, e.g. CAN, RS232, USB, RS485, PS2, I2C, and 1-wire through the serial interfaces. It is linked with the analog sensors, e.g. temperature or pressure, through Analog to Digital (AD) convertors. It utilizes the external interrupts I/O ports to interface with the switch buttons and the rotary encoder. The serial port I2C is connected with the data memory EEPROM (AT24C512 chip), and the real time chip RTC (DS1307 chip).

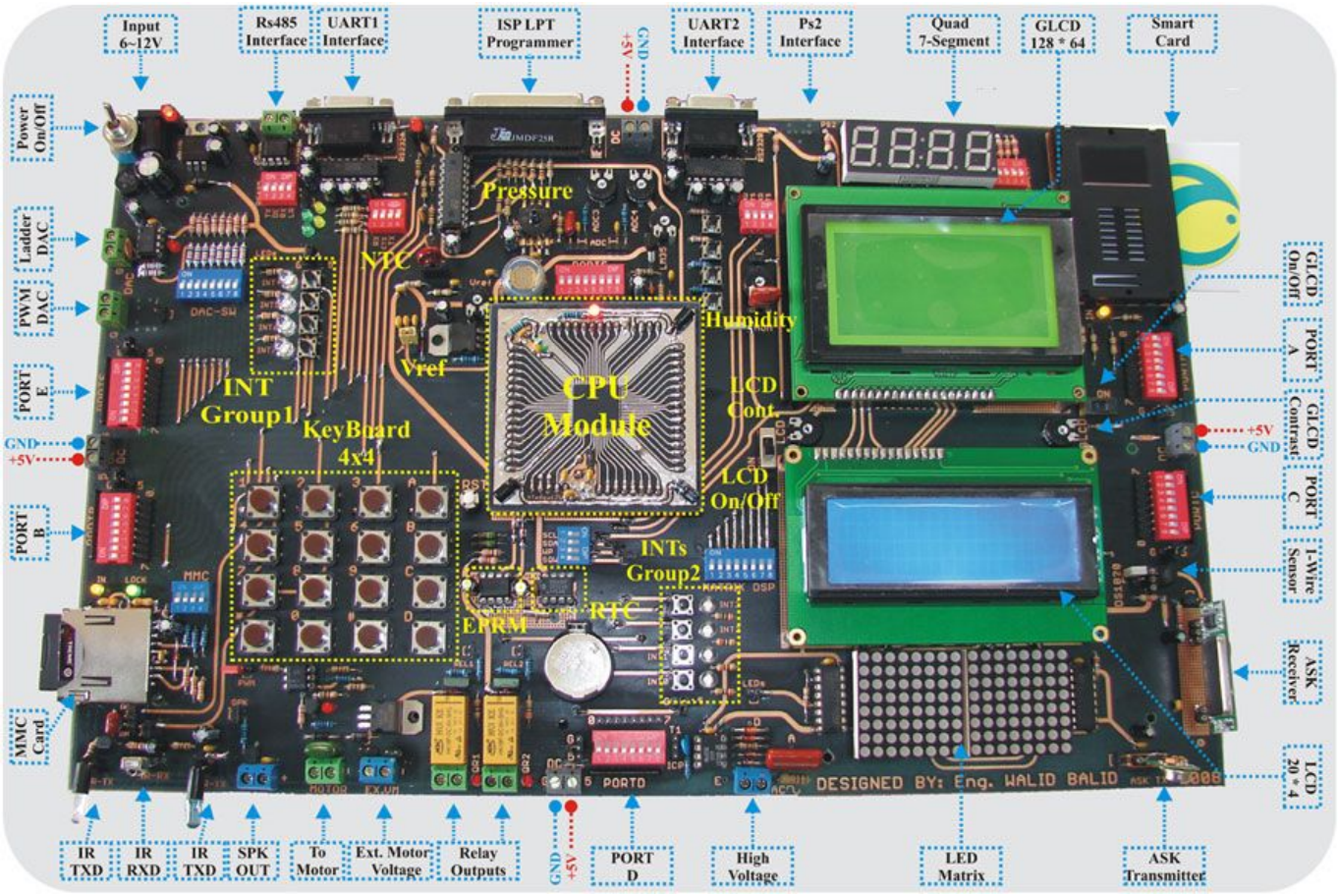


The serial port SPI is connected with the programming unit (STK300) of the MCU and the voice chip APR6016, it is also linked with an MMC card and a data flash memory (AT25xx). The serial port UART0 is connected with USB interface (FT232R chip) and with COM port through RS232 protocol for facilitating PC connection. The serial port UART2 is connected with a CAN port for facilitating industrial communication purposes. The Figure below shows the kit block diagram. All peripherals that needs external communication with the surrounding world are linked to the grey boxes.

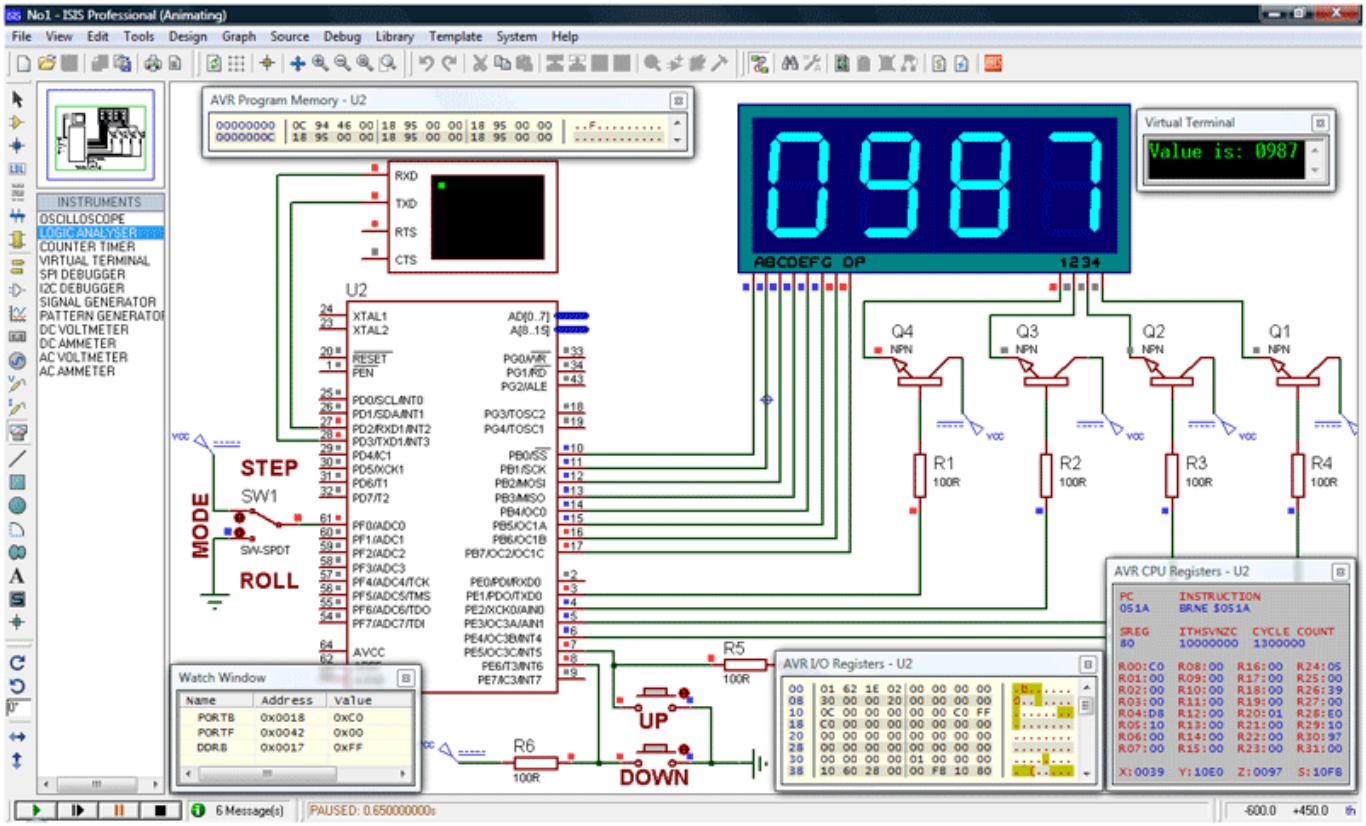
Hands-on Experiments that can be done with the kit:

1. Programming the MCU ports for displaying LEDs light movements
2. Interfacing switches with the MCU ports
3. Interfacing 4x4 Hexadecimal Array Keypad (16key) with the MCU
4. Interfacing and programming the MCU with a 20x4 LCD
5. Interfacing and programming the MCU with a 128x4 GLCD
6. Interfacing and scanning Quad seven-segment display
7. IR remote control sender/receiver based on RC5 code
8. Data Transfer using RF Transmitter based on FSK modulation
9. Interfacing and programming the MCU with Real-time clock chip
10. DAC by Interfacing 8-bit ladder network with the MCU
11. Programming the MCU analog comparator unit
12. Digital Frequency counter/meter 1HZ – 4MHz
13. Measuring the luminous intensity (Flux) using LDR
14. Measuring the barometric pressure and altitude using a Barometer
15. Interfacing with the LM35DZ analog temperature sensor
16. Storing Data using MMC/SD card in FAT23 format
17. Programming Smart-Card with high-security software algorithms
18. Interfacing and Programming 32x8pixel LED-Matrix scrolling Display
19. Speed control of DC motor using PWM
20. Interfacing the MCU with PC using RS232 protocol
21. wide area data transfer using the industrial CAN protocol
22. Interfacing with RS485 for wide area data transfer Digital Scientific calculator by Interfacing LCD (liquid crystal display) and Hexadecimal Keypad with AVR MCU.
23. Adjustable Signal Generator (Sin, Cos, Smooth, Triangular, square) by Interfacing GLCD (Graphical liquid crystal display), button Keys, and variable resistor with AVR MCU.
24. RC5 code based, IR (Infrared) remote control sender/receiver, by interfacing IR receiver module unit and IR transmitter diode with AVR MCU.
25. Obstacle detection using IR transmitter.
26. Interfacing Barometer with AVR MCU for measuring barometric pressure and altitude.
27. Interfacing RTC (Real Time Clock) chip (DS1307) for real time application.
28. Speed control of DC motor using PWM (Pulse width modulation)
29. Interfacing 8-bit ladder network with AVR MCU for DAC purpose.
30. Interfacing RS485 converter with UART serial interface for long area data transfer
31. Interfacing AVR MCU with PC using RS232 converter and USART serial interface
32. Digital Frequency counter/meter 1HZ – 4MHz
33. Programming Smart-Card with high-security software algorithms using the AES and DES symmetric-key algorithm
34. Storing Data using MMC/SD card in FAT23 format
35. Resistance and Capacitance Digital Meter
36. Wireless data transfer using IR 38KHz (Infrared) based on Ir-Data Protocol
37. Wireless data transfer using RF 433MHz (Radio Frequency) based on FSK modulation
38. wide area data transfer using the industrial CAN protocol for

39. Digital to Analog conversion using 8-bit Ladder network
40. Programming 32x8pixel LED-Matrix scrolling Display
41. Interfacing with LM35 analog temperature sensor (-45ºC ~ +125ºC)



Also, some of the experiments were provided with Simulation using Proteus.



جامعة حلب

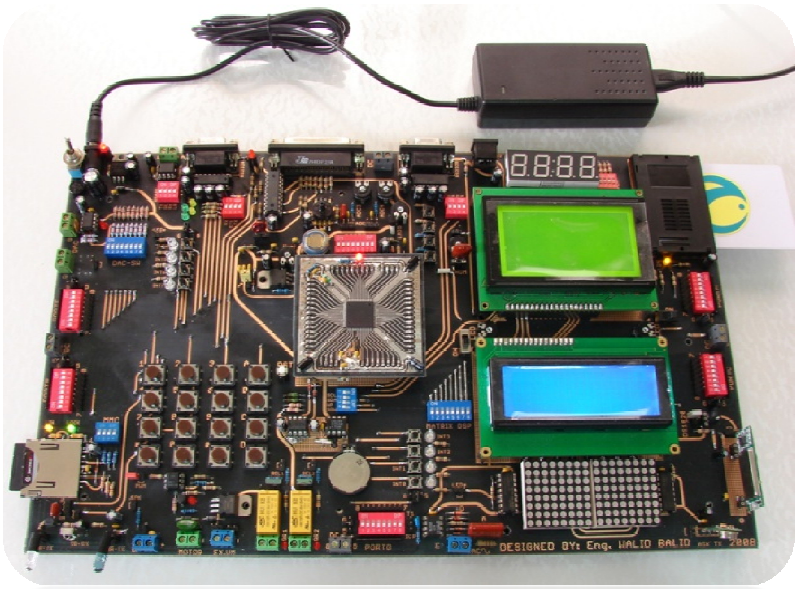
كلية الهندسة الكهربائية والإلكترونية

قسم هندسة التحكم الآلي والأتمتة

مشروع السنة الأولى ماجستير هندسة التحكم الآلي والأتمتة

دراسة وتصميم نظام تعليمي متكامل يعتمد المنهجيات العملية التفاعلية ويتضمن تصميم لوحة تطوير ذاتي تفاعلية وإعداد التجارب العملية لتعليم طلاب الكليات الهندسية برمجة نظم المتحكمات المصغرة

Designing A Novel Interactive Microcontroller Training Kit for Teaching Undergraduates Programming Embedded System Microcontroller Using Interaction Methodologies



BASCOM-AVR IDE
MCS Electronics

دراسة وتصميم: م. وليد بليد

WALID BALID, 2009

walidbalid81@gmail.com

برمجة المتحكمات المصغرة

التجارب العملية

الجلسة الأولى

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

Tuesday, December 15, 2009

مقدمة عامة عن المنهج: General Introduction about this course

في الوقت الحاضر أصبحت المتحكمات الرقمية هي القلب النابض في أنظمة التحكم وكذا في التجهيزات الكهربائية والإلكترونية، وازدادت النظم تعقيداً في بنيتها بسب تعقيد الوظائف المطلوبة من هذه النظم، وبقدر ما تزداد الوظائف المطلوبة من النظام يزداد تعقيد النظام وبالتالي تعقيد الكود البرمجي للمتحكم الذي يقود هذا النظام.

لقد بات من الصعب جداً بل من المستحيل برمجة نظم التحكم الرقمي المتقدمة بلغة التجميع (Assembly)، وأصبح تسارع الوقت وحاجة السوق وعامل الزمن سبباً أساسياً لابتكار لغات برمجية عالية المستوى لبرمجة المتحكمات الرقمية.

في هذا المنهاج، سوف نعالج برمجة المتحكمات المصغرة من العائلة AVR باستخدام لغة عالية المستوى تقارب لغة Basic من حيث تكوينها وشكل تعليماتها، والتي تتم في بيئة البرنامج Bascom-AVR.

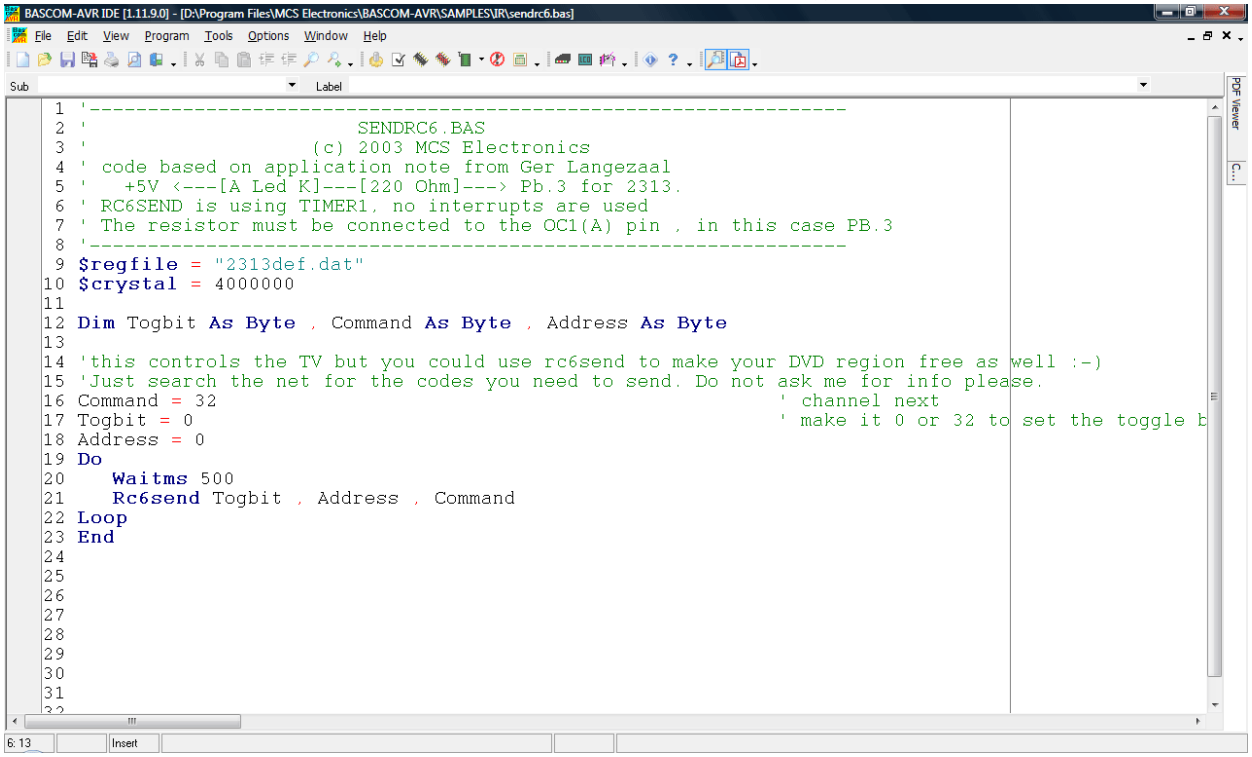
كذلك سنقوم بمحاكاة جميع الأمثلة والتطبيقات في بيئة البرنامج Proteus الذي يعد من أقوى البرامج التي تحاكي عمل المعالجات.

سنقوم بعدها بتنفيذ جميع التجارب عملياً على لوحة التطوير (Development Board) المخصصة والتي تم تصميمها خصيصاً لأغراض تعليمية بحيث ترتقي بالمتعلم من مستوى مبتدئ إلى مستوى متقدم وهي تحوي على أكثر من 50 تجربة تشمل جميع الوظائف الأساسية للمتحكمات بالإضافة إلى وظائف متقدمة أخرى.

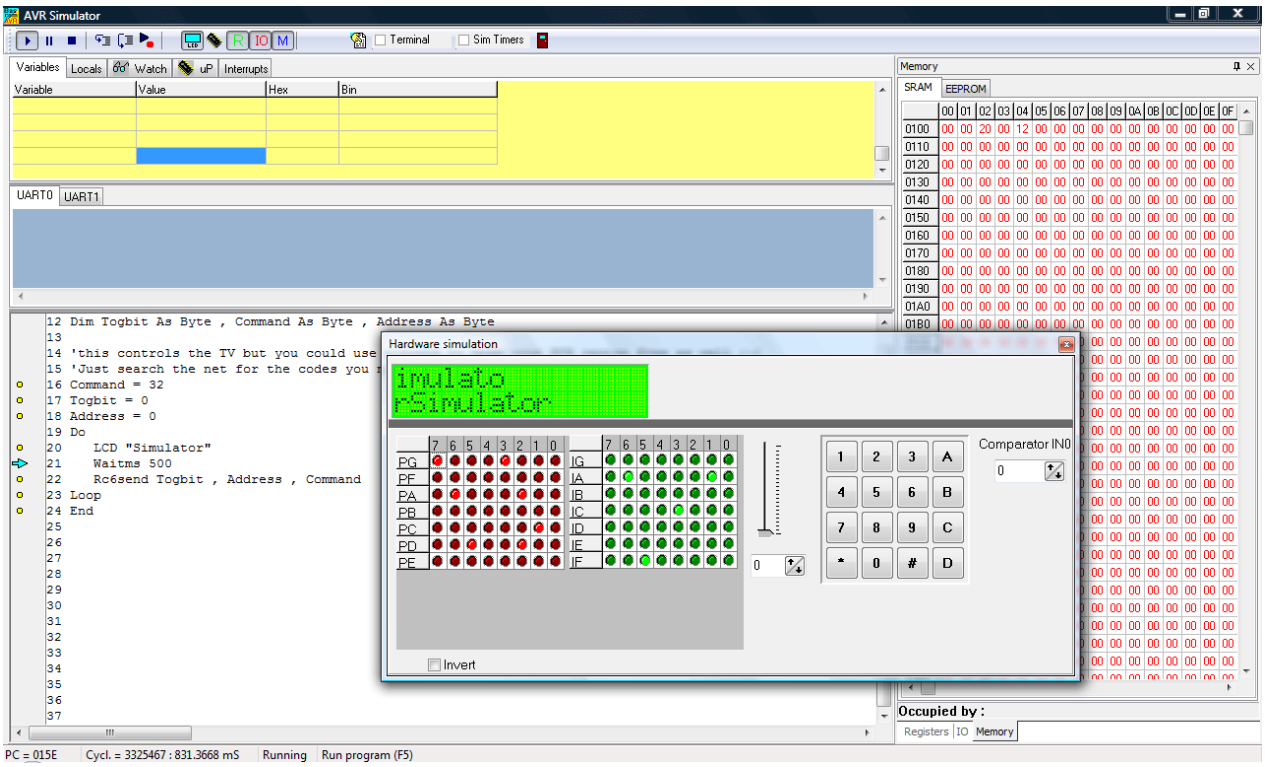
Bascom-AVR Compiler البيئة البرمجية Bascom-AVR:

كما أسلفنا بالذكر أننا سوف نستخدم البرنامج Bascom-AVR في برمجة جميع التجارب وذلك لما يوفره هذا البرنامج من بيئة برمجية قوية بالإضافة إلى المكتبات الأساسية الشاملة، ويحوي البرنامج على:

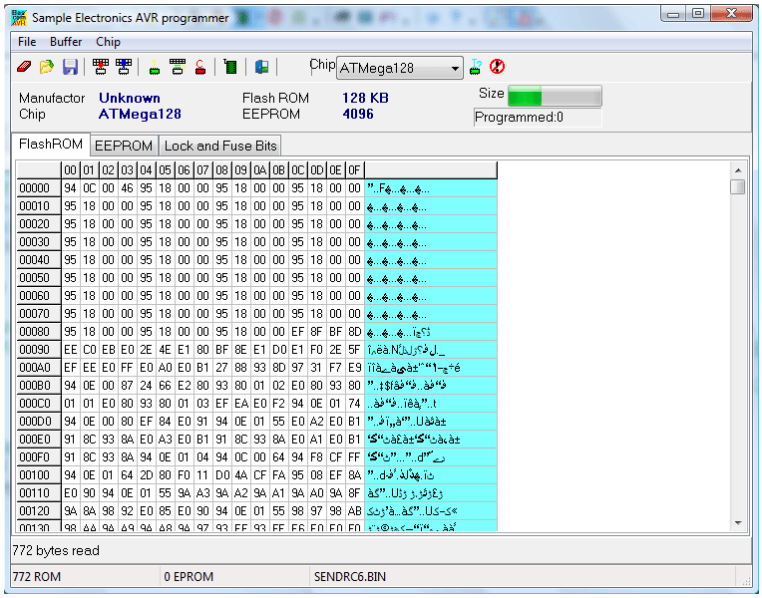
- الواجهة البرمجية الرئيسية: وهي محرر التعليمات والأوامر البرمجية.



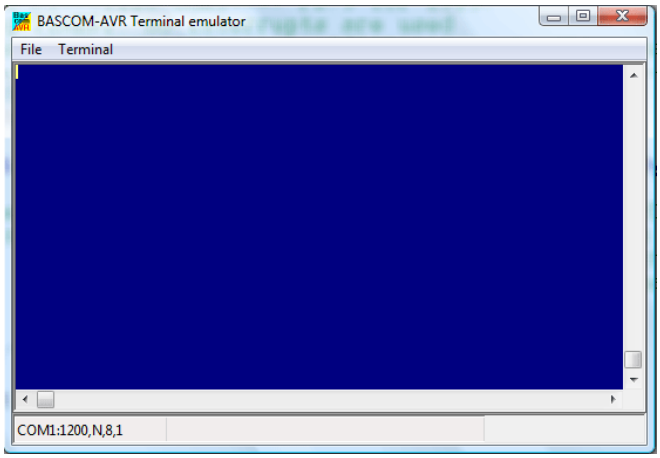
- واجهة المحاكاة: وفيها يتم تشغيل البرنامج خطوة_خطوة ومراقبة حالة المسجلات الداخلية الذواكر.



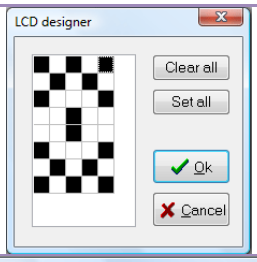
- واجهة المبرمجة: وفيها يتم برمجة المعالج بعد إجراء عملية توليد الملف البرمجي بالأمر Compile.



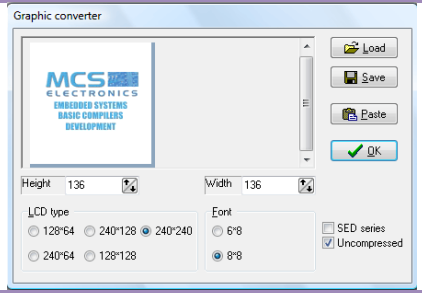
- واجهة الربط البيني: وفيها يتم عرض المعلومات المرسلة والمتلقاة بين المعالج والحاسب بهدف مراقبة بارامترات النظام بشكل آني.



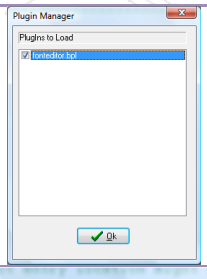
بالإضافة إلى الواجهات الأربعة يملك برنامج Bascom-AVR أدوات مساعدة وهي:



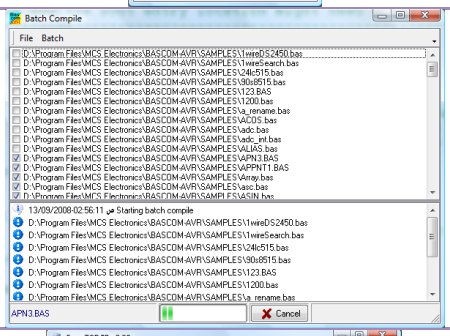
أداة تصميم المحارف (LCD Designer): وتستخدم لتصميم المحارف التي لا توجد على لوحة مفاتيح الحاسب من أجل إظهارها على شاشة الإظهار الكريستالية.



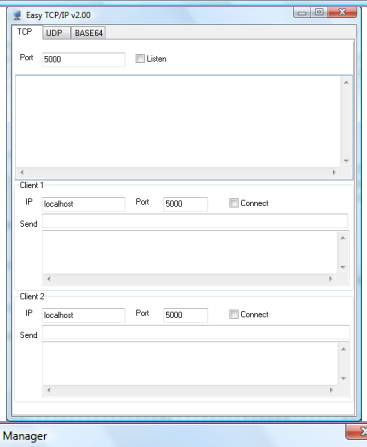
أداة تحويل الصور (Graphic Converter): وتستخدم لتحويل امتداد الصور المراد إظهارها على شاشة الإظهار الرسومية GLCD إلى الصيغة bgf.*



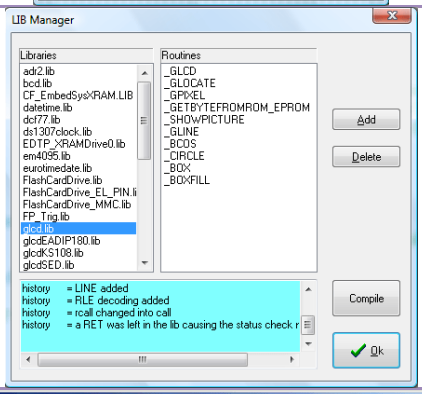
أداة مدير الإضافات (Plugin Manager): وتستخدم لإضافة/حذف الأدوات والموديولات الخارجية.



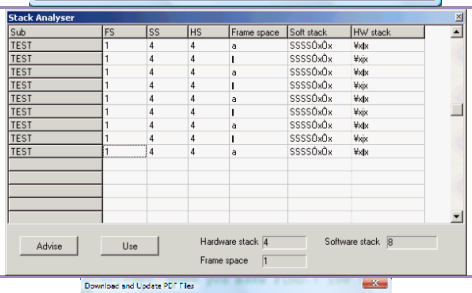
أداة مترجم الملفات المتعددة (Patch Compiler): وتستخدم لتوليد الملف البرمجي لعدة ملفات في آن واحد.



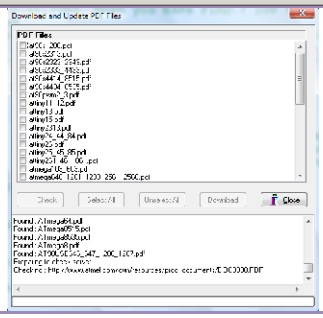
أداة التحكم بالبروتوكول TCP/IP: تستخدم للتحكم ومراقبة المعلومات الموجودة على خط المعطيات.



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



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



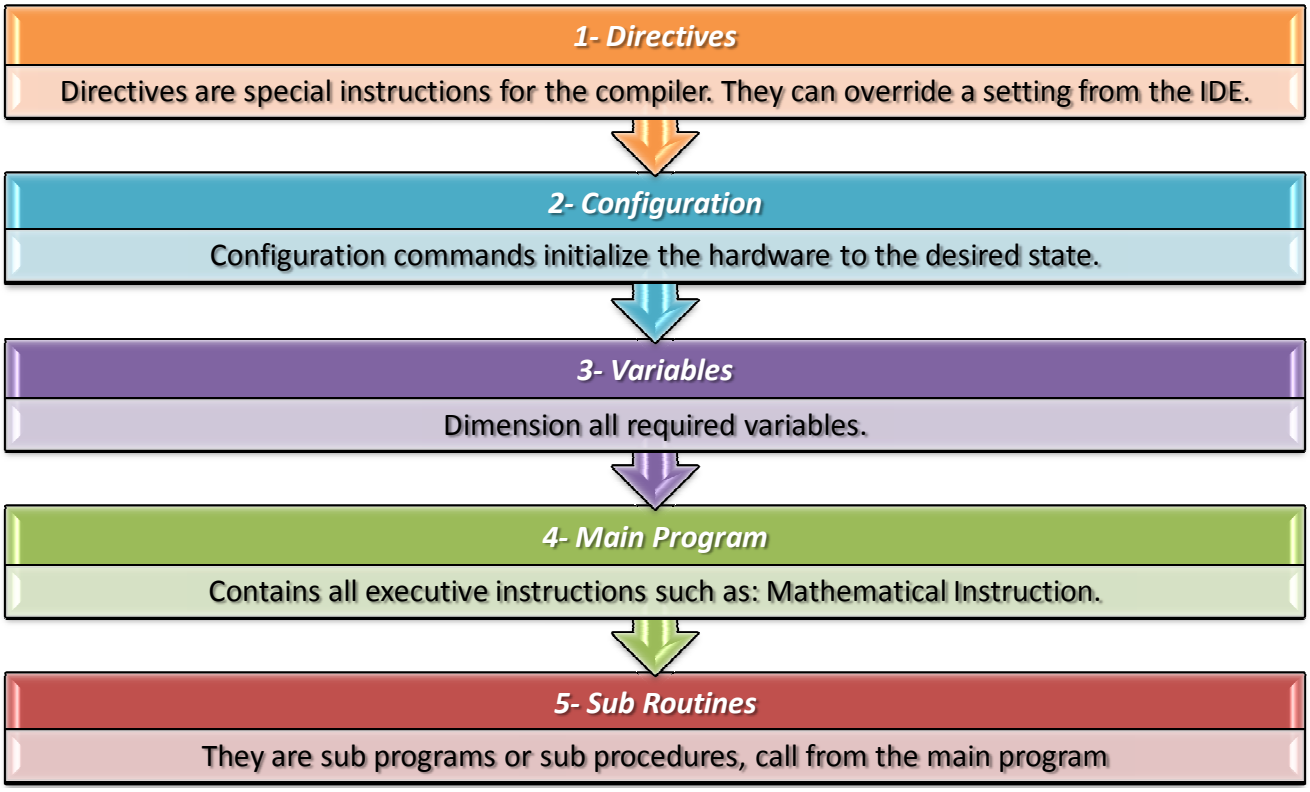
أداة ملفات الوثائق الفنية (PDF Update): تقوم هذه الأداة بالاتصال مع موقع شركة ATMEL وإحضار آخر تحديث للوثائق الفنية للمعالجات المستخدمة من العائلة AVR.

أداة تحويل الكود البرمجي إلى ملف نصي (Export to RTF fle): تصدر الكود إلى ملف WORD.

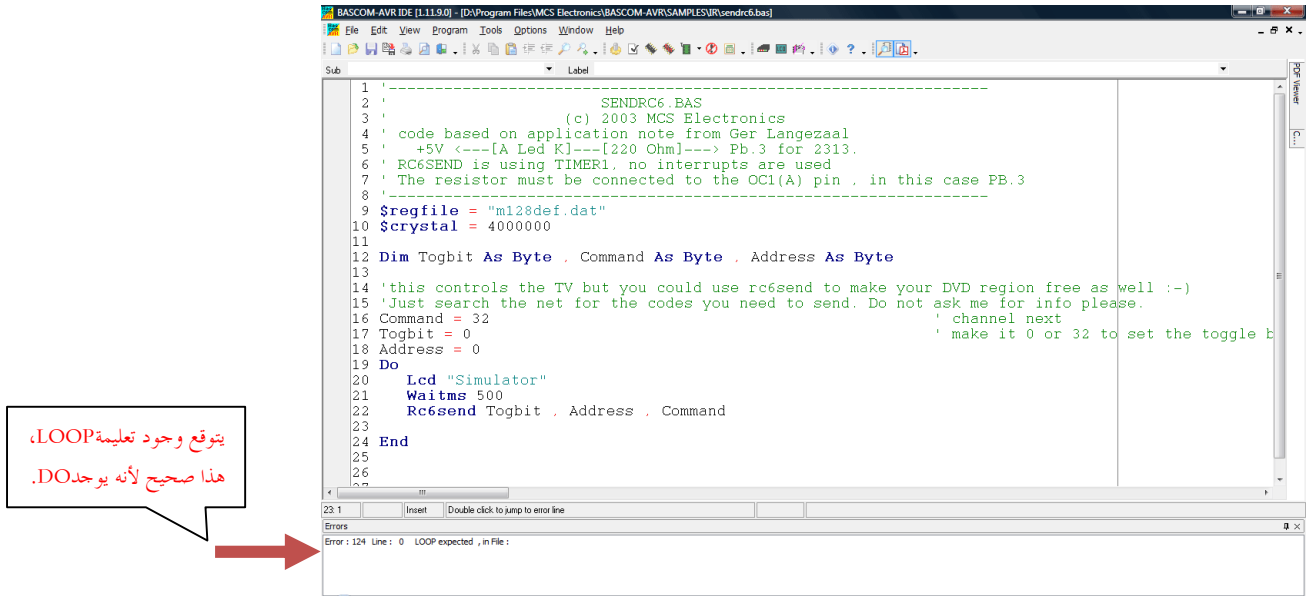
أداة تحويل الكود البرمجي إلى ملف نصي (Export to HTML): تصدر الكود إلى صفحة إنترنت.

خطوات كتابة كود البرمجي في البرنامج Bascom-AVR وبرمجته على شريحة متحكم مصغر:

1. قم بتشغيل البرنامج من قائمة البرامج واختر ملف جديد.
2. من أجل كتابة كود برمجي متماسك ومفهوم مع إمكانية تعديله أو تطويره بسهولة مستقبلاً، فإنه يجب الالتزام بالهيكلية التالية في مراحل كتابة البرنامج:



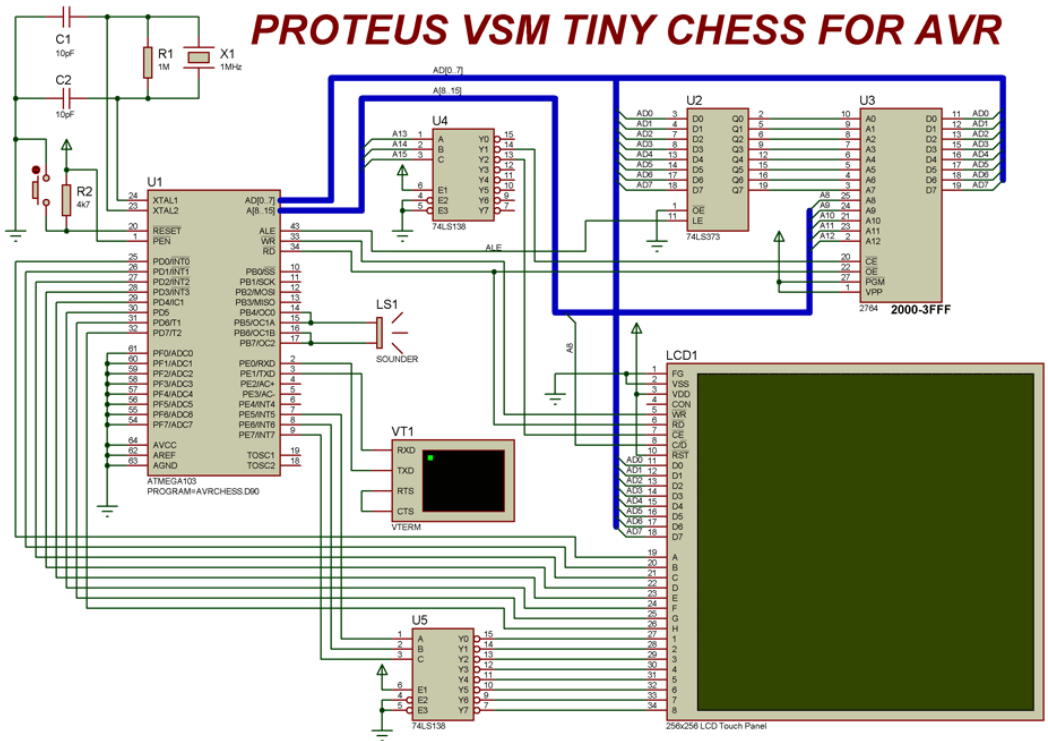
3. بعد الانتهاء من كتابة الكود قم باختيار أمر تفحص الأخطاء (Syntax Check) من القائمة (Program).
4. في حال وجود خطأ برمجي سوف تشير نافذة الأخطاء (أسفل الواجهة الرئيسية) إلى موقع الخطأ وسببه.



5. بعد الانتهاء من تفحص الأخطاء، قم باختيار أمر الترجمة (Compile) من القائمة (Program)، سيقوم البرنامج بتوليد الملفات البرمجية اللازمة للمبرمجة.
6. قم باختيار أمر الإرسال إلى المبرمجة (Send to programmer) من القائمة (Program).

قائمة تصنيف أنواع التعليمات في البيئة البرمجية Bascom-AVR حسب وظيفتها ...

بيئة المحاكاة ISIS Proteus: ISIS Proteus Simulation



يعتبر برنامج Proteus من أقوى برامج المحاكاة لدارة المتحكمات المصغرة وهو يملك العديد من المكتبات التي تغطي جميع أنواع المحيطيات التي يمكن وصلها مع المتحكم المصغر بالإضافة إلى أدوات القياس العديدة.

سوف نستخدم هذا البرنامج لمحاكاة جميع لتجاب التي سوف نتطرق إليها لاحقاً...

مقارنة بين أشهر عائلات المتحكمات المصغرة: Comparison between most famous µC families

الجدول التالي يقارن يبين الميزات الأساسية للمتحكمات AVR, PIC, 8051.

	AVR	PIC	8051
تردد المعالج الأعظمي	16MHZ	20MHZ	24MHZ
عدد النبضات لكل تعليمة	1Cycle	4Cycle	12Cycle
عدد التعليمات في الثانية	16/1 = 16MIPS	20/4 = 5MIPS	24/12 = 2MIPS
نسبة الميزات المحيطية	100%	70%	50%
عدد التعليمات	132	35	215
حجم ذاكرة البرنامج	>256KBytes	<64KBytes	<32KBytes
بنية الذاكرة	Liner	banked	Liner
بنية المعالج	RISC/Harvard	RISC/Harvard	CISC/Von Neumann
عرض ناقل البيانات	16 Bit	12 Bit	8 Bit

متحكمات العائلة AVR® 8-Bit: AVR Microcontrollers Family

تضم عائلة المتحكمات AVR عشرة أصناف أساسية وذلك تبعاً لنوع التطبيق المستخدم موضحة بالجدول التالي:

الصنف	الاستخدام
Automotive AVR	تستخدم في أنظمة التحكم بالمحركات وأنظمة التحكم بالسيارات
AVR Z-Link	تستخدم في بروتوكولات الإرسال الراديوي اللاسلكي في IEEE 802.15.4 / ZigBee
Batery Management AVR	تستخدم للتحكم في شحن المدخرات ومراقبتها وهي تعمل في جهود مرتفعة 1.8~25 فولت
CAN AVR	تستخدم للتحكم ببروتوكول الشبكات CAN وتدعم: CANopen, DeviceNet, OSEK
LCD AVR	تستخدم كمعالجات تشغيل أساسية لشاشات الإظهار الكريستالية LCD
Lighting AVR	تستخدم في تطبيقات التحكم الاستطاعية بسرعة المحركات وشدة الإضاءة
USB AVR	تستخدم كموزع أو مخدم للبروتوكول USB
Tiny AVR	تستخدم لأغراض التحكم العامة وتتميز بصغر حجمها
MEGA AVR	تستخدم لأغراض التحكم العامة وبميزات متعددة وبتردد عمل أعظمي 20MIPS
XMEGA AVR	تستخدم لأغراض التحكم العامة وتعتبر أضخم العائلات وميزاتها كثيرة جداً 32MIPS
AT90Sxxxx	وقد تم استبدالها مؤخراً بالعائلة MEGA وهي تشكل متحكمات العائلة المتوسطة

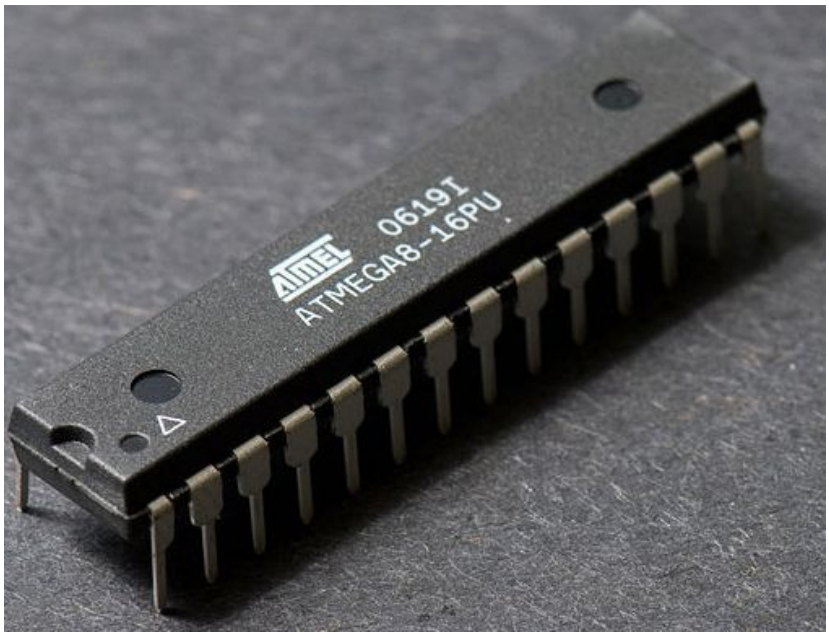
BatteryM AVR	Lighting AVR	USB AVR	megaAVR	tinyAVR
18 ~ 48 Pin	24 ~ 32 Pin	32 ~ 64 Pin	28 ~ 100 Pin	8 ~ 32 Pin
MAX I/O 4~18	MAX I/O 19~27	MAX I/O 22~48	MAX I/O 23~86	MAX I/O 6~28
4KB~40KB Flash	8KB~16KB Flash	8KB~128KB Flash	4KB~256KB Flash	1KB~8KB Flash
256B~1KB EPROM	512B EPROM	512B~4KB EPROM	512B~4KB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

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

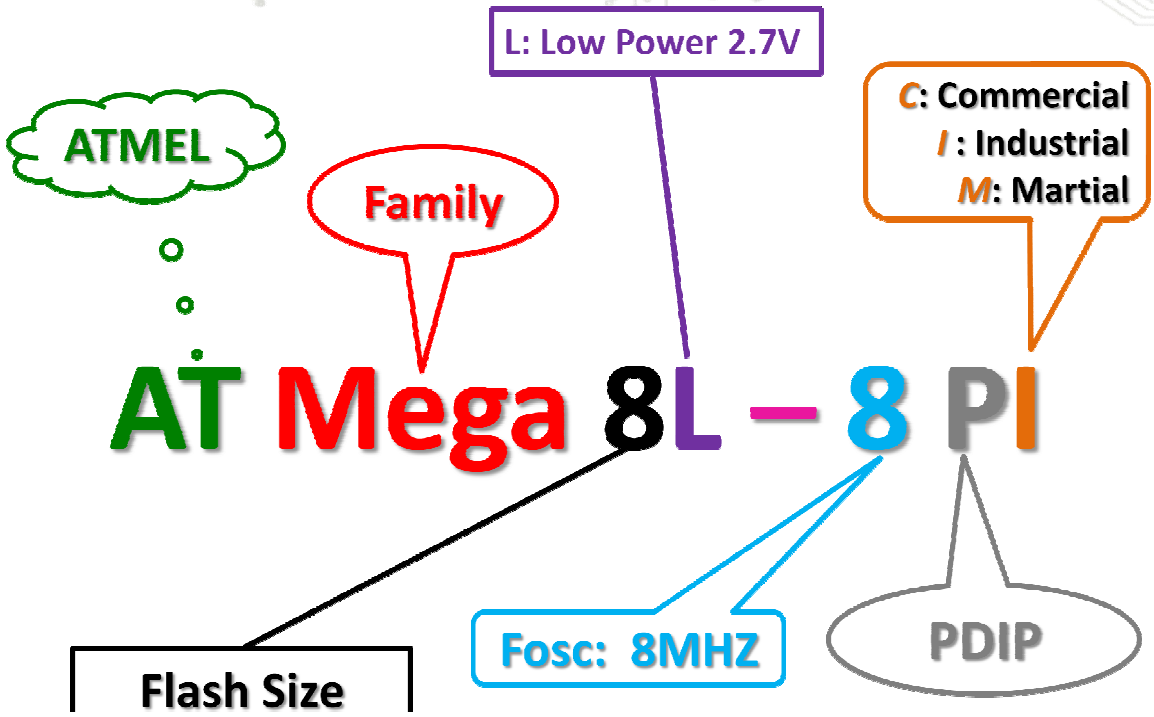
Reading AVR Package information — قراءة تكويد معالجات العائلة AVR:

إن أي شريحة متكاملة تملك رقماً (كوداً) على الوجه العلوي لها يعطي هذا الرقم دلالة معينة لهذه الشريحة.

بالنسبة لمعالجات العائلة AVR فإنها تملك – تكويداً – مخصصاً كما في الشكل التالي:



إن هذا التكويد يعطي معلومات أساسية عن الشريحة لها الدلالات التالية:



AT: هي اختصار لاسم الشركة المصنعة ATMEL.

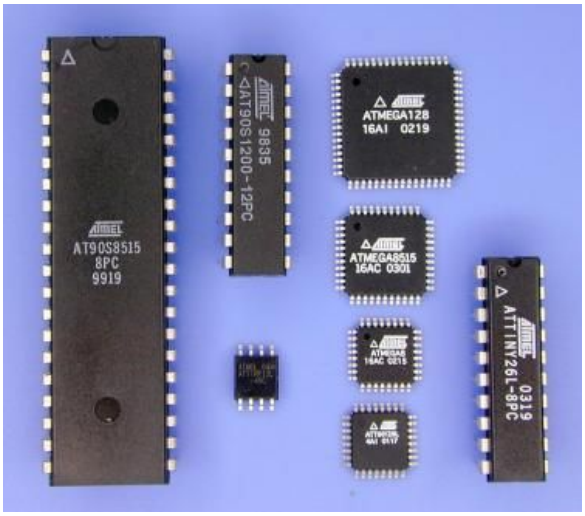
Mega: العائلة التي ينتمي إليها هذا المعالج.

8: هذا الرقم يعبر عن حجم ذاكرة البرنامج ويمكن أن يكون: 8,16,32,64,128,256, etc...

L: في حال أن المعالج يملك حرف L فهذا يعني أنه قابل على العمل بجهود منخفضة (2.7V~5.5V) وبدون هذا الحرف – كما في الصورة أعلاه – فهذا يعني أن المعالج يعمل عند جهود (4.5V~5.5V).

8: تدل على تردد العمل الأعظمي للمعالج، ويمكن أن تكون 16 أو 20.

P: تدل على شكل غلاف الشريحة، فإما أن تكون من نوع يتم لحامه على الطبقة السفلية للدارة المطبوعة (PDIP)، أو من النوع السطحي الذي يتم لحامه على الطبقة العلوية للدارة المطبوعة (SOIC, TQFP, LQFP...).



I: تدل على نوع التطبيق الذي يستخدم المعالج لأجله، فإما أن يكون تجارياً (C) أو صناعياً (I) أو عسكرياً (M) والاختلاف في ذلك هو من حيث قدرة المعالج على تحمل درجات الحرارة والضجيج العالي.

كيفية قراءة الوثائق الفنية للدارات المتكاملة: Reading Datasheet of ICs

عند شراء دارة متكاملة وأريد تشغيلها، فإن أول ما يحتاج إليه لمعرفة كيفية عملها ووظائف أقطابها توزع تلك الأقطاب، هو قراءة المعلومات المهمة من الوثيقة الفنية – Datasheet – الخاصة بالدارة المتكاملة.

وهنا أود التنويه إلى أنه لن يحتاج المبرمج إلى قراءة الوثيقة الفنية كاملةً للمتحكم المصغر من أجل برمجته عن العمل في بيئة البرنامج Bascom-AVR وهذا بدوره يختصر وقتاً كبيراً في تعلم برمجة المتحكمات المصغرة بدون اللجوء إلى دراسة البنية الداخلية للمعالج مفصلةً كما هو الحال عند البرمجة بلغة التجميع (Assembly).

لذلك، سوف أشرح المعلومات التي تفيدنا في الوثيقة الفنية كالميزات الأساسية للمعالج وتوزع الأقطاب، مع العلم أنه لابد – لاحقاً – من العودة إلى بعض التفاصيل في البنية الداخلية للمعالج عن مرحلة متقدمة.

ملاحظة: إن المعلومات التي تهمنا دائماً في الوثيقة الفنية تكون موجودة في الصفحات الثلاث الأولى.

قراءة الوثيقة الفنية للمعالج ATmega128: Reading Datasheet of ATmega128

- الصفحة الأولى والتي تحوي على ميزات المعالج (Features).

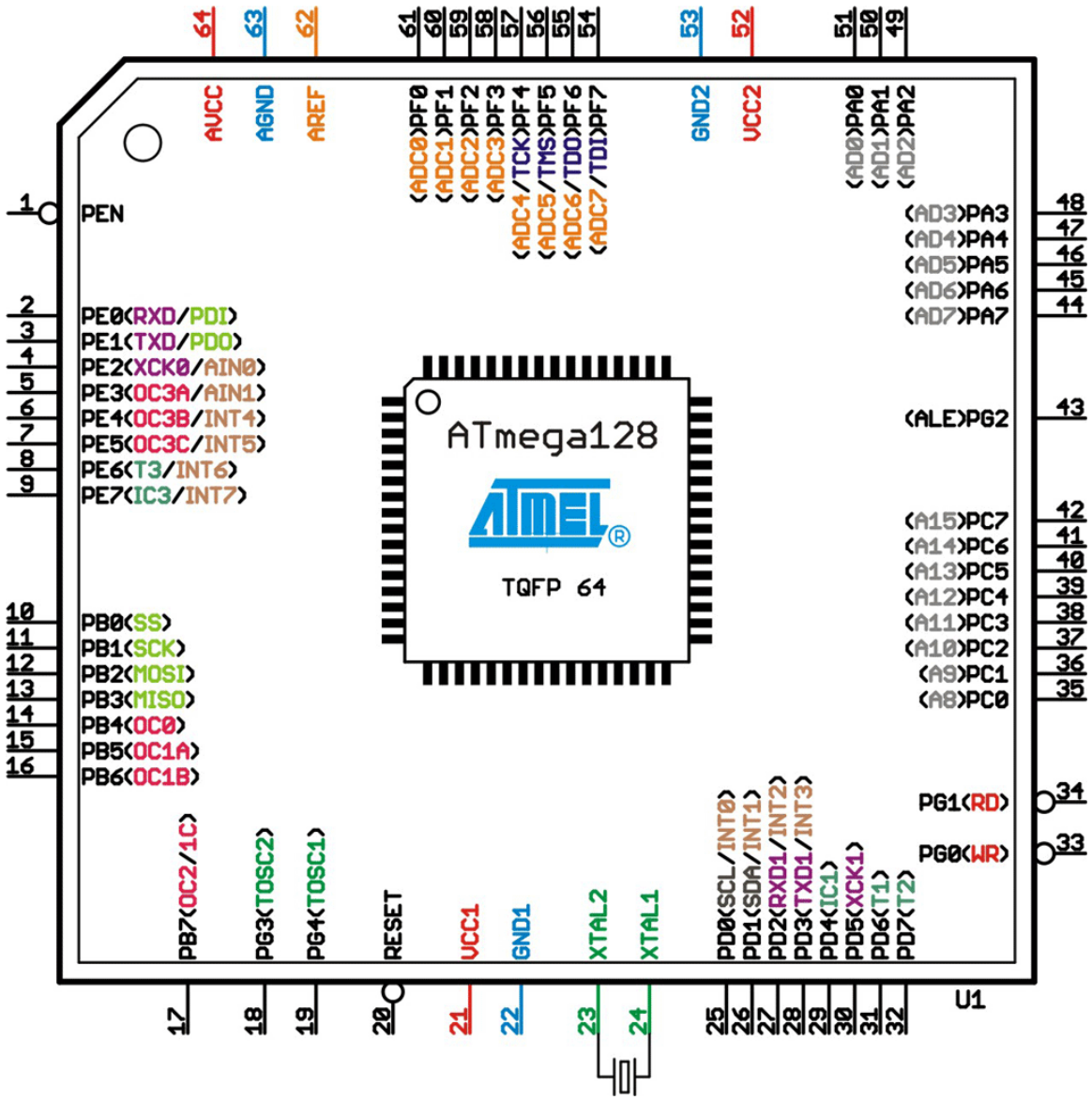
متحكم 8-bit بأداء عالٍ واستهلاك منخفض للطاقة.	High-performance, Low-power AVR® 8-bit Microcontroller
• **بنية متطورة من النوع RISC (أقل عدد ممكن من التعليمات):** – 133 تعليمة معظمها تنفذ بدورة آلة واحدة – 32 x 8 مسجلات أغراض عامة + ومسجلات تحكم محيطية – عمل مستقر ومناعة ضد الضجيج – قادر على تنفيذ 16 مليون تعليمة في الثانية عند تردد 16 MHz – يحوي على مضاعف دورة العمل	• Advanced RISC Architecture. – 133 Powerful Instructions Most Single Clock Cycle. – 32 x 8 General Purpose Working Registers + Peripheral Control Registers – Fully Static Operation – Up to 16 MIPS Throughput at 16 MHz – On-chip 2-cycle Multiplier
• **ذاكرة معطيات دائمة:** – 128KB ذاكرة برنامج يمكن برمجتها بدون فصل المعالج عن الدارة، قابلة للمسح والكتابة 10,000 مرة – أقفال برمجية مستقلة مع قطاع مخصص لكود إقلاع. – 4KB ذاكرة معطيات دائمة EEPROM قابلة للمسح والكتابة 100,000 مرة – 4KB ذاكرة وصول عشوائي مؤقتة SRAM – إمكانية عنونة 64KB (وصل) ذاكرة برنامج خارجية – أقفال برمجية من أجل حماية البرنامج على الشريحة – واجهة ربط تسلسلية (SPI) من أجل برمجة المعالج دون فصله	• Nonvolatile Program and Data Memories – 128K Bytes of In-System Reprogrammable Flash Endurance: 10,000 Write/Erase Cycles – Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program - True Read-While-Write Operation – 4K Bytes EEPROM Endurance: 100,000 Write/Erase Cycles – 4K Bytes Internal SRAM – Up to 64K Bytes Optional External Memory Space – Programming Lock for Software Security – SPI Interface for In-System Programming
• **واجهة اختبار (JTAG):** – قابلية مسح المسجلات الداخلية للمعالج وقراءة حالاتها – دعم متقصي أخطاء (Debug) شامل للشريحة – إمكانية برمجة ذاكرة البرنامج وذاكرة المعطيات.	• JTAG (IEEE std. 1149.1 Compliant) Interface – Boundary-scan Capabilities According to the JTAG Standard – Extensive On-chip Debug Support –Programming of Flash, EEPROM, Fuses and Lock Bits through the JTAG Interface

• **الميزات المحيطية:** – اثنان مؤقت/عداد 8-bit مزود بأنماط مقسم ترددي وحادثة مقارنة – اثنان مؤقت/عداد 16-bit موسع مزود بأنماط مقسم ترددي وحادثة مقارنة وحادثة مسك – عداد الزمن الحقيقي مع هزاز مستقل – قناتي خرج (PWM) تعديل عرض النبضة 8-bit – ستة قنوات خرج (PWM) تعديل عرض النبضة 16-bit مع إمكانية التحكم بالدقة من 2 وحتى 16 بت – ثمان قنوات تبديل تشابهي/رقمي بدقة 10-bit – نافذة اتصال تسلسلية ثنائية (I^2C) – نافذتي اتصال تسلسلي (USARTs) قابلة للبرمجة – نافذة اتصال تسلسلية (SPI) بنمطي عمل قائد/تابع – مؤقت مراقبة قابل للبرمجة مع هزاز مستقل – نافذة مقارن تشابهي	• Peripheral Features –Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes –Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode – Real Time Counter with Separate Oscillator – Two 8-bit PWM Channels – 6 PWM Channels with Programmable Resolution from 2 to 16 Bits – Output Compare Modulator – 8-channel, 10-bit ADC – Byte-oriented Two-wire Serial Interface – Dual Programmable Serial USARTs – Master/Slave SPI Serial Interface – Programmable Watchdog Timer with On-chip Oscillator – On-chip Analog Comparator
• **الميزات الخاصة للمعالج:** – تصفير عند وصل التغذية وكاشف انخفاض جهد التغذية للشريحة – هزاز داخلي معاير – مصادر مقاطعة خارجية وداخلية – ستة أنماط لتخفيض الطاقة ولتخفيض ضجيج المبدل – إمكانية تحديد تردد الهزاز الداخلي برمجياً – نمط تلاؤمي مع المعالج ATmega103 يحدد عن طريق الفيوزات – إلغاء شامل لمقاومات الرفع الداخلية للبوابات	• Special Microcontroller Features – Power-on Reset and Programmable Brown-out Detection – Internal Calibrated RC Oscillator – External and Internal Interrupt Sources – Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby – Software Selectable Clock Frequency – ATmega103 Compatibility Mode Selected by a Fuse – Global Pull-up Disable
• **عدد أقطاب الدخل/الخرج وشكل الغلاف الخارجي المعالج:** – 53 قطب دخل/خرج قابل للبرمجة من أجل الغلاف PDIP – 64 قطب دخل/خرج قابل للبرمجة من أجل الغلاف TQFP	• I/O and Packages – 53 Programmable I/O Lines–PDIP – 64-lead TQFP and 64-pad MLF
• **جهود العمل للشريحة:** – جهد عمل في المجال 2.7 - 5.5V أجل المعالج ATmega128L – جهد عمل في المجال 4.5 - 5.5V أجل المعالج ATmega128	• Operating Voltages – 2.7 - 5.5V for ATmega128L – 4.5 - 5.5V for ATmega128
• **درجات سرعة عمل المعالج:** – تردد عمل أعظمي حتى 8-MHz من أجل المعالج ATmega128L – تردد عمل أعظمي حتى 16-MHz من أجل المعالج ATmega128	• Speed Grades – 0 - 8 MHz for ATmega128L – 0 - 16 MHz for ATmega128

بالنظر إلى هذه الميزات المذكورة أعلاه فإن معظمها ربما يكون غريباً للوهلة الأولى، ولكن سيأتي دو كل منها مفصلاً في التجارب اللاحقة.

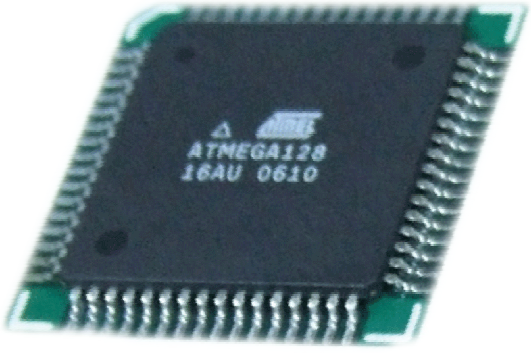
إن هذه الميزات تختلف من معالج إلى آخر من حيث تطورها وعددها، ويعتبر المعالج ATmega128 من أكثر المعالجات المتقدمة في العائلة ATmegaxxx وبالتالي فإننا ندرس الآن الحالة الأعم والأشمل.

- الصفحة الثانية تحوي على توزع أقطاب المعالج (Pin Configurations).

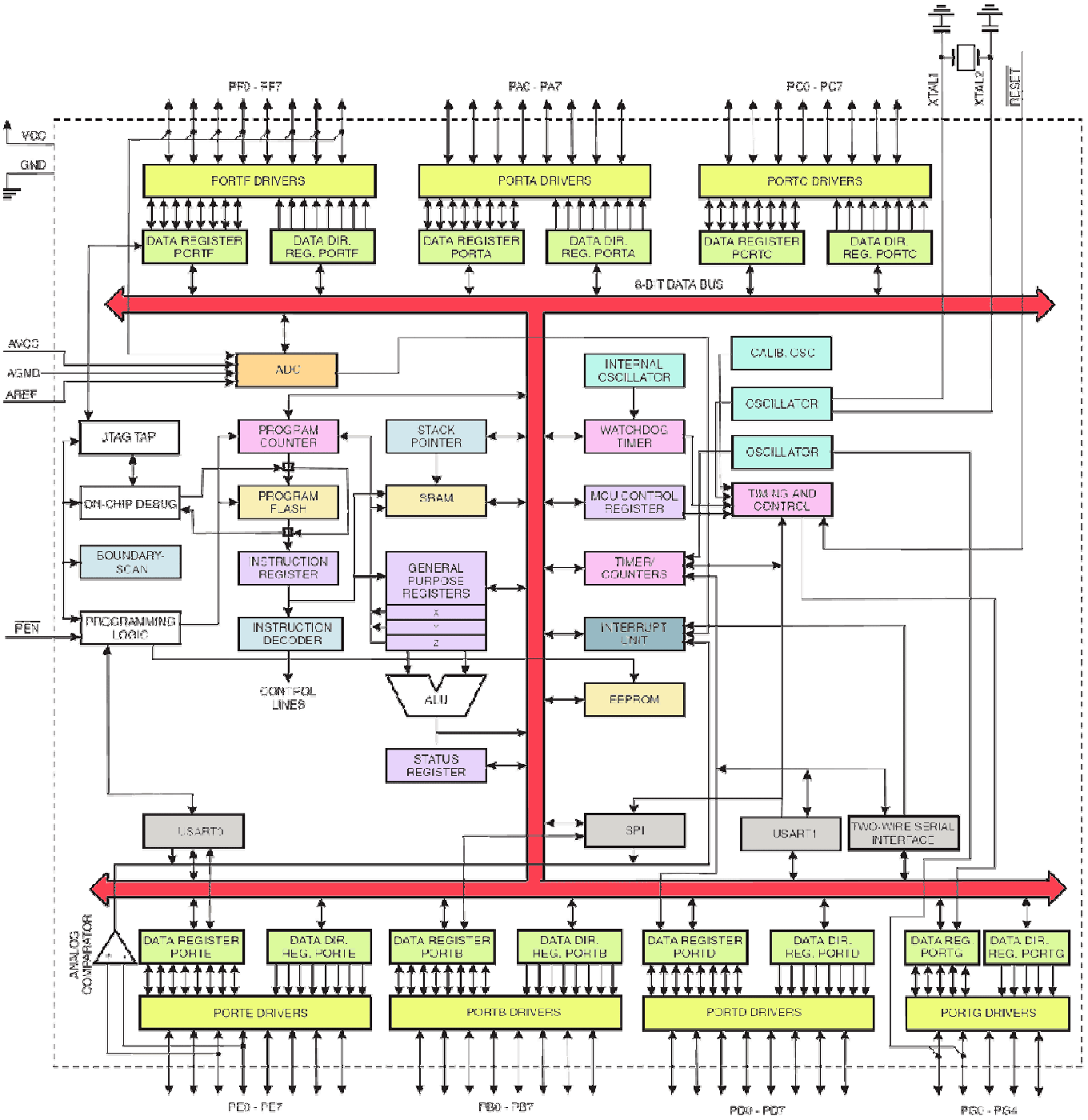


إن معظم الأقطاب تملك أكثر من وظيفة ومثال ذلك:

القطب رقم 2 (PE0(RXD/PDI: إن هذا القطب يملك ثلاث وظائف مختلفة ومنفصلة عن بعضها (لا يمكن أن تعمل جميعها في نفس الوقت)، حيث إما أن يكون قطب دخل/خرج من البوابة PE0 أو أن يكون قطب الاستقبال للنافذة التسلسلية UART أو أن يكون قطب الاستقبال للنافذة JTAG. سيأتي لاحقاً شرح وظائف الأقطاب عملياً من خلال التجارب.



- الصفحة الثالثة تحوي على المخطط الصندوقي للبنية الداخلية للمعالج (Block Diagram).



إن مخطط البنية الداخلية للمعالج يعطي فكرة عامة عن طريقة ربط الوحدات المحيطية والمسجلات مع وحدة المعالجة المركزية.

إن البنية الداخلية لمعالجات العائلة AVR تم تصميمها وفقاً لمعمارية Harvard وتقنية RISC، ولابد من شرح بسيط لهاتين الفكرتين...

Standard Systems Design معيارية تصميم النظم:

تعرف المعيارية بأنها الطريقة التي يتعامل بها المعالج مع الذاكرة في جلب وتنفيذ التعليمات وتخزين البيانات.

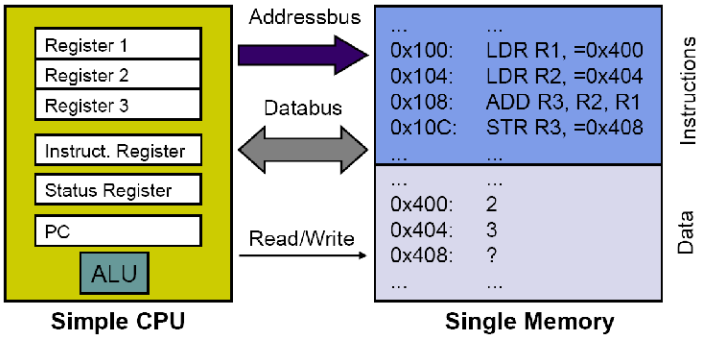
يوجد معياريتين أساسيتين في تصميم المعالجات: معيارية (Harvard) ومعيارية (Von Neumann).

معيارية Von-Neumann:

تتكون هذه المعيارية من المعالج وناقل وحيد لنقل التعليمات والبيانات كل على حدا، وبالتالي سوف يحتاج إلى نبضات ساعة أكثر من أجل تنفيذ عملية واحدة، لذلك تتصف هذه النظم بكونها بطيئة نسبياً.

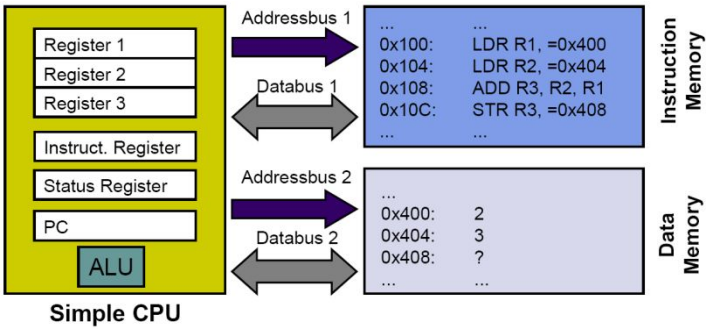
عملها:

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



معيارية Harvard:

تتكون هذه المعيارية من المعالج وناقلين منفصلين أحدهما لنقل التعليمات والآخر لنقل البيانات، وتختلف ذاكرة البيانات عن ذاكرة التعليمات ولكل واحدة خطوط عنونة وتحكم وممر معطيات مختلف عن الأخرى، ولهذه الطريقة ميزة كبيرة وهي أن عملية قراءة التعليمات والبيانات تتم في نفس الوقت وبالتالي سرعة أكبر بكثير في عمل المعالج.



Methods Architecture Systems Design تقنيات تصميم بنية النظم:

يتم تصميم بنية نظم المعالجات وفقاً لثلاث تقنيات:

CISC : Complex Instruction Set Computer (150 ~ 1500 Instruction).

RISC : Reduced Instruction Set Computer (30 ~ 130 Instruction).

MISC: Minimum Instruction Set Computer (15 ~ 30 Instruction).

تتعارض هذه التقنيات في الهدف من تصميم المعالجات، فتقنية CISC تهتم بسرعة عمل المعالج، فلذلك تقوم بوضع تعليمة لكل حدث يتم في المعالج، وبالتالي يمكن أن تصل مجموعة تعليمات هذه المعالجات إلى آلاف

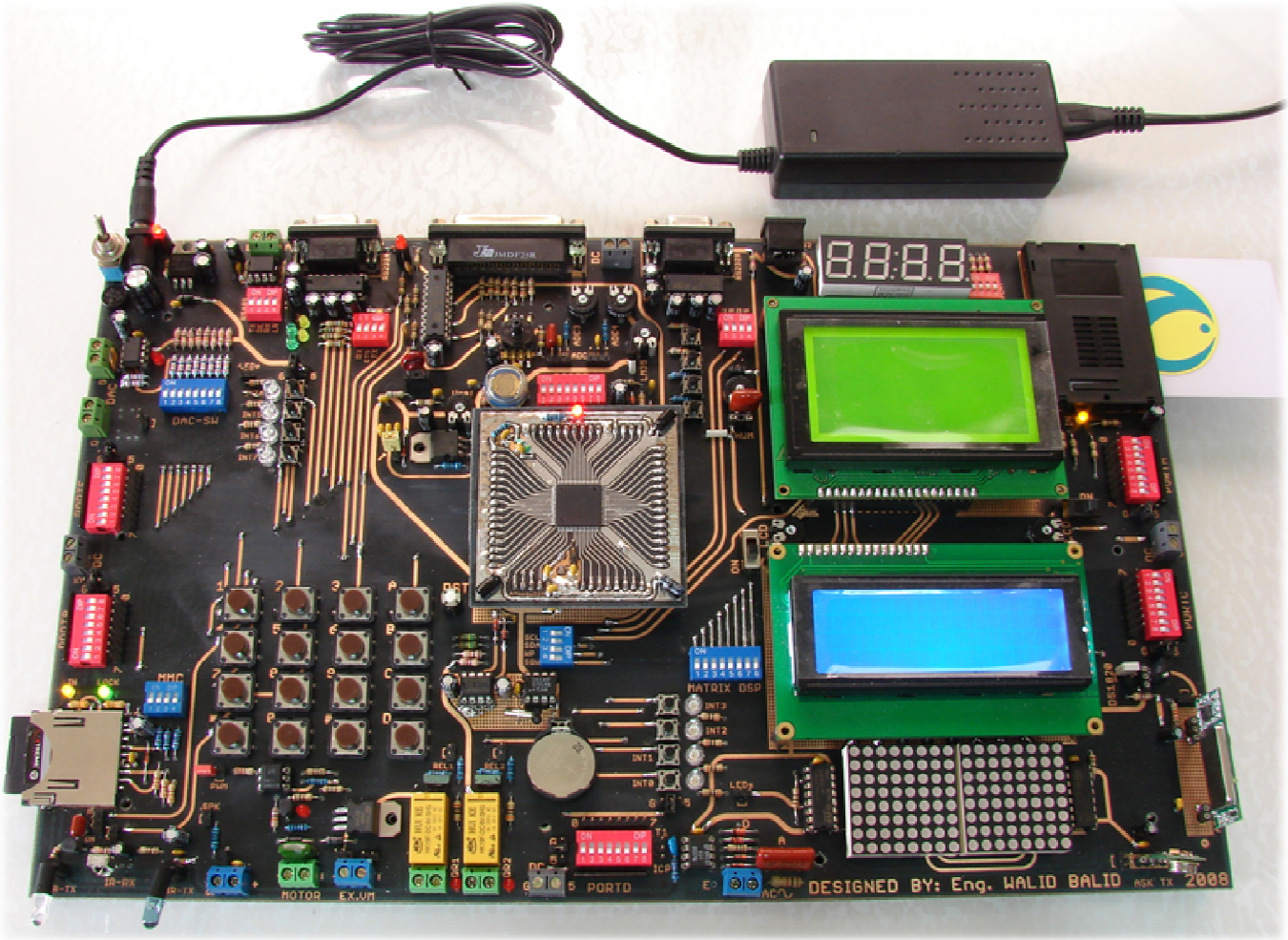
التعليمات المتخصصة، كما أن زيادة عدد التعليمات يؤدي إلى زيادة تعقيد العتاد لأننا سنحتاج إلى مسجلات داخلية إضافية لهذه التعليمات وبالتالي زيادة عدد الترانزستورات ومنه تزداد تكلفة المعالج وتزداد ضياعات الطاقة فيه مما ينتج عنه ارتفاع في درجة حرارته وهذا بالفعل ما نلاحظه في معالجات AMD & INTEL الحديثة.

أما معالجات RISC فتهتم ببساطة التعليمات "استخدام تعليمات عامة" وتقليل عددها ولكن هذا ينعكس سلباً على سرعة أداء البرنامج وطوله، ومن ناحية أخرى فإن تقليل عدد التعليمات يؤدي إلى تقليل العتاد ومنه إلى انخفاض عدد الترانزستورات وبالتالي تخفيف الضياعات الكهربائية وانخفاض درجة حرارة المعالج و سعره، ومع وجود لغات برمجة عالية المستوى لم تعد هناك مشكلة تعقيد البرنامج في المعالجات ذات البنية RISC، وهذه البنية تعتمدها معظم أنواع المتحكمات المصغرة (Microcontrollers) ومعالجات الإشارة الرقمية (DSPs).

مؤخراً أصبح من المهم جداً وجود معالجات تعمل تحت جهود عمل منخفضة جداً واستجرار تيار تشغيل أصغري (µA)، ومن أجل هذا يجب تقليل عدد المسجلات الداخلية للمعالج من خلال تقليل عدد التعليمات، ولهذا وجدت بنية جديدة MISC، ولكن هذا النوع من المعالجات يتم برمجته باستخدام مترجمات عالية المستوى.

AVR Development Board لوحة التطوير لعائلة المتحكمات AVR:

لقد تم تصميم هذه البورد بحيث تخدم المبتدئ والمتقدم في تعلم برمجة متحكمات العائلة AVR، حيث أنها تضم أكثر من 50 تجربة مختلفة ومستقلة بالإضافة إلى إمكانية ربط موديولات خارجية عن طريق وحدات التوسعة المحيطية، كما أنها تعتمد في تصميمها على تقنيات التعليم التفاعلي -المتطورة- بين المستخدم والحاسب وهو ما يسمى بـ: (Human-Computer Interaction Methods).



التجارب الأساسية للوحة التطوير مرتبةً:

1. استثمار بوابة المتحكم كبوابة خرج من أجل تشكيل حركة ضوئية باستخدام الثنائيات ضوئية.
2. استثمار بوابة المتحكم كبوابة دخل لقراءة حالة مفاتيح لحظية.
3. استثمار بوابات المتحكم كبوابات دخل/خرج لأغراض التحكم.
4. تشغيل زمني لمخارج تحكم استطاعية (ريليه).
5. برمجة لوحة مفاتيح ست عشرية موصلة بطريقة مصفوفة.
6. برمجة شاشة إظهار كريستالية محرفية LCD 20x4.
7. برمجة شاشة إظهار كريستالية رسومية GLCD 128x64.

8. توليد نغمات صوتية.
9. توليد نغمات DTMF.
10. تعليمات الإزاحة والدوران.
11. تحويل قطب رقمي إلى قطب تشابهي لأغراض قياس السعات والمقاومات.
12. برمجة مستقبل أشعة تحت الحمراء وفق البروتوكولات RC5, RC5-Extended.
13. برمجة مرسل أشعة تحت الحمراء لأغراض التحكم عن بعد وفق البروتوكولات RC5, RC5-Extended.
14. تشغيل لوحة إظهار سباعية فردية كعداد عشري 9 – 0.
15. تشغيل أربع لوحات إظهار سباعية كعداد عشري 9999 – 0.
16. برمجة النافذة التسلسلية UART1 وتقنيات الربط التسلسلي مع الحاسب باستخدام البروتوكول RS232.
17. برمجة النافذة التسلسلية UART2 وتشكيل نوافذ تسلسلية إضافية برمجياً.
18. نقل المعطيات باستخدام بروتوكول الاتصال التسلسلي التفاضلي RS485.
19. التحويل التشابهي الرقمي والجهد المرجعي للمبدل.
20. قياس شدة الإضاءة.
21. قياس قيمة مقاومة ومكثف.
22. قياس درجة الحرارة في المجال $-45^{o}C$ ~ $+100^{o}C$ باستخدام حساس حرارة تشابهي LM35DZ.
23. ربط لمقاومة ذات العامل الحراري السالب NTC مع مبدل ADC لأغراض قياس درجات الحرارة العالية.
24. طريقة ربط عدة مفاتيح لحظية إلى قطب وحيد.
25. ربط حساس قياس الضغط الجوي إلى مبدل ADC.
26. كاشف العوائق باستخدام الأشعة تحت الحمراء.
27. نقل البيانات لاسلكياً باستخدام الأشعة تحت الحمراء.
28. نقل البيانات لاسلكياً باستخدام الأمواج الراديوية.
29. التحكم بسرعة محرك تيار مستمر باستخدام تقنيات تعديل عرض النبضة (PWM).
30. قياس تردد إشارة منبع كهربائي عالي.
31. مقياس وعداد ترددي رقمي.
32. البرامج الوظيفية وتمرير المتحولات وهيكلية تنظيم البرامج المعقدة لتسهيل تطويرها مستقبلاً.
33. ربط شبكة لادر (Ladder) لأغراض تحويل الإشارة الرقمية إلى إشارة تشابهية (DAC).
34. استخدام تعديل عرض النبضة مع مرشح لأغراض تحويل الإشارة الرقمية إلى إشارة تشابهية (DAC).

35. تقنيات التحكم بالمحركات الخطوية.
36. برمجة شريحة توليد الزمن الحقيقي (RTC) لأغراض التوقيت الدقيق.
37. تخزين وقراءة المعطيات على ذاكرة خارجية EEPROM.
38. ساعة رقمية بالاعتماد على ساعة الزمن الحقيقي الداخلية للمعالج.
39. برمجة المؤقتات في أنماط العمل: Overflow, Compare & Capture modes.
40. تخزين البيانات على شريحة وسائط رقمية (MMC).
41. استخدام البطاقة الذكية (Smart Card) لأغراض التشفير والحماية.
42. ربط لوحة مفاتيح حاسوبية (PS2) مع المتحكم وطباعة الأحرف والأرقام على شاشة الإظهار المحرفية.
43. ربط فأرة حاسوبية (PS2) مع المتحكم وإظهار الحركة على شاشة الإظهار الرسومية.
44. استثمار البروتوكول 1-Wire من خلال ربط حساس حرارة رقمي تسلسلي (DS1821) مع المتحكم.
45. برمجة وحدات إظهار نقطية (LED-Matrix Displays) لتعمل كجريدة إلكترونية متحركة.
46. الوصلة المطورة للبروتوكول RS232 مع النافذة USART باستخدام خطوط المصافحة.
47. برمجة ذاكرة المعطيات الداخلية للمعالج EEPROM.
48. برمجة التعليمات الرياضية المعقدة.
49. استثمار المقارن التشابهي في المتحكم.
50. قراءة وتوليد قطار نبضات تسلسلي.
51. برمجة أنماط توفير الطاقة ومؤقت المراقبة.
52. تتبع حالة مسجلات المعالج الداخلية لاكتشاف الأخطاء عن طريق النافذة التسلسلية JTAG.
53. تقنيات تصميم وحدات ربط مخارج التحكم للمتحكم المصغر إلى مخارج استطاعية.
54. اعتبارات تصميم دارات الكيان الصلب (PCBs) للمتحكمات وإلغاء تأثيرات الـ ESD, EMC & EMI.
55. البرمجة التفرعية من أجل وظائف متعددة (Multitasking) في أنظمة الزمن الحقيقي (RTS).
56. ربط موديول خارجي إلى لوحة التطوير لتمييز الصوت المحيط وقياس شدته.
57. ربط موديول خارجي إلى لوحة التطوير لبرمجة وتشغيل شريحة تسجيل صوتي عن طريق النافذة SPI.

خطة العمل على المشروع: AVR Development Board Designing schedule

1- البحث المكتبي عن جميع الشركات المصنعة للوحات التطوير المتخصصة في برمجة المتحكمات AVR.

2- جمع الميزات والوظائف الأساسية لهذه اللوحات ومقاطعتها مع بعضها للحصول على ميزات جديدة متكاملة مستندة إلى تلك الميزات.

3- دراسة الحاجة إلى تجارب عملية تغطي معظم الاختصاصات في الهندسة الالكترونية بأقسامها (تحكم، اتصالات، قيادة، إلكترون).

4- جمع هذه الدراسة الأكاديمية وتنفيذها على شكل لوحة تطوير عملية تراعي في تصميمها جميع الاعتبارات التصميمية الاختصاصية (EMC,EMR,ESD) في تصميم لوحات الأنظمة المضمنة (Embedded Systems) بالإضافة إلى مرونة التعامل مع محيطياتها.

5- بناء التجارب الأساسية ليتم برمجتها على النظام أو محاكاتها باستخدام برنامج Proteus.

الاعتبارات التصميمية:

تم تصميم هذا النظام وفقاً للاعتبارات التالية:

1- متوافقة مع جميع البيئات البرمجية (compilers) المستخدمة في برمجة المتحكمات المصغرة من العائلة AVR والمذكورة آنفاً.

2- يمكن برمجة المعالج دون الحاجة إلى فصله عن النظام وهو ما يسمى بـ: (In System Programming).

3- تم وضع المحيطيات لتغطي التجارب العملية لمعظم الاختصاصات الهندسية.

4- تحوي على أكثر من 60 تجربة أساسية مستقلة ويمكن أن تصل إلى أكثر من 100 تجربة بالدمج.

5- تم بناؤها استناداً إلى خبرة عملية كبيرة تزيد عن ستة سنوات في مجال تصميم الأنظمة المضمنة (Embedded Systems Design) واستناداً إلى خبرة في مجال التدريس لمدة تزيد عن ثلاث سنوات وهذا بدوره كان عاملاً أساسياً لاختيار التجارب التي تفيد المبرمج.

6- يستطيع الطلاب التعامل مع اللوحة بشكل مباشر دون الحاجة إلى موجه لتشغيلها (self learners)، كل ما يحتاج هو اتباع الخطوات المشروحة في الدليل.

7- إن عامل الكلفة هو ما يميز لوحة التطوير هذه حيث أن الطلاب يستطيعون بناء لوحاتهم الخاصة بما لا يزيد عن USD120، بينما إذا أردنا الحصول على نفس المحيطيات والوظائف الموجودة في هذه اللوحة فإننا ربما نحتاج إلى أكثر من لوحتي تطوير بالإضافة إلى العديد من الموديلات الإضافية وبالتالي فإن الكلفة ربما تصل إلى USD700.

-8 بالإضافة لكون البرامج (التجارب) يمكن كتابتها وإرسالها إلى النظام بشكل مباشر، فإنه تم بناء المخططات الأساسية للدارة في بيئة برنامج المحاكاة Lab-Center Proteus-7.2 الذي يعد البرنامج الأقوى في محاكاة دارات المعالجات والمتحكمات المصغرة وبالتالي يمكنك تشغيل البرامج التي يتم كتابتها في بيئة البرنامج Bascom-AVR دون الحاجة إلى لوحة التجارب وهذا غير موجود في لوحات التطوير الأخرى!

-9 تم تزويد لوحة التطوير بنافذتي مراقبة (Debugger) حيث يمكن وصل اللوحة مع نافذة الاتصالات التسلسلية للحاسب (RS232 Interface) ومراقبة عمل النظام وقراءة المتحولات.

-10 بالإضافة إلى التجارب الأساسية للوحة التطوير فإنه تم وضع نقاط توسعة لجميع بوابات المتحكم المصغر على أطراف البورد (48 I/O) ليسهل عملية توسعة البورد أو ربط تجارب (موديولات) خارجية يتم بناؤها مستقبلاً.

-11 إن هذه البورد متوافقة مع جميع متحكمات العائلة AVR بالإضافة إلى جميع المتحكمات التي تعتمد على البروتوكول SPI في برمجتها حيث تم مراعاة وضع المعالج على موديول منفصل من أجل تغييره.

-12 تم اختبار لوحة التجارب هذه عملياً في متناول أيدي طلاب المرحلة الجامعية وقد أثبتت قدرتها على التجاوب المباشر ومرونة العمل عليها.

-13 تم إجراء استقصاء رأي حول هذه اللوحة وكانت النتائج:

a. مستوى إلمام الطلاب في برمجة المتحكمات المصغرة من قبل هذا كان: 5%
b. مستوى إلمام الطلاب ببرمجة المتحكمات المصغرة بعد انتهاء التجارب كان: 85%
c. ساهمت هذه المحاضرات العملية التطبيقية على هذه البورد في إغناء المعرفة ببرمجة المتحكمات المصغرة بنسبة: 90%.
d. كان تقييم الطلاب لدرجة استفادتهم من المحاضرات والتجارب بنسبة: 96%.
e. كان تقييم الطلاب لسهولة التعامل مع لوحة الاختبار والتجارب بنسبة: 98%.
f. كان نسبة الطلاب الذين رغبوا في الحصول على لوحة التجارب الخاصة بهم من خلال الاستفادة من ملفات التصميم أكثر من 45% وهذا يعني 70 طالباً من أصل 150!

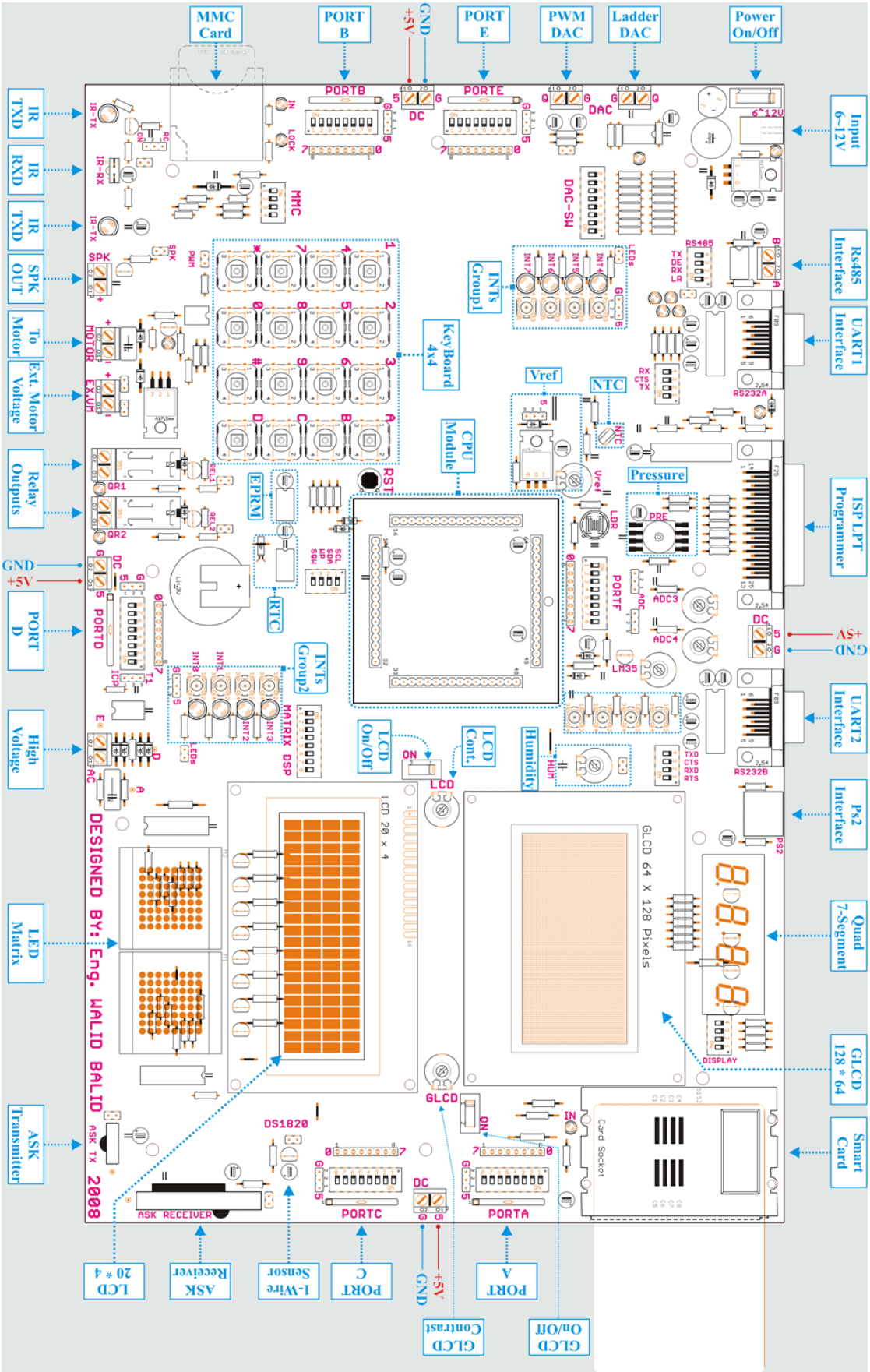
-14 الآن هناك خطة لتسويق لوحة التطوير هذه على مستوى الشرق الأوسط وأوربا لأغراض تدريسية.

-15 أخيراً، ما يميز هذه البورد هو تصميم النظام بأقل عدد ممكن من الطبقات في الدارة المطبوعة (ليسهل على الطلاب تنفيذ لوحات لمخابرهم الإلكترونية المنزلية) ووضع ملفات التصميم والمقرر التدريسي (التجارب العملية) في متناول أيدي الطلاب (Open Source) لإتاحة الفرصة لهم في بناء لوحة التطوير ووضعها في مخبر منزلي مصغر بحيث يستطيع أن يطور نفسه بشكل مستمر.

-16 إمكانية التخاطب مع البورد من خلال البيئات البرمجية LabVIEW, VB6, Matlab, etc...

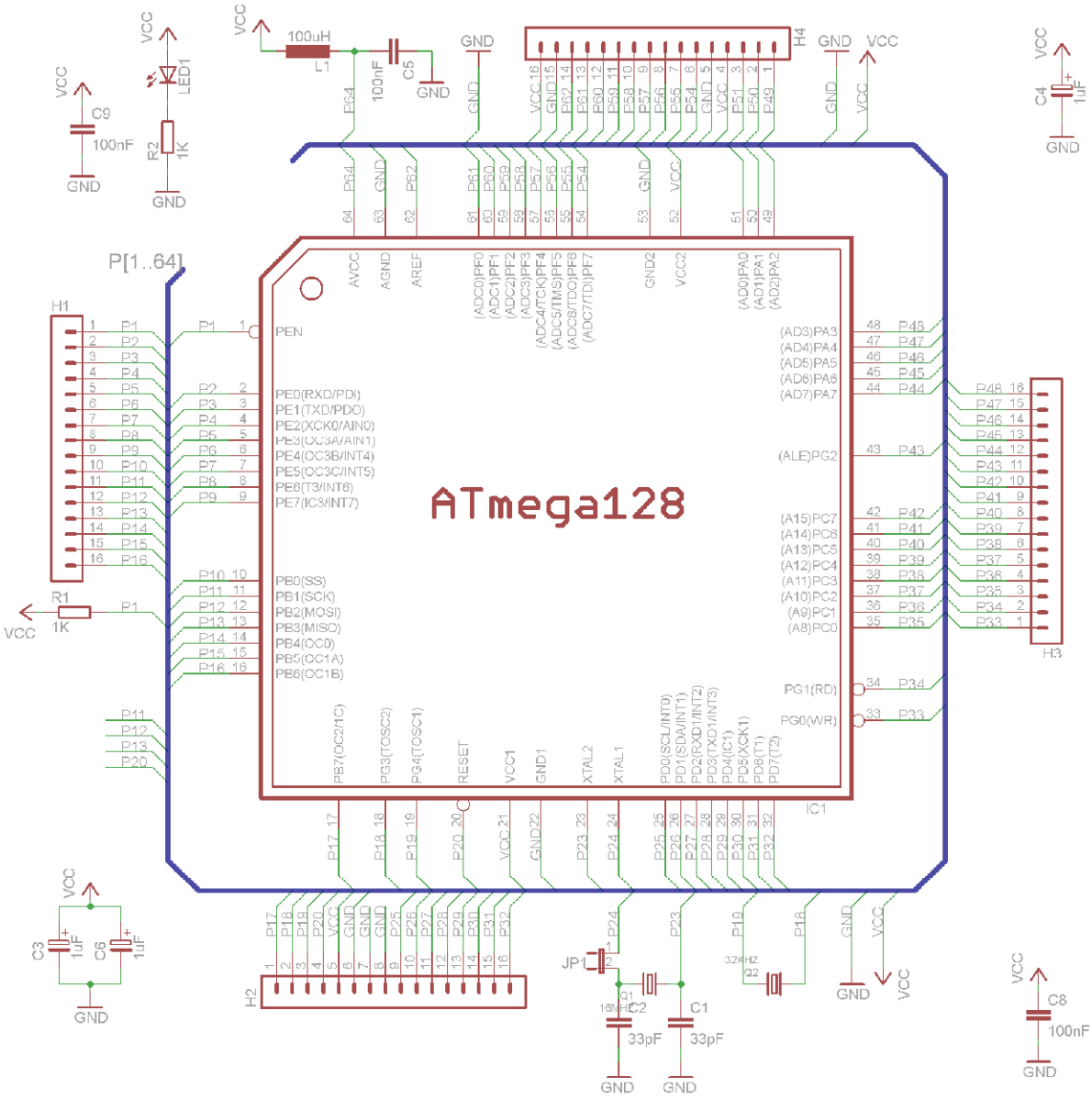
مخطط لوحة التطوير:
Development Board Layout Diagram

الشكل التالي يبين توزع المحيطيات والوظائف الأساسية للجكات على بورد لوحة التطوير...

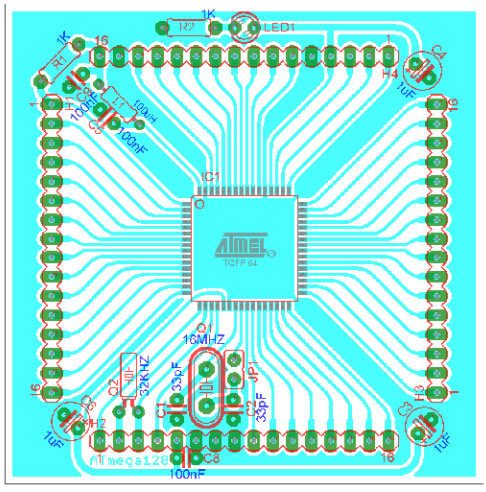


Development Board Schematic & Board Design المخطط التصميمي للوحة التطوير:

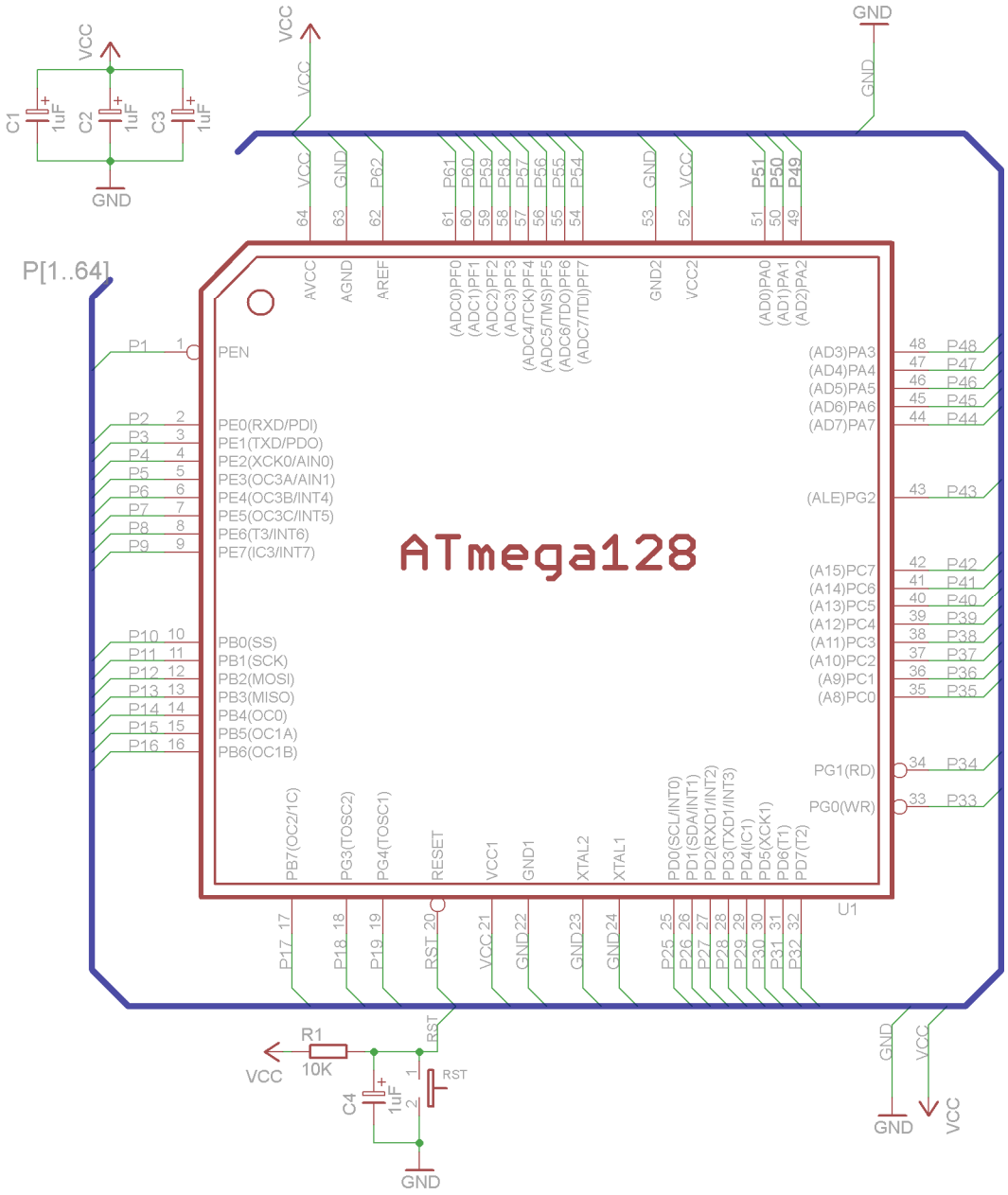
الشكل التالي يبين المخطط التصميمي لموديول المعالج الخاص بلوحة التطوير...



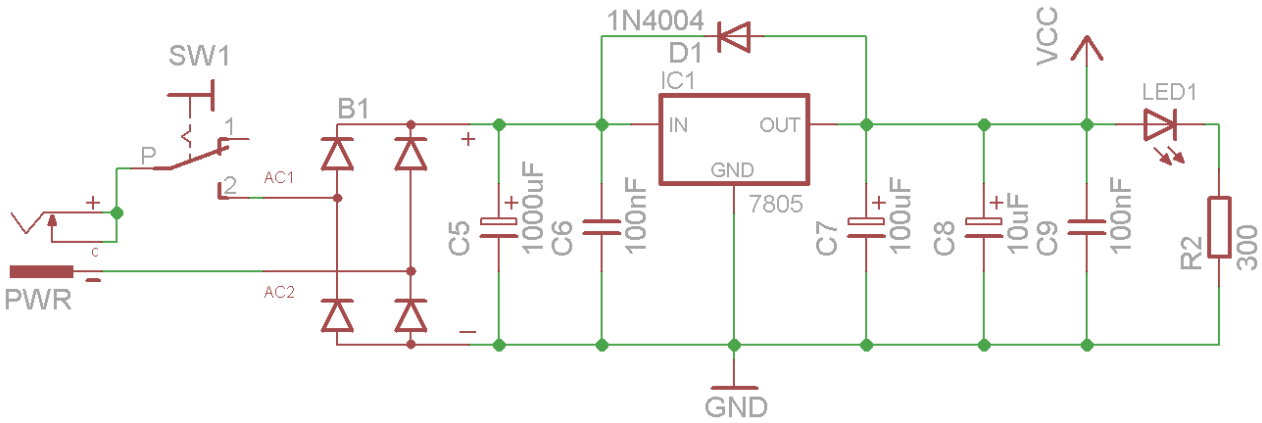
الشكل التالي يبين مخطط الدارة المطبوعة لموديول المعالج الخاص بلوحة التطوير...



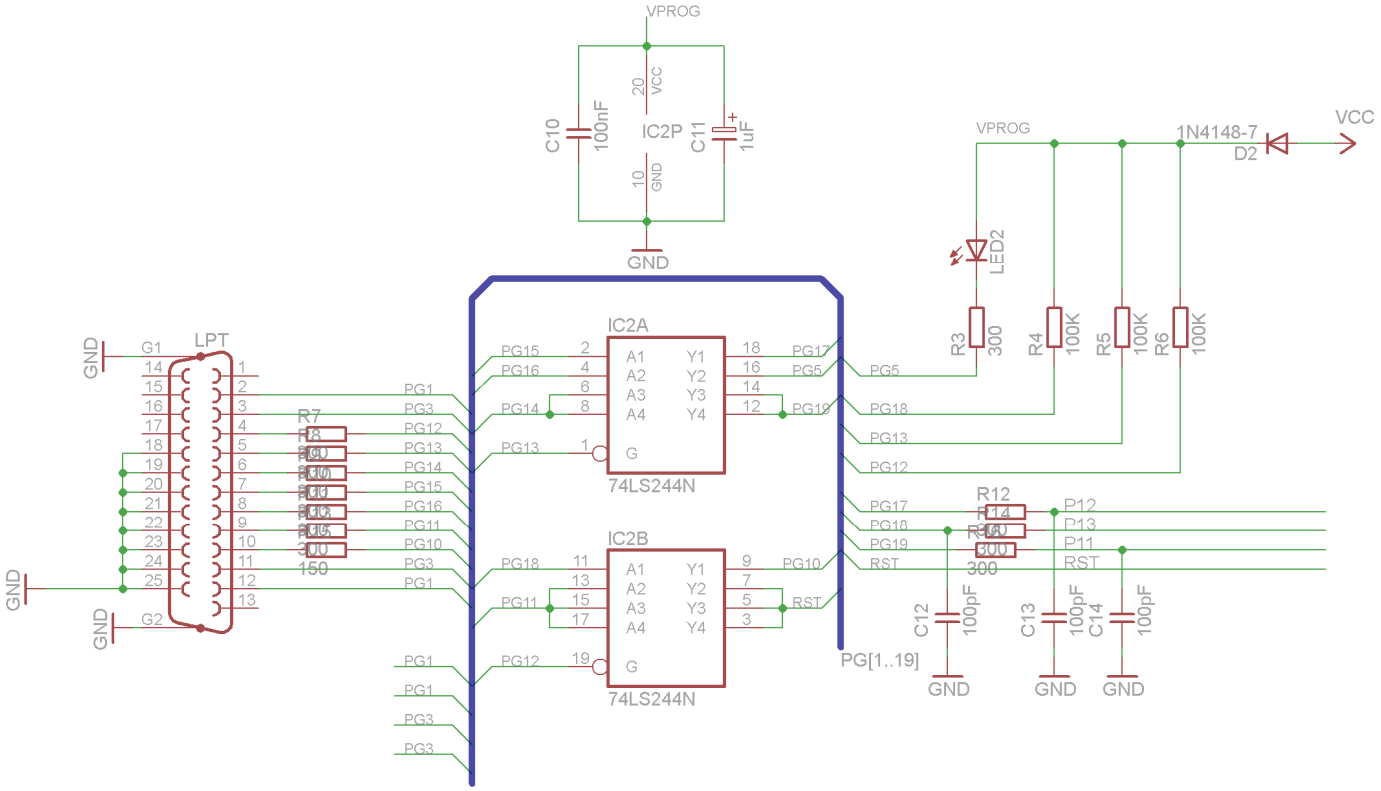
المخطط التصميمي لقاعدة موديول المعالج في لوحة التطوير...



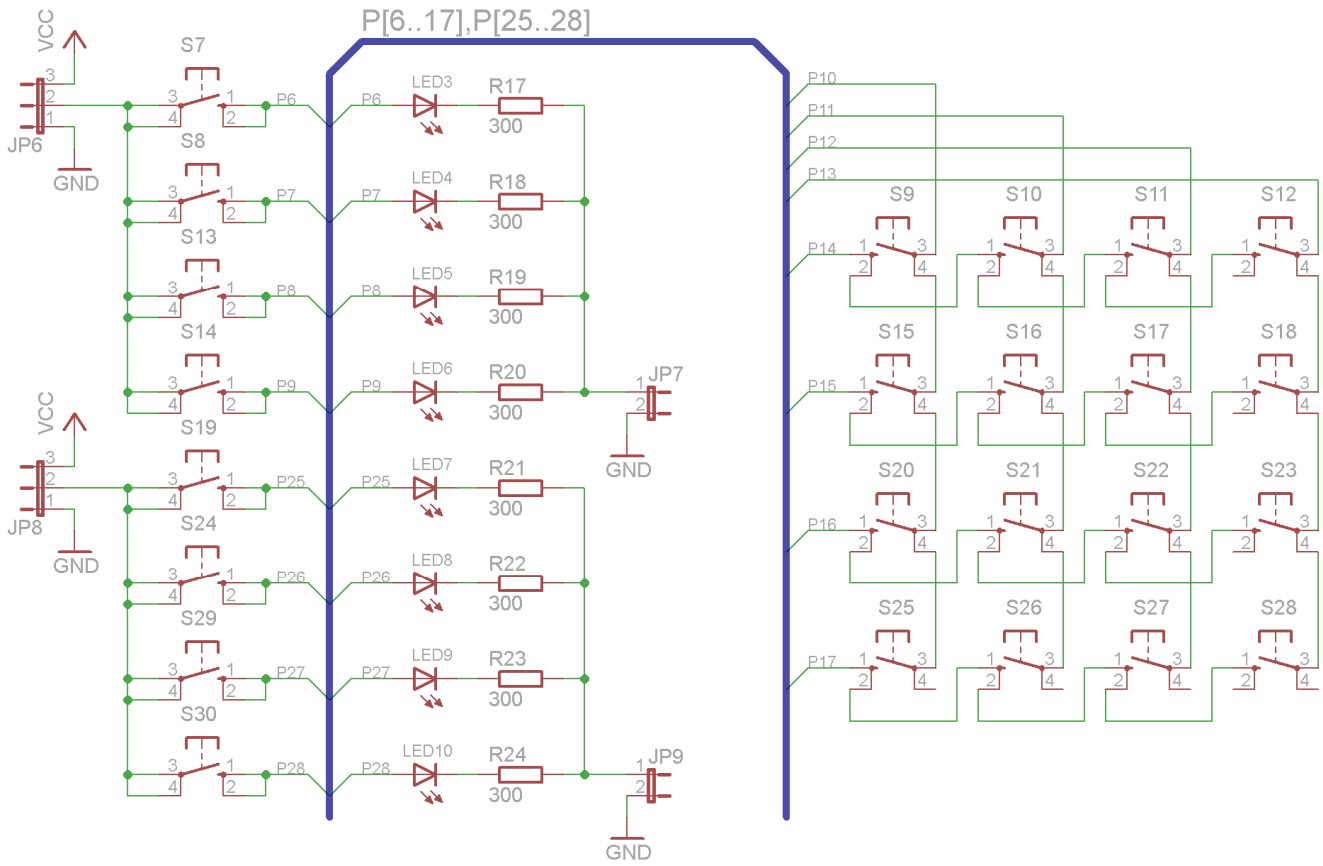
المخطط التصميمي لوحدة التغذية الرئيسية في لوحة التطوير...



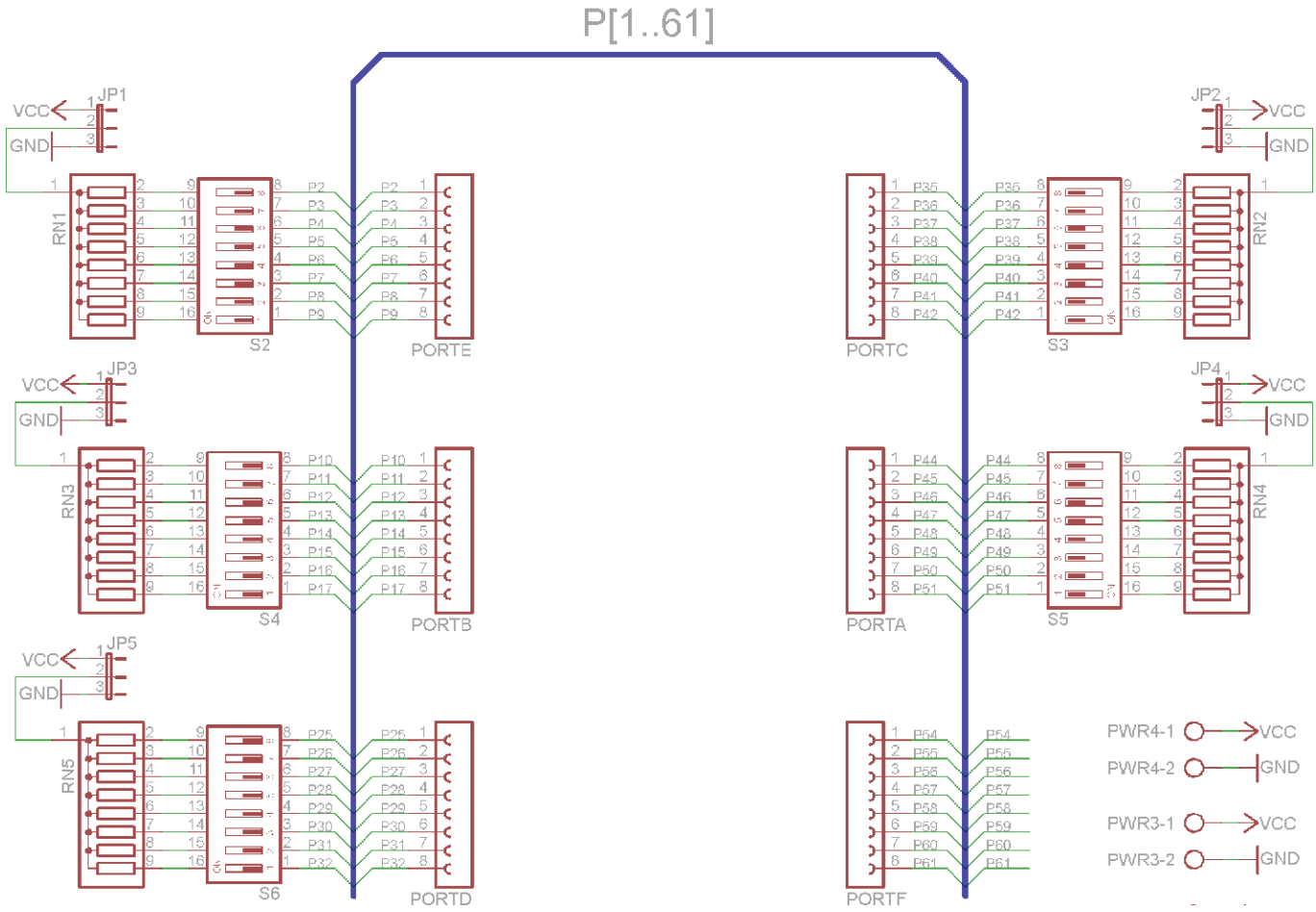
المخطط التصميمي لدارة المبرمجة في لوحة التطوير...



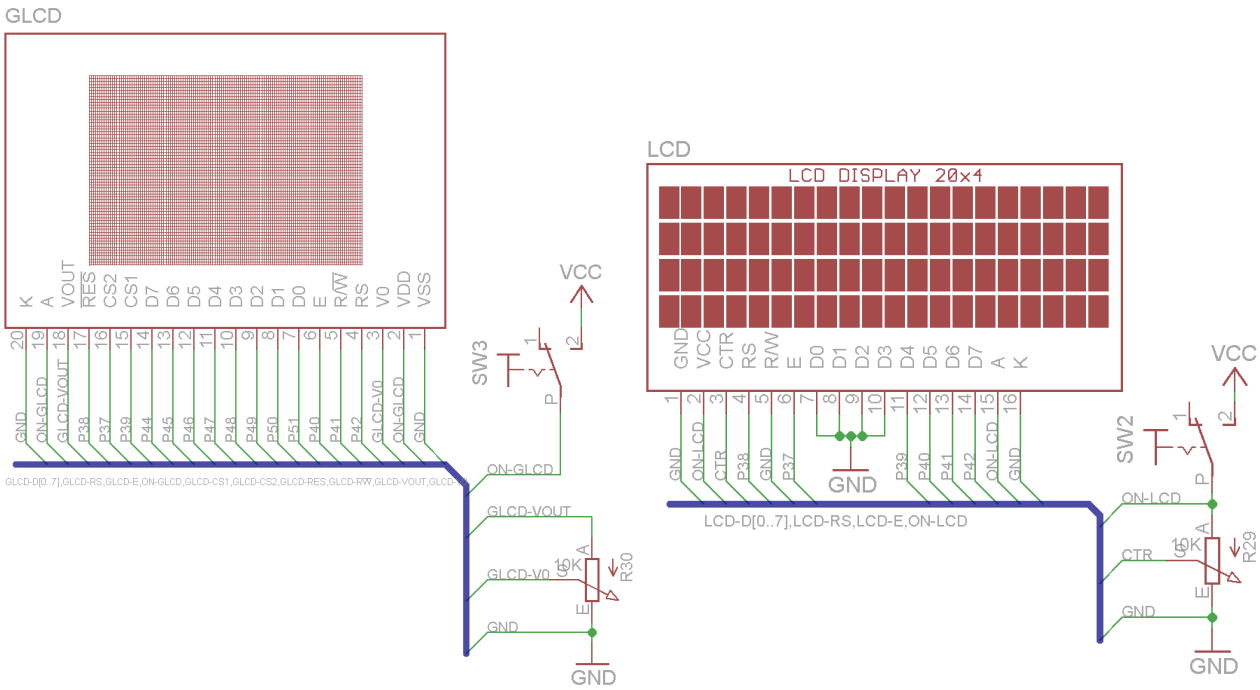
المخطط التصميمي للمفاتيح اللحظية والثنائيات الضوئية في لوحة التطوير...



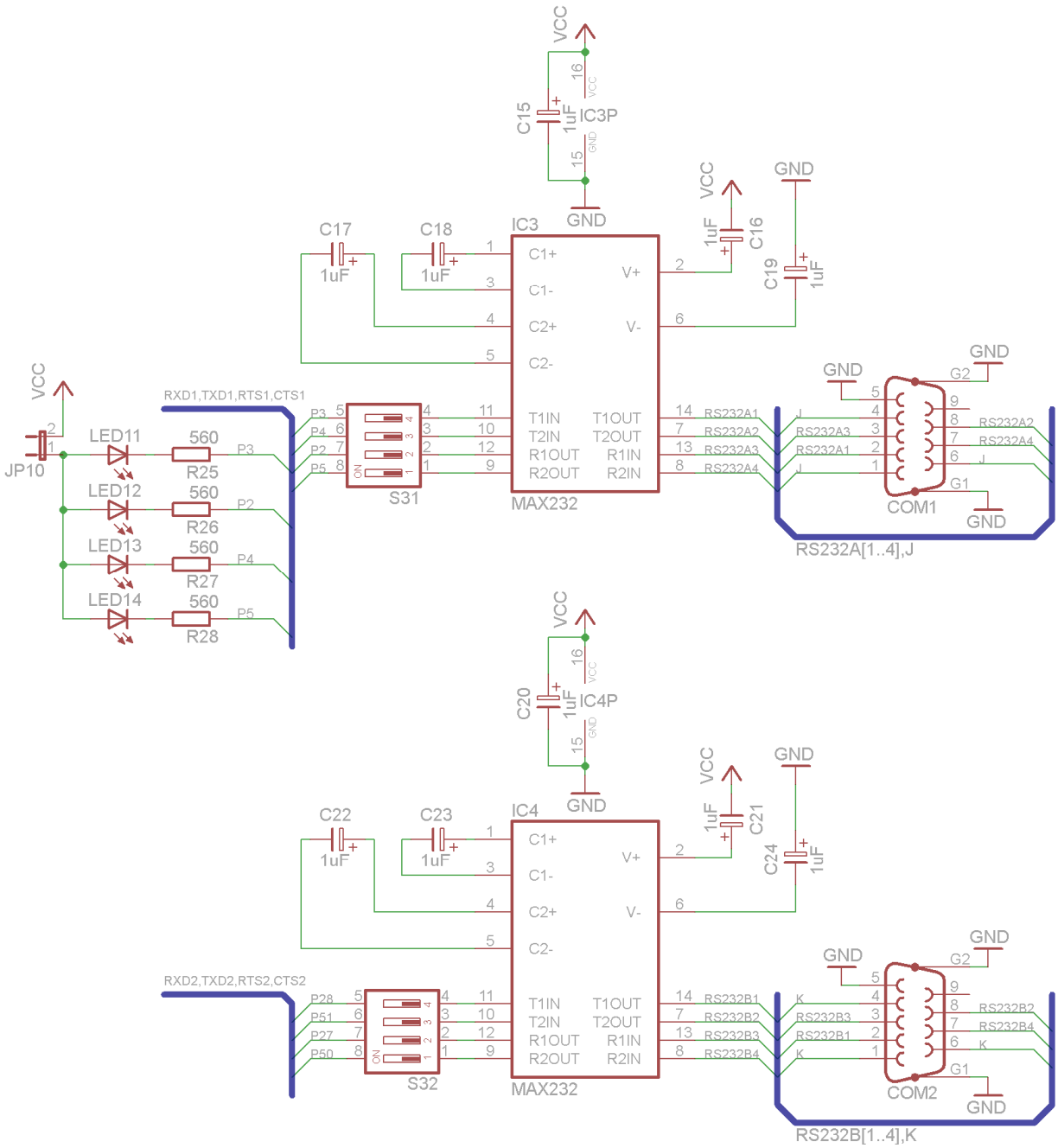
المخطط التصميمي للتوسعات الرئيسية في لوحة التطوير...



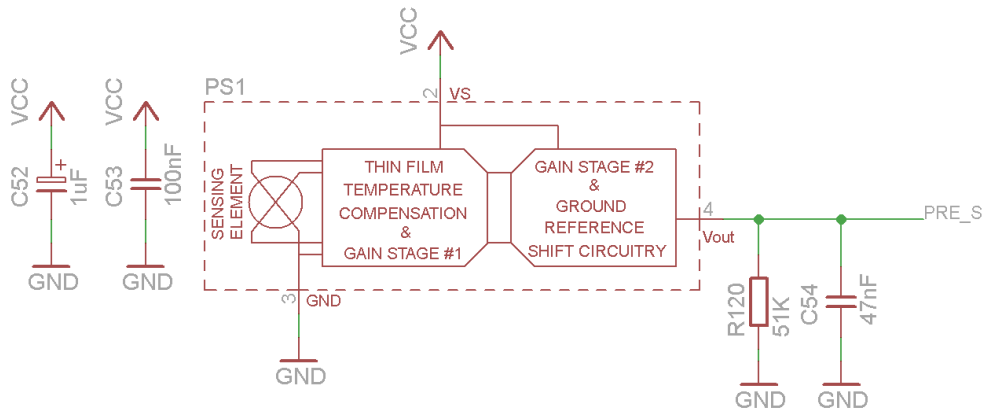
المخطط التصميمي لشاشتي الإظهار الرسومية والمحرفية في لوحة التطوير...



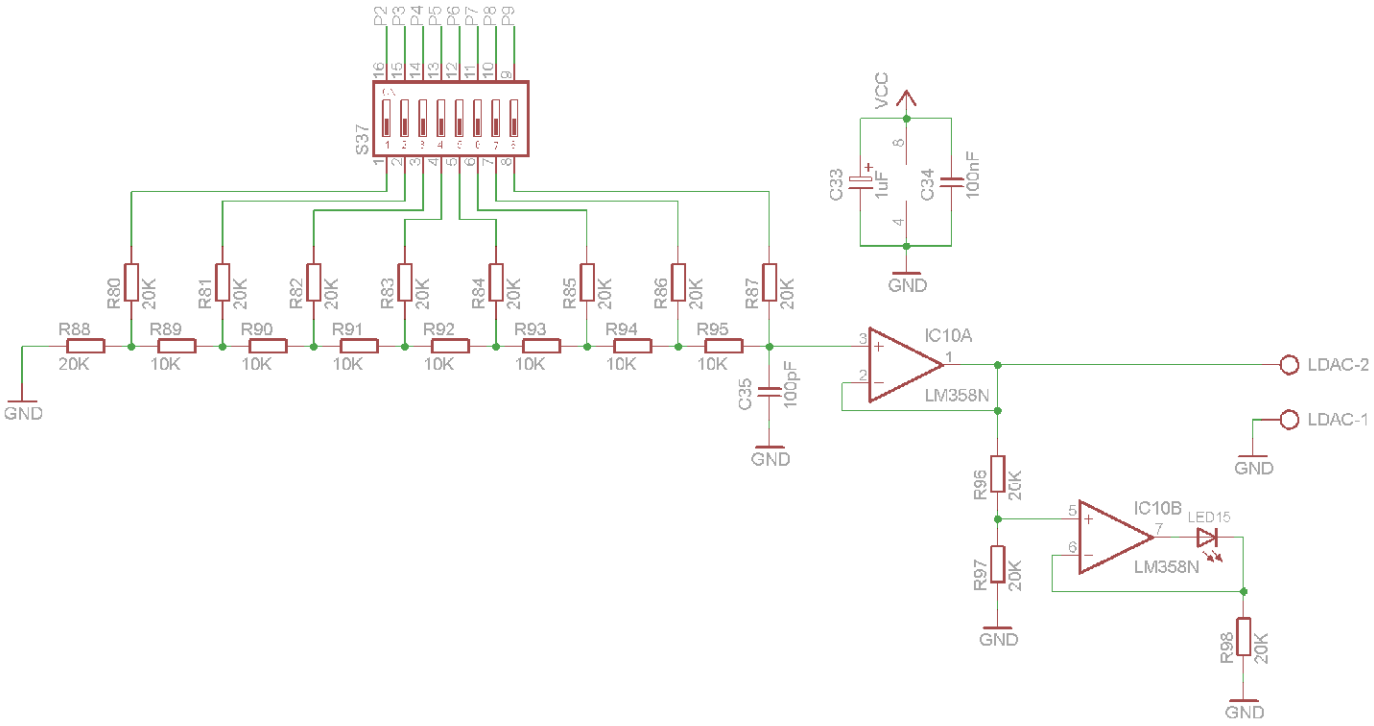
المخطط التصميمي لنافذتي الاتصال التسلسلي USART في لوحة التطوير...



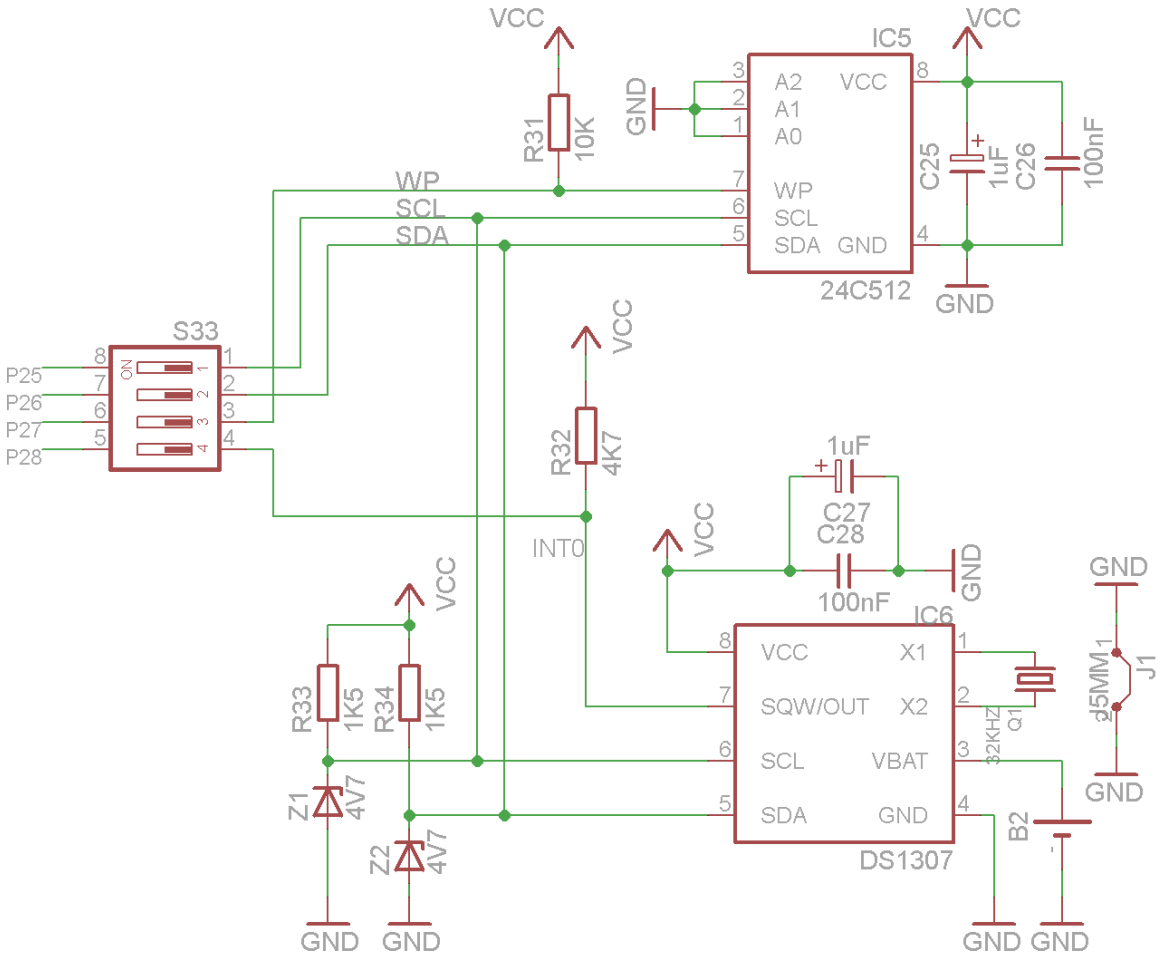
المخطط التصميمي لحساس قياس الضغط الجوي في لوحة التطوير...



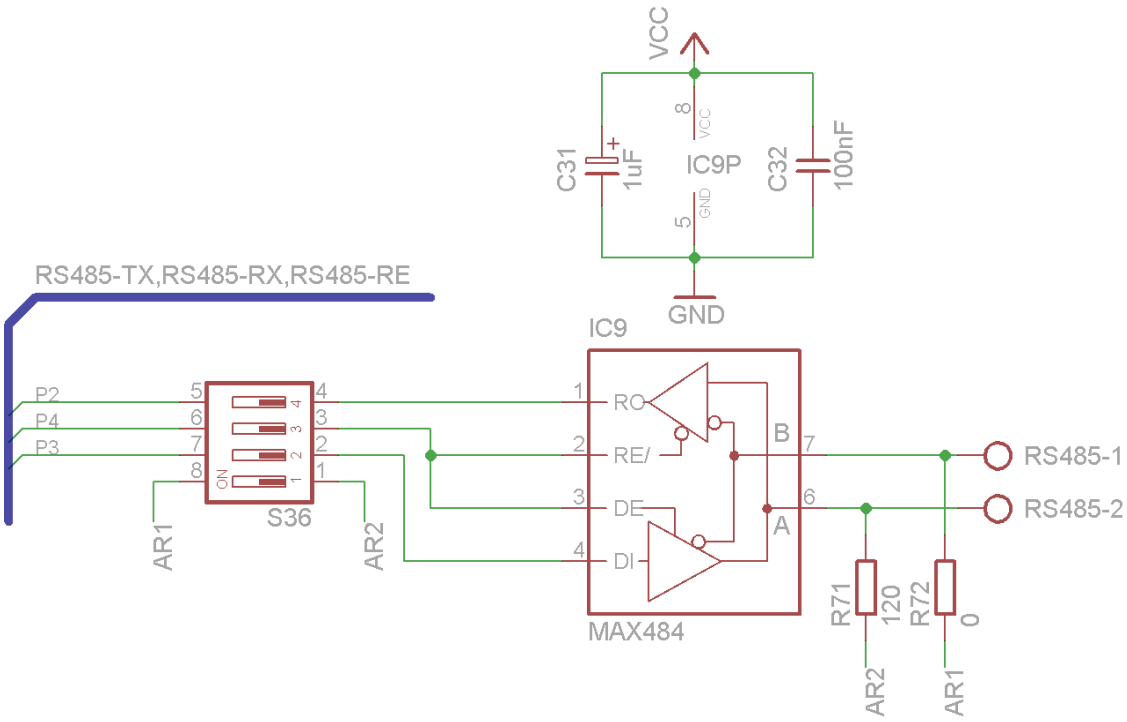
المخطط التصميمي لشبكة لادر في لوحة التطوير...



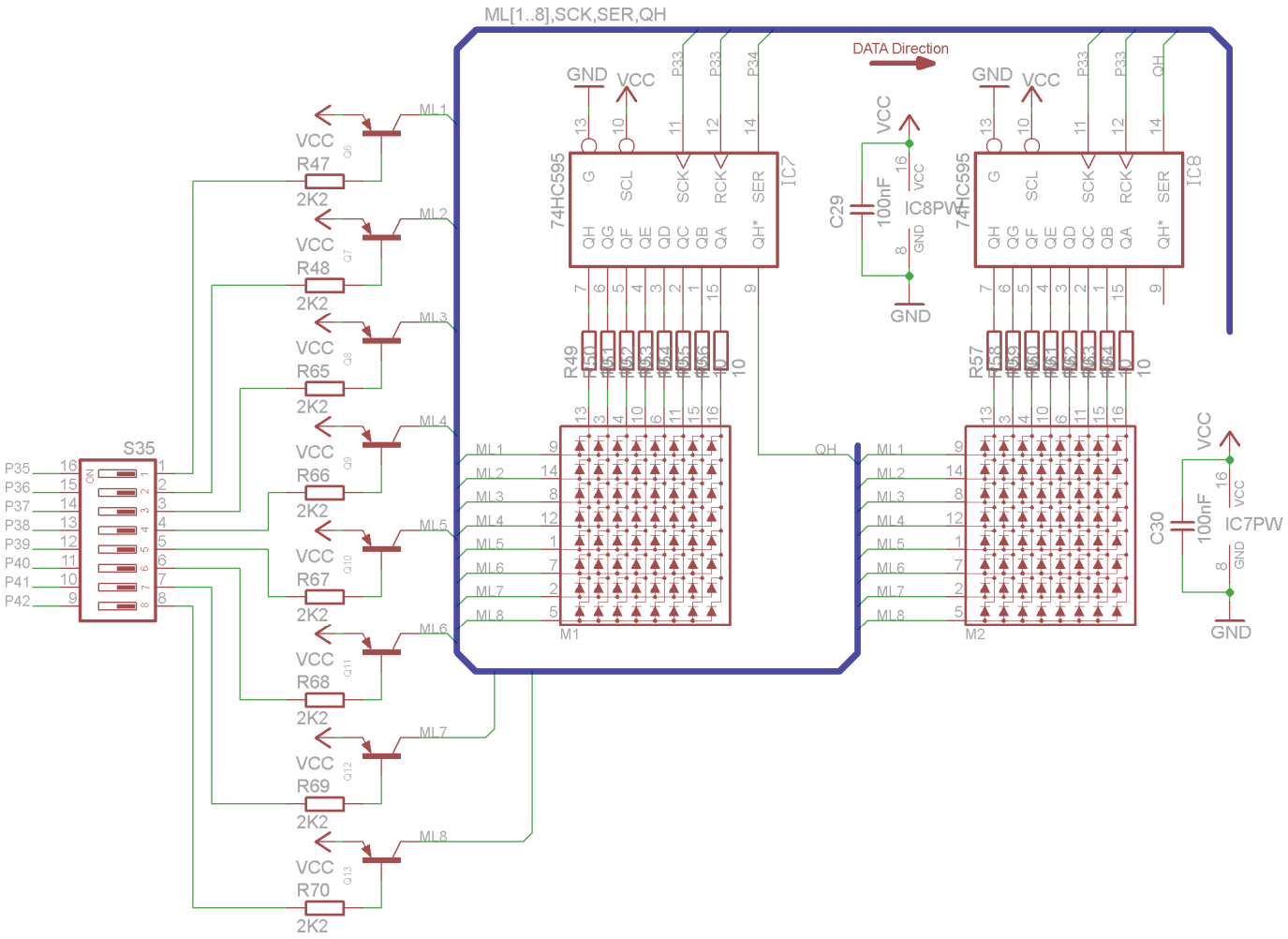
المخطط التصميمي لنافذة الربط I2C والتي تحوي EEPROM & RTC في لوحة التطوير...



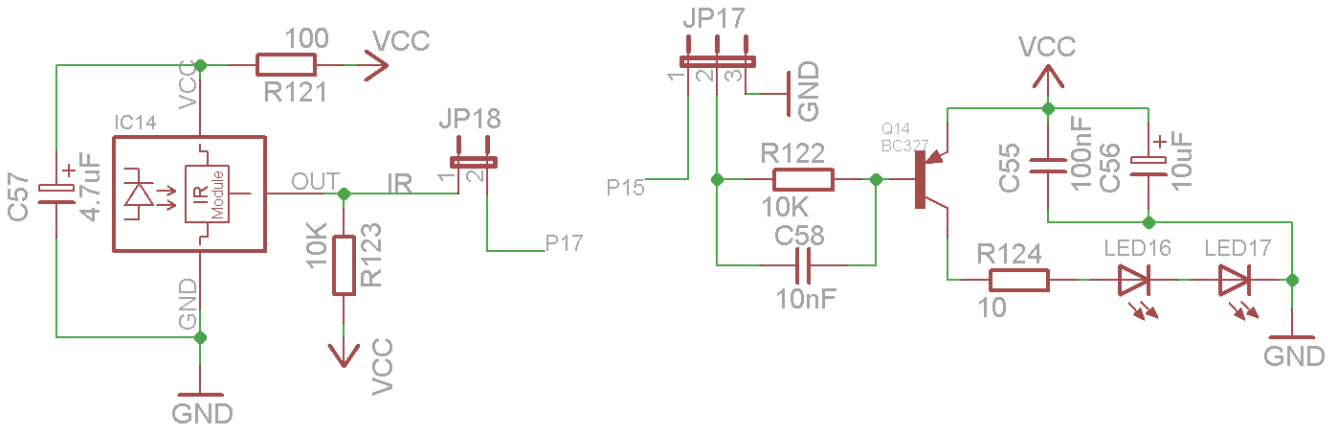
المخطط التصميمي لدارة الإرسال التسلسلي RS485 في لوحة التطوير...



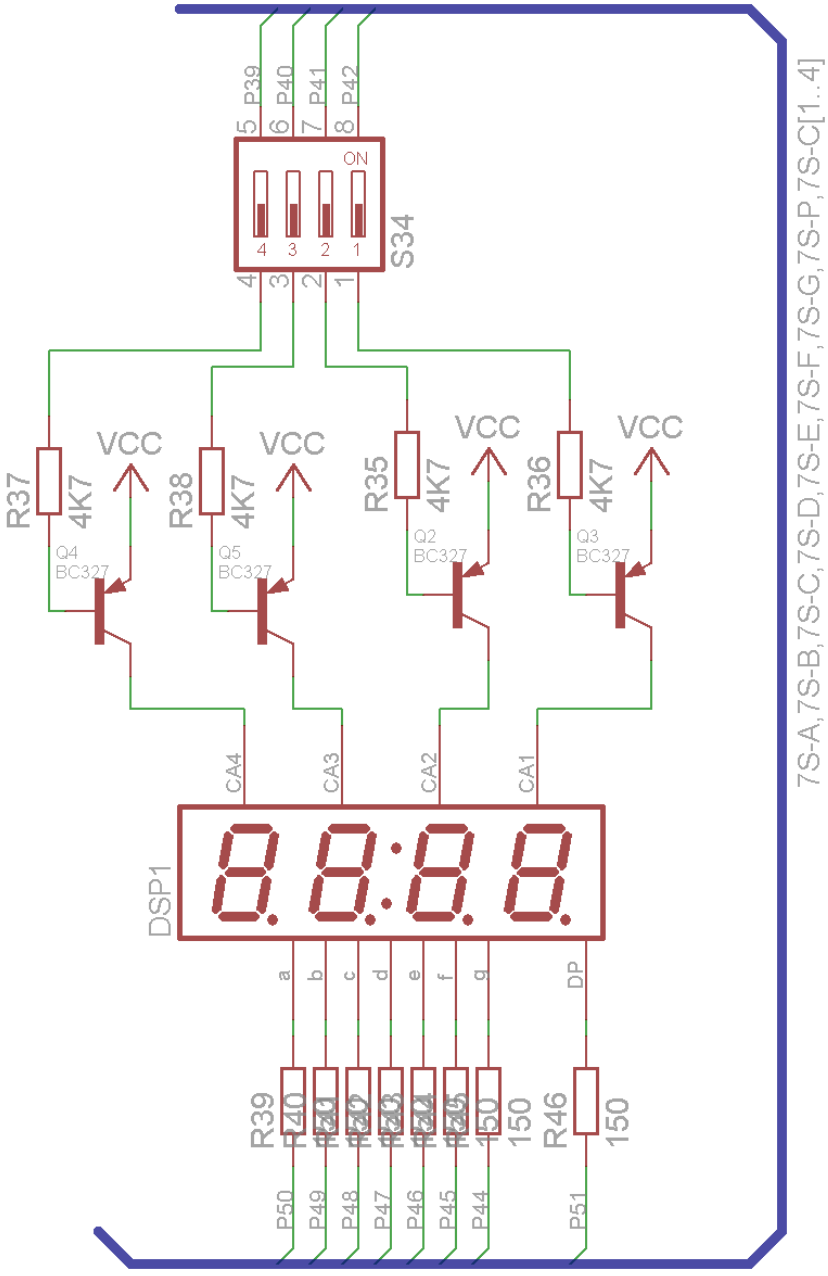
المخطط التصميمي لماتريكسات الجريدة الإلكترونية في لوحة التطوير...



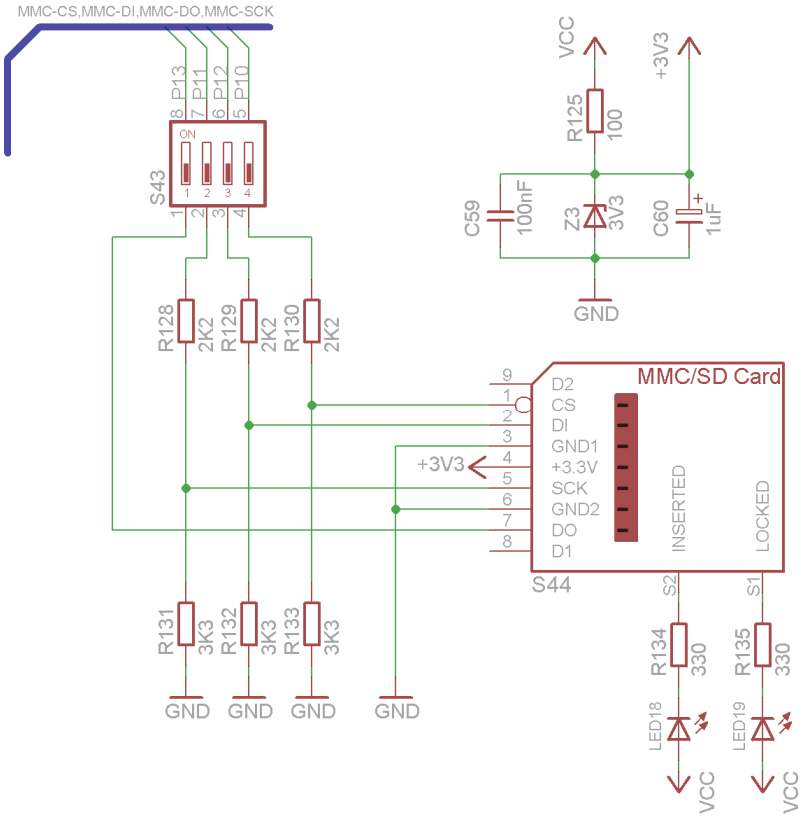
المخطط التصميمي لوحدتي الإرسال والاستقبال بالأشعة تحت الحمراء في لوحة التطوير...



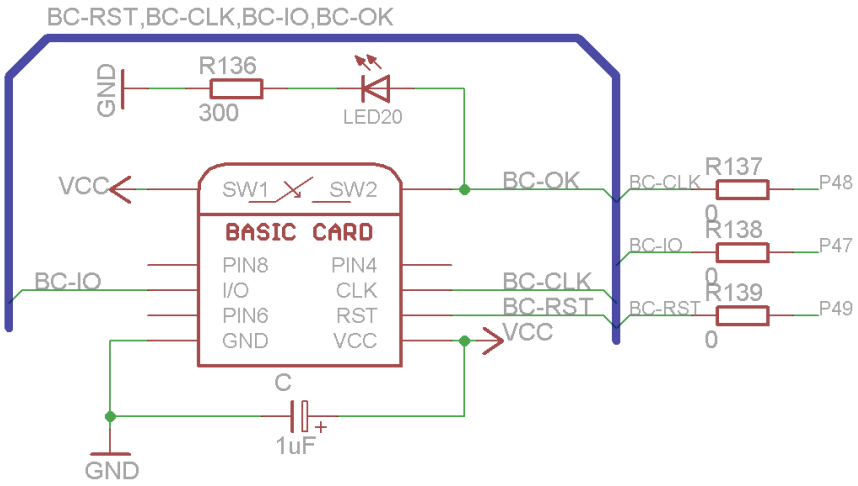
المخطط التصميمي للوحات الإظهار السباعية في لوحة التطوير...



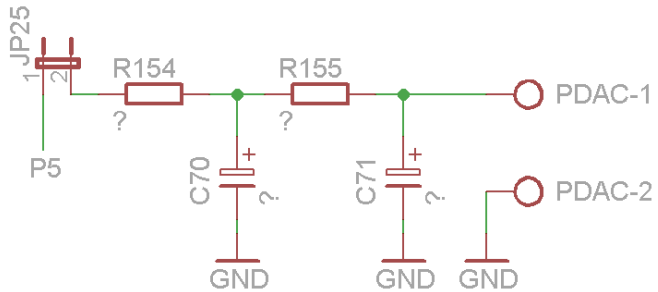
المخطط التصميمي لوحدة الربط مع شريحة ذاكرة MMC/SD في لوحة التطوير...



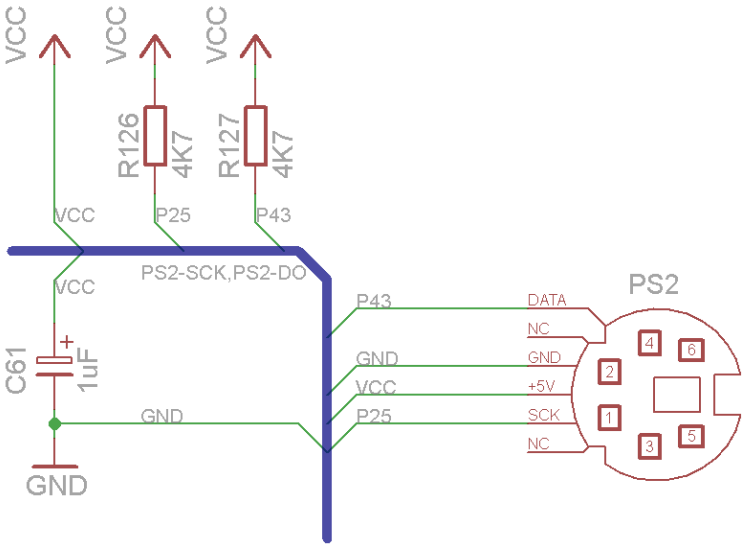
المخطط التصميمي لدارة مدخل البطاقات الذكية في لوحة التطوير...



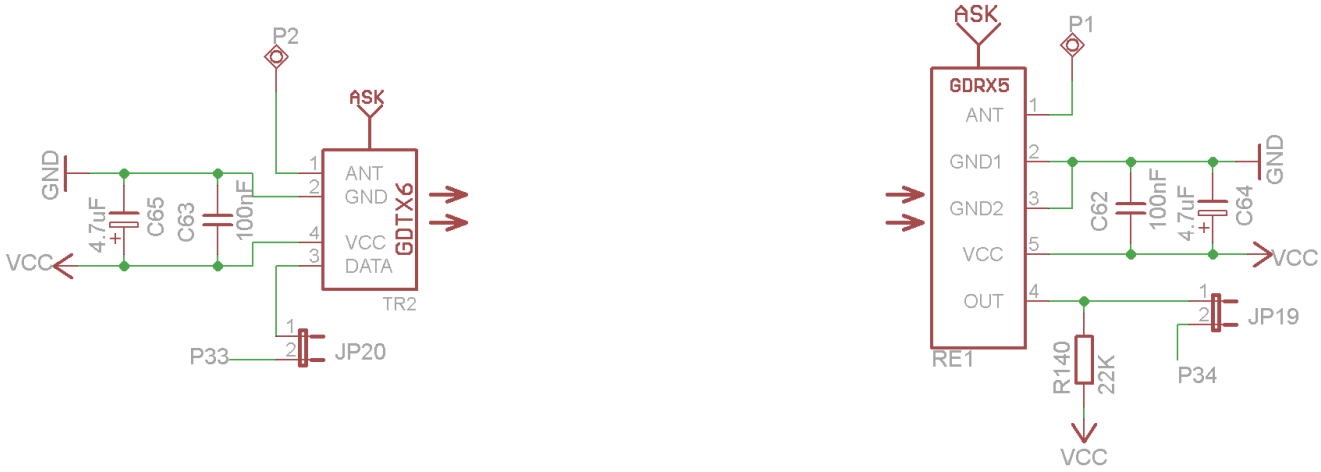
المخطط التصميمي لدارة التحويل الرقمي التشابهي باستخدام PWM في لوحة التطوير...



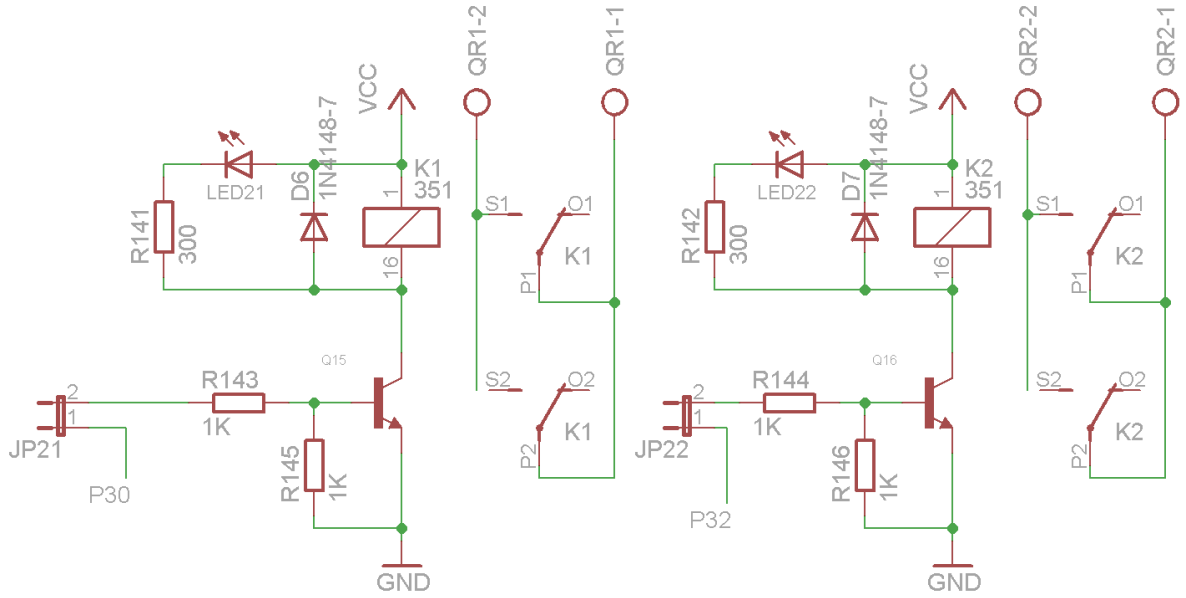
المخطط التصميمي لنافذة الاتصال PS2 في لوحة التطوير...



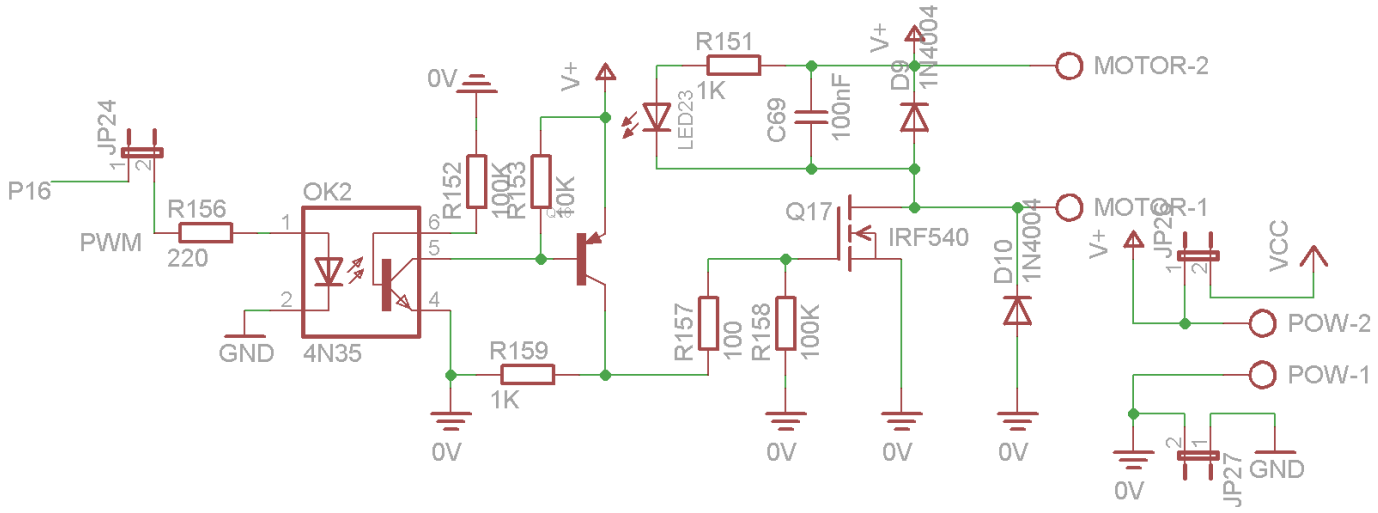
المخطط التصميمي لدارتي الإرسال والاستقبال الراديوي في لوحة التطوير...



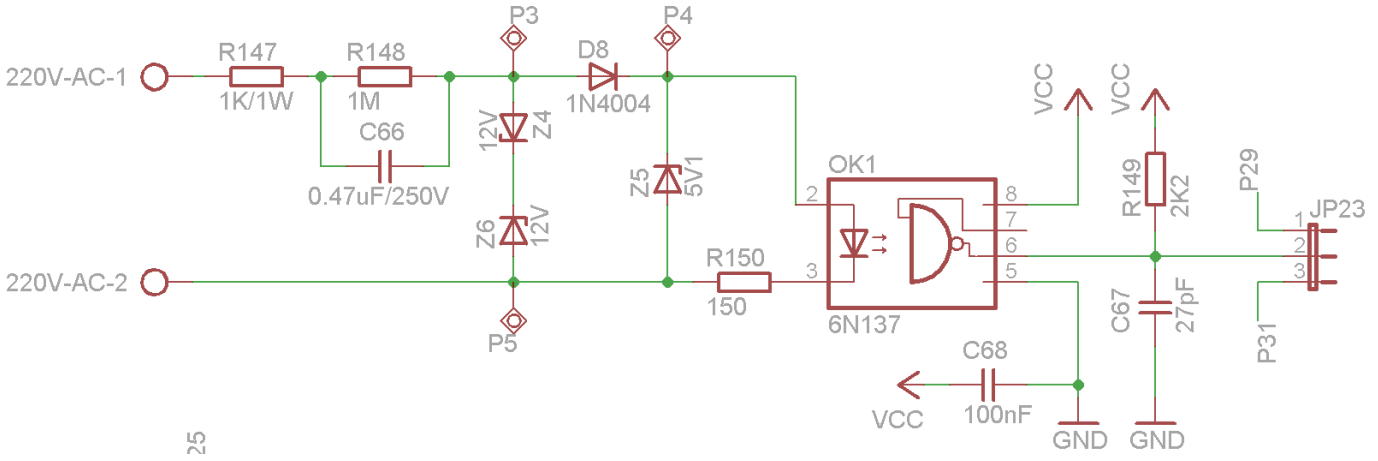
المخطط التصميمي لدارتي الخرج من نوع ريليه في لوحة التطوير...



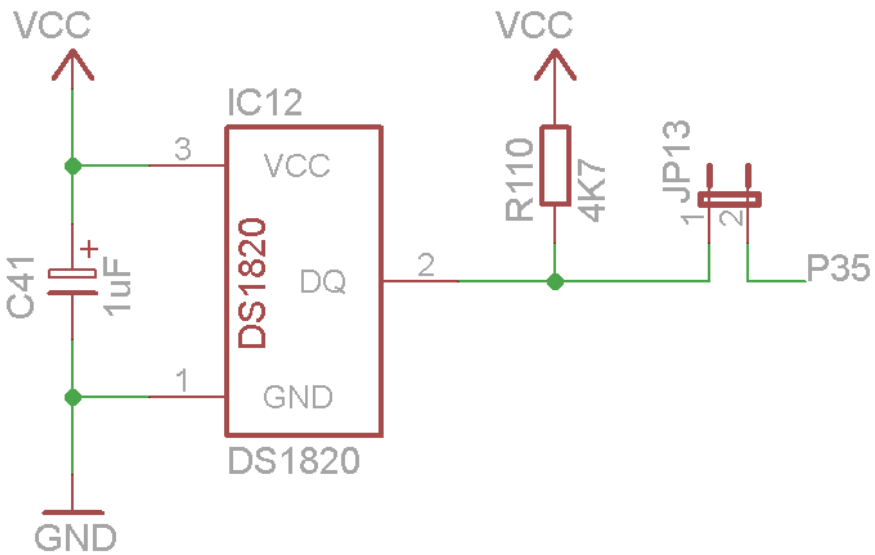
المخطط التصميمي للدارة الاستطاعية للتحكم بمحرك تيار مستمر في لوحة التطوير...



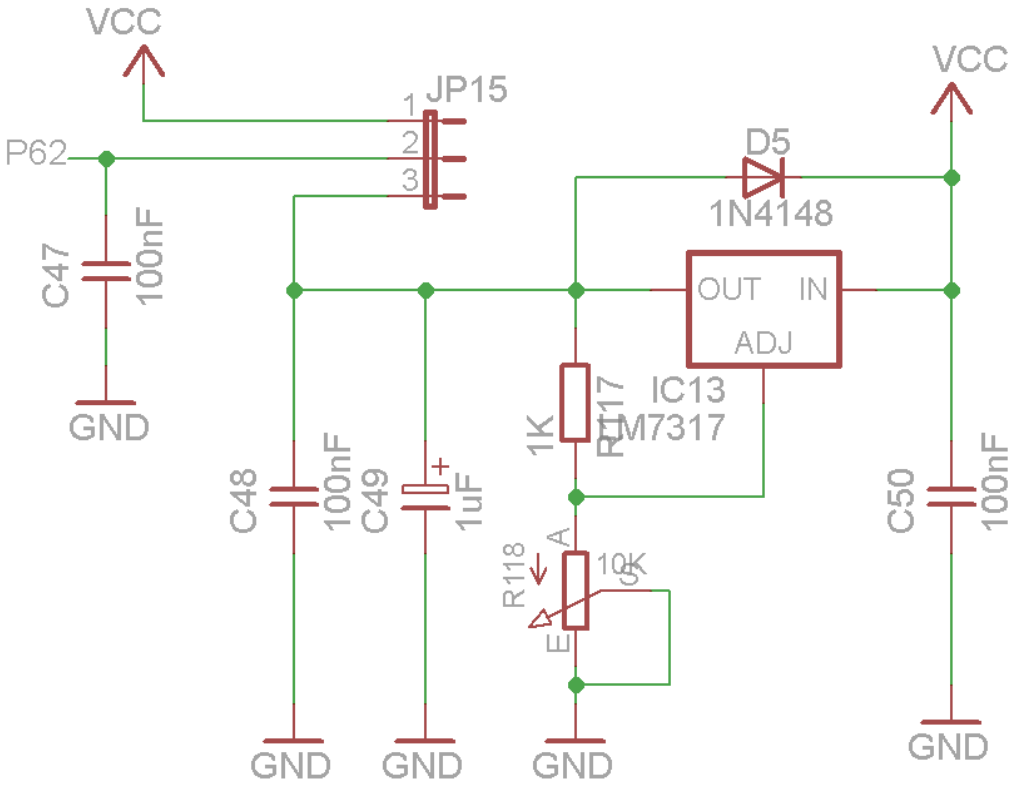
المخطط التصميمي لدارة مدخل الجهود العالية المستخدمة لقياس الإشارات في لوحة التطوير...



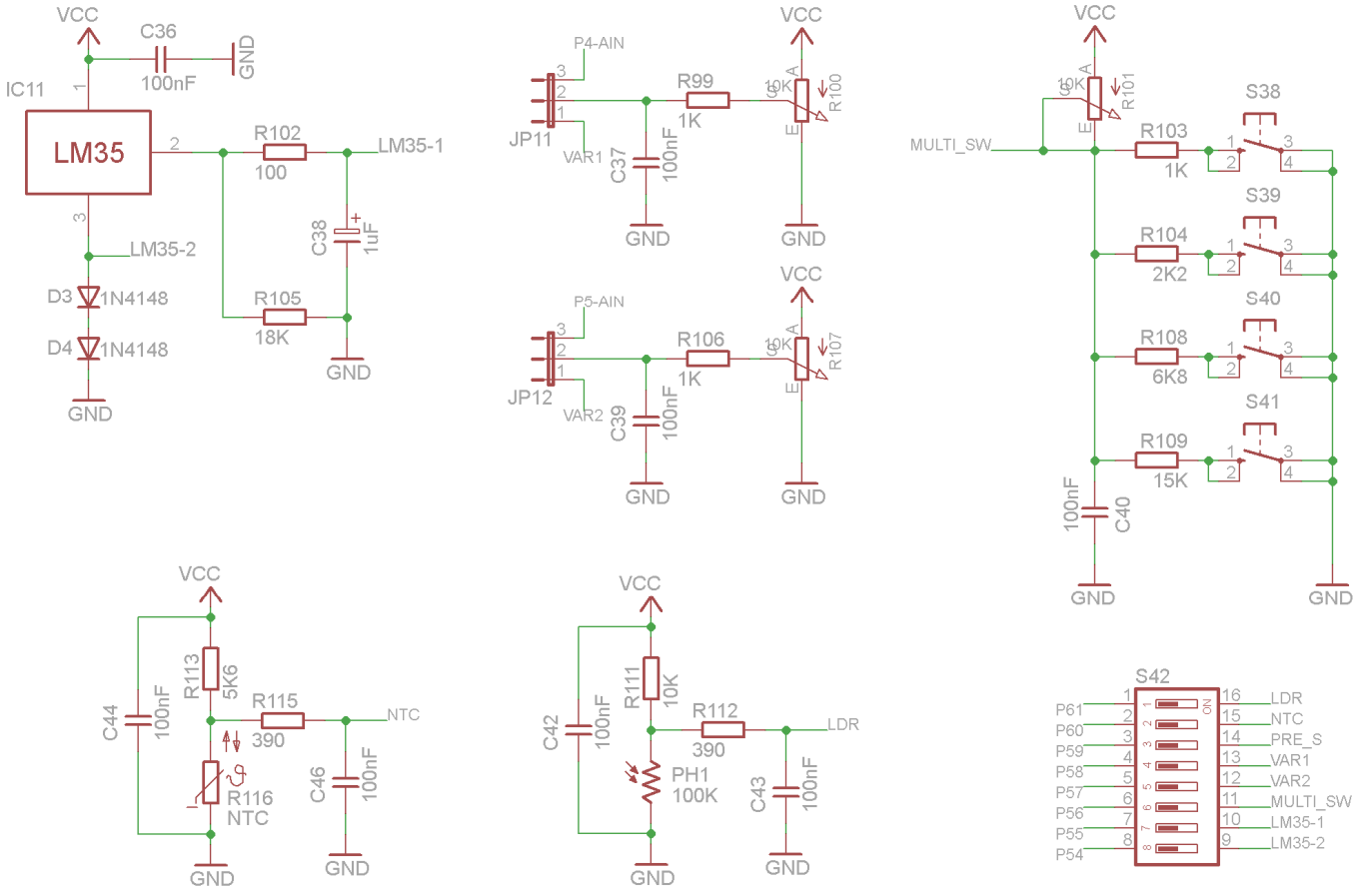
المخطط التصميمي لدارة مقياس الحرارة الرقمي باستخدام المعيار 1-Wire في لوحة التطوير...



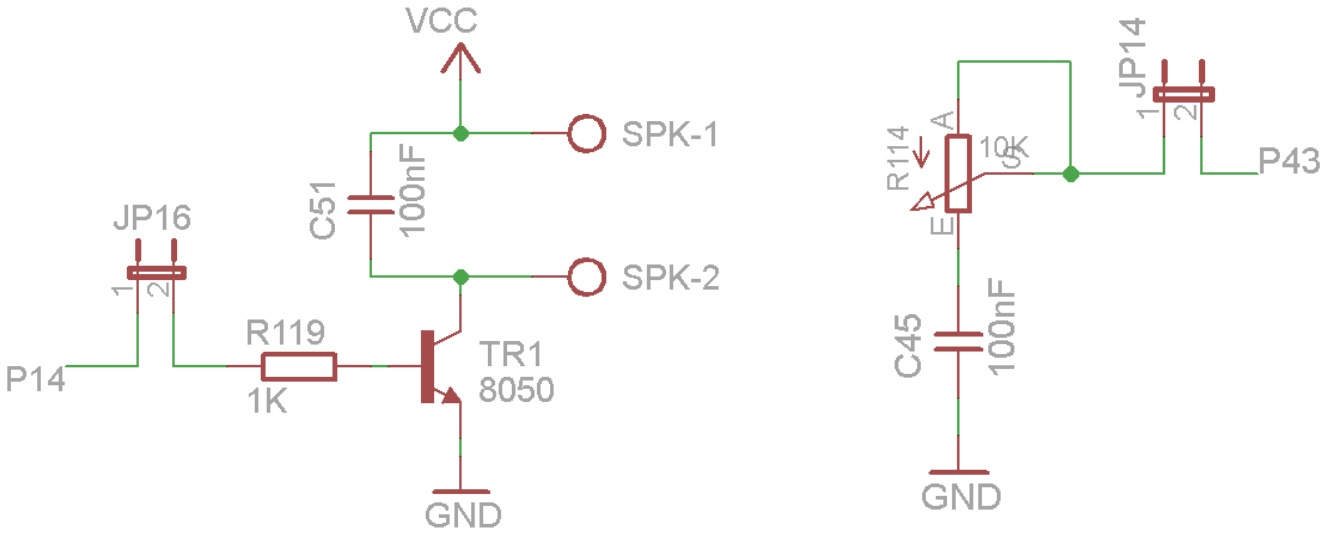
المخطط التصميمي لدارة الجهد المرجعي القبل للتعديل من أجل المبدلات التشابهية الرقمية في لوحة التطوير...



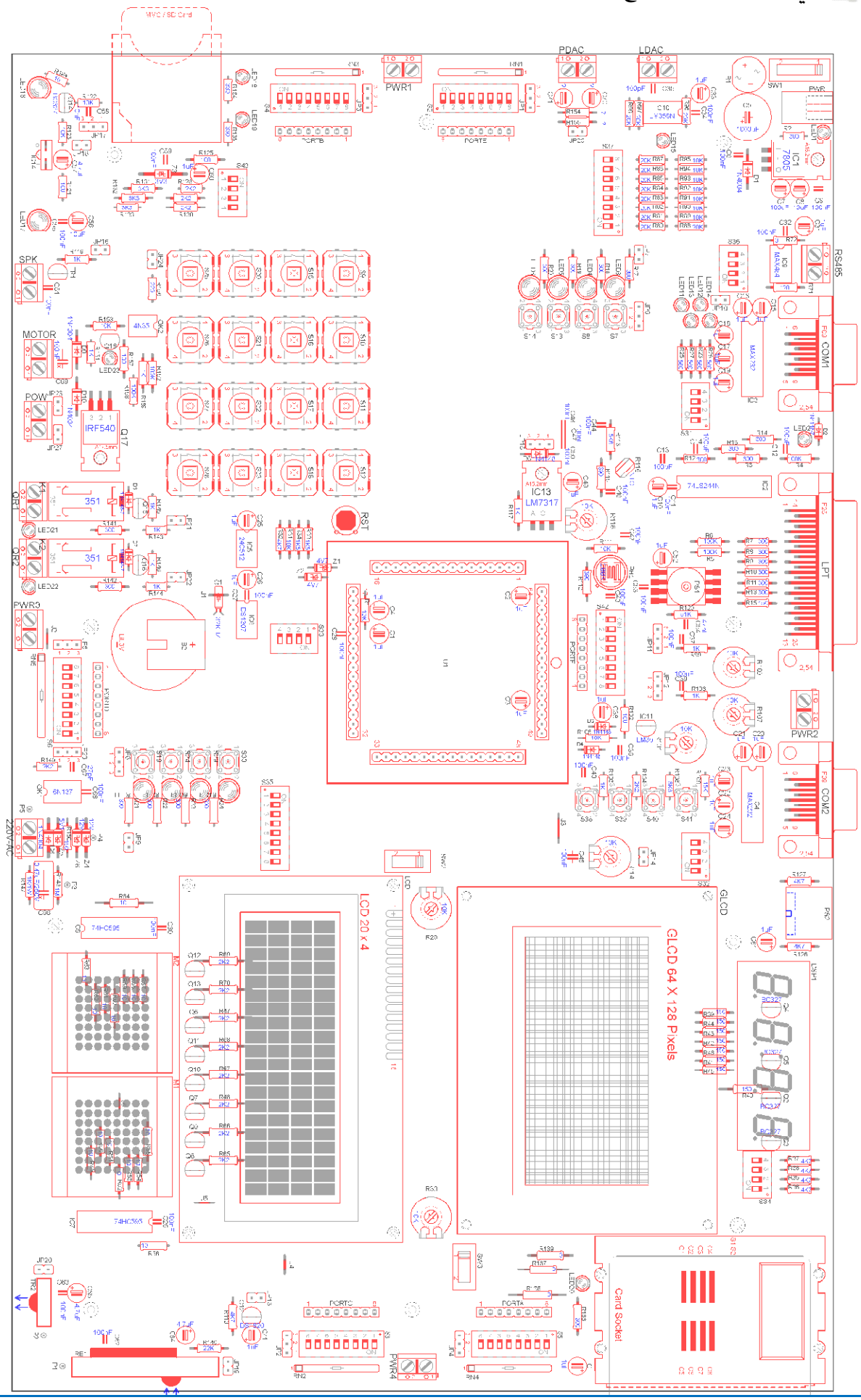
المخطط التصميمي لدارات الحساسات الموصلة مع أقطاب المبدلات التشابهية الرقمية في لوحة التطوير...



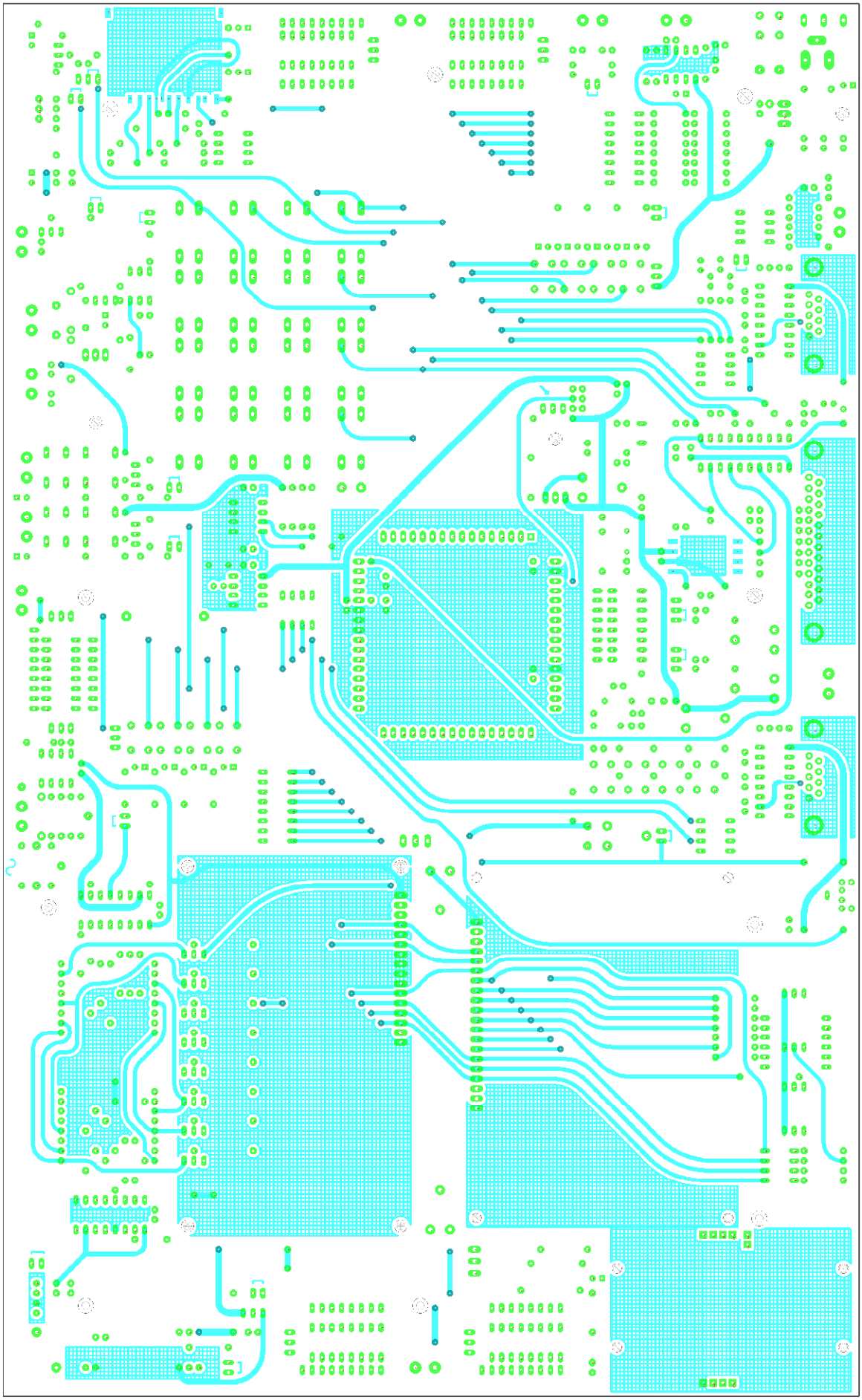
المخطط التصميمي لدارة قياس السعات والمقاومات ودارة مخرج المجهار الصوتي في لوحة التطوير...



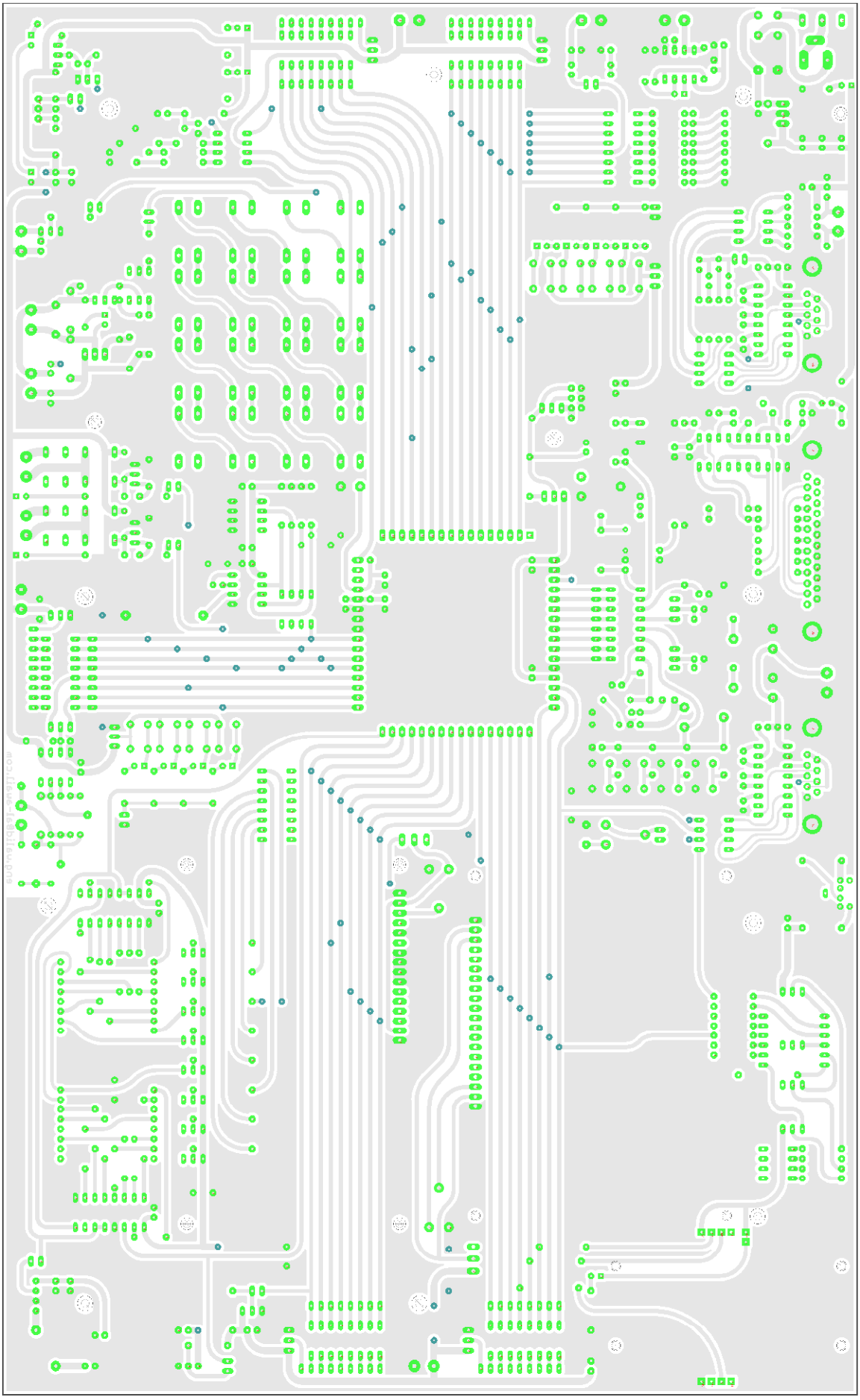
الشكل التالي يبين مخطط توضع العناصر على الدارة المطبوعة للوحة التطوير...



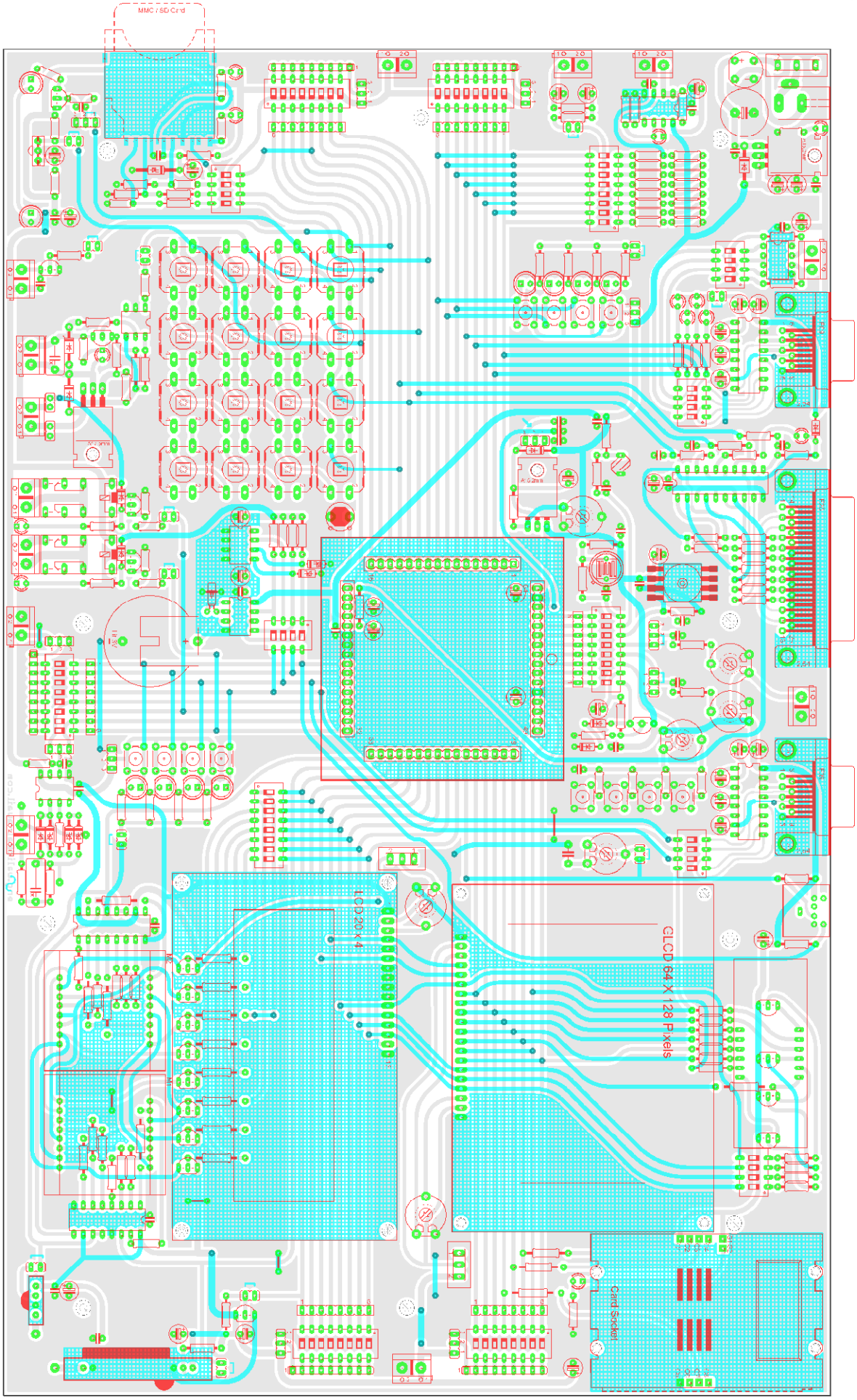
الشكل التالي يبين مخطط الطبقة السطحية للدارة المطبوعة للوحة التطوير...



الشكل التالي يبين مخطط الطبقة السفلية للدارة المطبوعة للوحة التطوير...



الشكل التالي يبين مخطط الطبقات الثلاث للدارة المطبوعة للوحة التطوير...



الجدول التالي يبين توزع الوظائف على أقطاب المعالج...

		Px.0	Px.1	Px.2	Px.3	Px.4	Px.5	Px.6	Px.7
Port E	INT4~7 UART1 AIN OC3A,B,C T3 ICP3	RS485 Interface			PWM>DAC	Four Buttons/Leds[1]			
		UART1 with Hand-checking and LEDs Indicators							
		8-bit DAC Interface							
		External Port Connector for further connecting and can be set to Pull Up/Down Resistor							
Port B	SPI OC1A,B OC0,2 OC1C		Programmer						
		MMC/SD Card SPI Interface				Speaker	IR Sender	PWM	IR Receiver
		Hexadecimal Keypad							
		External Port Connector for further connecting and can be set to Pull Up/Down Resistor							
Port D	INT0~3 UART2 TWI T1~2 ICP1	Four Buttons/Leds[2]				ICP1	Relay1	T1	Relay2
		PS2 SCK		UART2					
		RTC & EEPROM							
		External Port Connector for further connecting and can be set to Pull Up/Down Resistor							
Port C	Ex.MI-H	DS1820		GLCD Control Bus					
				LCD					
						Quad Seven Segment Control Lines			
		Dual Led-Matrix Display Data Bus							
		External Port Connector for further connecting and can be set to Pull Up/Down Resistor							
Port A	Ex.MI-L	GLCD Data Bus							
		UART2 Hand-checking		Basic Card					
		Quad Seven Segment Data Bus							
		External Port Connector for further connecting and can be set to Pull Up/Down Resistor							
Port F	ADC0~7 JTAG	LDR Resistor	NTC Resistor	Pressure sensor	Variable Resistor	Variable Resistor	4 Switches On a line	Temperature Sensor	
					JTAG Interface				
Port G	TOSC WR/RD	Dual Led-Matrix Display Control Lines		RC Circuit	23KHZ Crystal		x	x	x
		ASK TR	ASK RE	PS2 Data	x	x	x	x	x

[1] Buttons for Interrupt 4~7 can be set to VCC or GND by Jumper, connected with led indicators.
[2] Buttons for Interrupt 0~3 can be set to VCC or GND by Jumper, connected with led indicators.

برمجة المتحكمات المصغرة

التجارب العملية

الجلسة الثانية

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

WALID BALID, Tuesday, December 15, 2009

- تعليمات التوجيهات الأساسية:

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

- تعليمات التأخير الزمني:

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

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

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

ملاحظة: عند تعريف بوابة أو قطب منها كدخل، فإنه يجب تحديد الحالة الابتدائية للقيمة على البوابة وإلا ستكون حالة ممانعة عالية على قطب البوابة. إن القيمة الابتدائية يمكن تحديدها من خلال تفعيل مقاومة الرفع الداخلية لقطب الدخل أو إلغائها.

- تعليمات التعامل على مستوى البت (Set/Reset):

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

- التعليمات الشرطية:

وظيفة التعليمة	شكل التعليمة

اختبار **حالة أو قيمة** متحول وتنفيذ تعليمات معينة تبعاً لنتيجة شروط الاختبار.
إذا تحقق الشرط1 فنفذ التعليمات1
وإلا إذا تحقق الشرط2 فنفذ التعليمات2
وغير ذلك نفذ التعليمات3
	`If Expression1 Then`
	`  Statements1`
	`  ...`
	`Elseif Expression2 Then`
	`  Statements2`
	`  ...`
	`Else`
	`  Statements3`
	`  ...`
	`End If`

اختبار **حالة أو قيمة** متحول وتنفيذ تعليمات معينة تبعاً لنتيجة شرط الاختبار المتحقق.
إذا كان var = Test1 فنفذ التعليمات1
إذا كان var = Test2 فنفذ التعليمات2
وغير ذلك نفذ التعليمات3
	`SELECT CASE var`
	`  Case Test1 : Statements1`
	`  Case Test2 : Statements2`
	`  Case Else : Statements3`
	`END SELECT`

- تعليمات الحلقات:

وظيفة التعليمة	شكل التعليمة

يستمر بالدوران في الحلقة وتنفيذ التعليمات الموجودة في جسم الحلقة حتى تحقق الشرط أو الخروج القسري من الحلقة.	`Do`
	`  Statements`
	`Loop [until Expression]`

تنفيذ جملة من التعليمات طالما أن الشرط محقق.	`While Condition`
	`  Statements`
	`Wend`

تنفيذ جملة من التعليمات عدداً من المرات يبدأ من القيمة Start وينتهي عند القيمة End. يمكن تحديد خطوة العد بالمتحول step.	`For Var = Start To End [step Value]`
	`  Statements`
	`Next Var`

خروج قسري من الحلقة For
خروج قسري من الحلقة Do
خروج قسري من الحلقة While
	`Exit For`
	`Exit Do`
	`Exit While`

- تعليمات تعريف المتحولات في الذاكرة SRAM:

وظيفة التعليمة	شكل التعليمة
تعريف متحول عددي نوع بت (0 or 1).	`Dim Var1 As Bit`
تعريف متحول عددي نوع بايت (0 to 255).	`Dim Var2 As Byte`
تعريف متحول عددي صحيح (-32,768 to +32,767).	`Dim Var3 As Integer`
تعريف متحول عددي نوع وورد (0 to 65535).	`Dim Var4 As Word`
تعريف متحول عددي طويل (-2147483648 to 2147483647).	`Dim Var5 As Long`
تعريف متحول عددي مؤشر (1.5 x 10^-45 to 3.4 x 10^38).	`Dim Var6 As Single`
تعريف متحول عددي مؤشر مضاعف.	`Dim Var7 As Double`
تعريف متحول نوع محرف محدد المحارف بـ (chr_num *).	`Dim Var8 As String * 1`
تعريف مصفوفة بثمان بايتات.	`Dim Array(8) As Byte`
تعريف متحول رقمي ثابت.	`Const Symbol = Numconst` `Ex. Const Pi = 3.14159265358979`
تعريف متحول محرف ثابت.	`Const Symbol = Stringconst` `Ex. Const S = "TEST"`
تعريف تعبير رياضي ثابت.	`Const Symbol = Expression` `Ex. Const E =(b1 * 3) + 2`
تعريف متحول محلي في برنامج فرعي أو برنامج فرعي وظيفي.	`Local Var As Type`

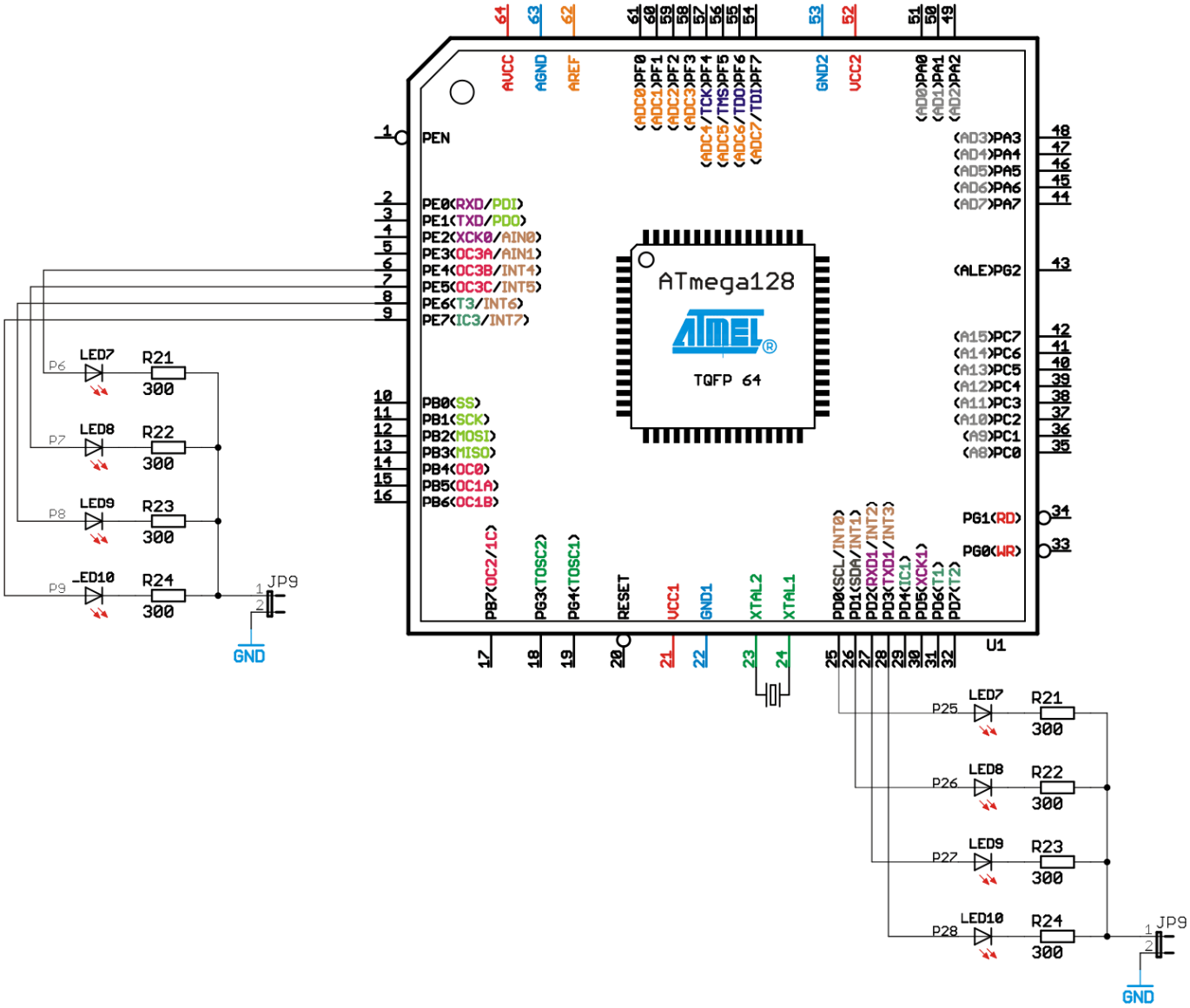
- تعليمات قراءة حالة المداخل:

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

الغاية من التجربة:

استثمار بوابات المتحكم كبوابات خرج رقمي من أجل تشكيل حركة ضوئية باستخدام الثنائيات ضوئية.

مخطط التوصيل:



متطلبات توصيل:

يجب قصر الوصلة JP9, JP10.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج من أجل تشكيل حركة للثنائيات الضوئية بحيث أنه سوف يتم تشغيلها كما يلي:

1- تشغيل LEDs7.
2- انتظار مدة تأخير زمني ثانية.
3- تشغيل الثنائي التالي... وهكذا حتى يتم تشغيل جميع الثنائيات.

4- بعدها يتم إطفاء الثنائيات الضوئية بشكل معاكس.

برنامج تشغيل الدارة:

التوجيهات.	`$regfile = "m128def.dat"` `$crystal = 1000000`

تعريف الأقطاب.	`Config Portd.0 = Output`
	`Config Portd.1 = Output`
	`Config Portd.2 = Output`
	`Config Portd.3 = Output`
	
	`Config Porte.4 = Output`
	`Config Porte.5 = Output`
	`Config Porte.6 = Output`
	`Config Porte.7 = Output`

الإشارة إلى الأسماء الفيزيائية للبوابات أثناء البرنامج بأسماء حقيقية.	`Leds1 Alias Portd` `Leds2 Alias Porte`
تعريف المتحولات.	`Dim I As Byte , J As Byte`

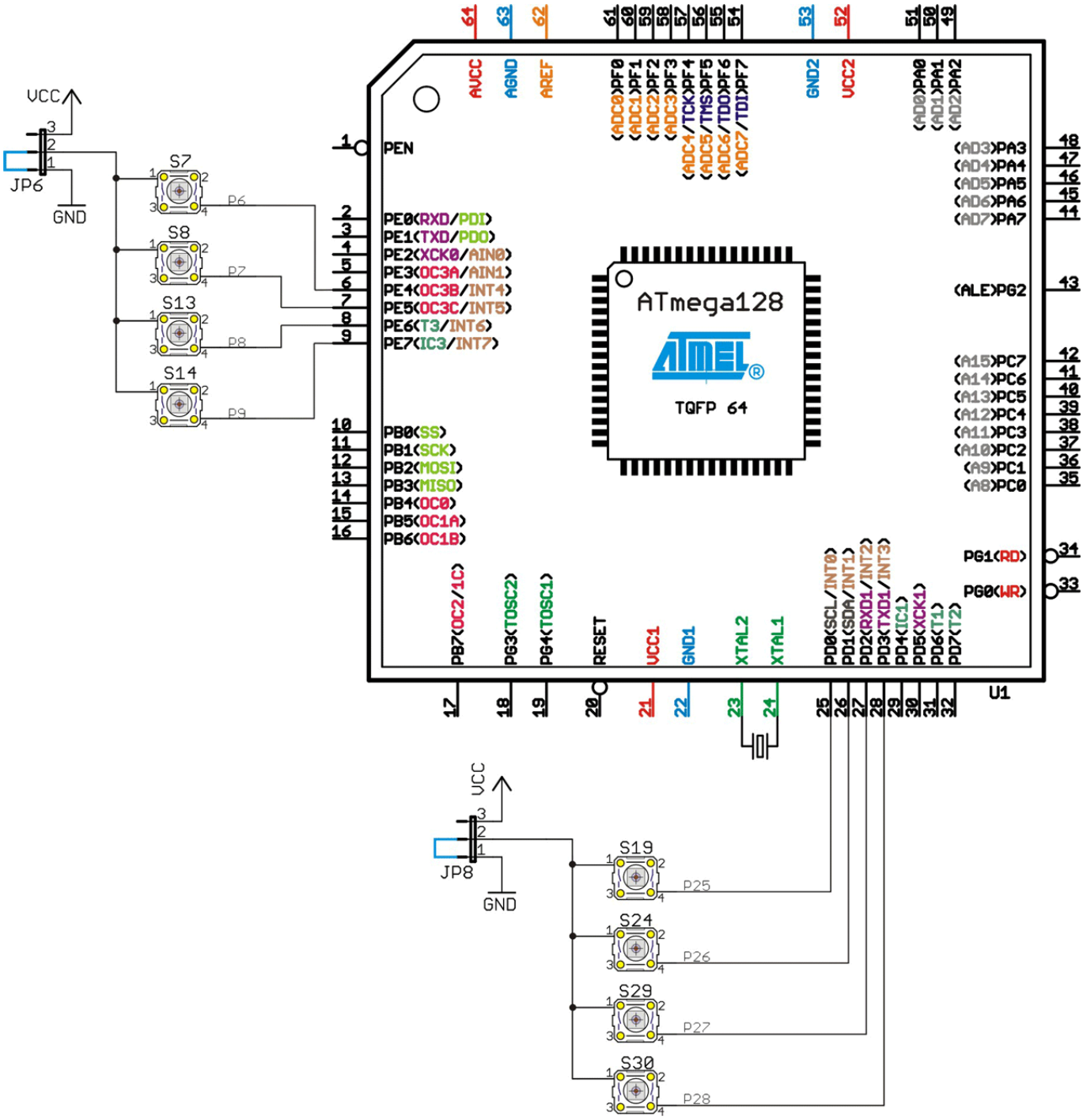
حلقة تشغيل الثنائيات الضوئية بالتتالي وبفواصل زمنية واحد ثانية.	`Do`
	`For I = 0 To 3`
	`J = I + 4`
	`Set Leds1.i`
	`Set Leds2.j`
	`Wait 1`
	`Next I`

حلقة إطفاء الثنائيات الضوئية عكسياً وبالتتالي وبفواصل زمنية واحد ثانية.	`For I = 3 To 0 Step -1`
	`J = I + 4`
	`Reset Leds1.i`
	`Reset Leds2.j`
	`Wait 1`
	`Next I`
	`Loop`
	`End`

الغاية من التجربة:

استثمار أقطاب المتحكم كأقطاب دخل رقمي من أجل قراءة حالة مفاتيح لحظية.

مخطط التوصيل:



متطلبات توصيل:

يجب قصر الوصلة JP6, JP8.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج من أجل قراءة حالة المفاتيح اللحظية في حالتين:

1- الوصلات JP6, JP8 موصلة إلى GND (قراءة الجبهة الهابطة على القطب).

2- الوصلات JP8 موصلة إلى GND, JP6 موصلة إلى VCC.

برنامج تشغيل الدارة في حالة JP6, JP8 موصلة إلى GND:

التوجيهات.	`$regfile = "m128def.dat"`
	`$crystal = 1000000`
	`$baud = 4800`

تحديد التأخير الزمني للقراءة الجديدة لحالة المفتاح	`'-----------------------` `Config Debounce = 150`

تعريف الأقطاب.	`Config Pind.0 = Input`
	`Config Pind.1 = Input`
	`Config Pind.2 = Input`
	`Config Pind.3 = Input`
	
	`Config Pine.4 = Input`
	`Config Pine.5 = Input`
	`Config Pine.6 = Input`
	`Config Pine.7 = Input`

تفعيل المقاومات الداخلية	`Portd.0 = 1 : Portd.1 = 1`
	`Portd.2 = 1 : Portd.3 = 1`
	`Porte.4 = 1 : Porte.5 = 1`
	`Porte.6 = 1 : Porte.7 = 1`
	`'-----------------------`

الإشارة إلى الأسماء الفيزيائية للبوابات أثناء البرنامج بأسماء حقيقية.	`Key1 Alias Pind.0`
	`Key2 Alias Pind.1`
	`Key3 Alias Pind.2`
	`Key4 Alias Pind.3`
	`Key5 Alias Pine.4`
	`Key6 Alias Pine.5`
	`Key7 Alias Pine.6`
	`Key8 Alias Pine.7`
	`'-----------------------`

حلقة البرنامج الرئيسي
قراءة حالة المفاتيح الثمانية
	`Do`
	`  Debounce Key1 , 0 , Sw1 , Sub`
	`  Debounce Key2 , 0 , Sw2 , Sub`
	`  Debounce Key3 , 0 , Sw3 , Sub`
	`  Debounce Key4 , 0 , Sw4 , Sub`
	`  Debounce Key5 , 0 , Sw5 , Sub`
	`  Debounce Key6 , 0 , Sw6 , Sub`
	`  Debounce Key7 , 0 , Sw7 , Sub`
	`  Debounce Key8 , 0 , Sw8 , Sub`
	
	`Waitms 200`
	
	`Loop`
	`End`
	`'-----------------------`

```
Sw1:
   Print "Switch(1) is pressed!"
Return
'/------------
Sw2:
   Print "Switch(2) is pressed!"
Return
'/------------
Sw3:
   Print "Switch(3) is pressed!"
Return
'/------------
Sw4:
   Print "Switch(4) is pressed!"
Return
'/------------
Sw5:
   Print "Switch(5) is pressed!"
Return
'/------------
Sw6:
   Print "Switch(6) is pressed!"
Return
'/------------
Sw7:
   Print "Switch(7) is pressed!"
Return
'/------------
Sw8:
   Print "Switch(8) is pressed!"
Return
```

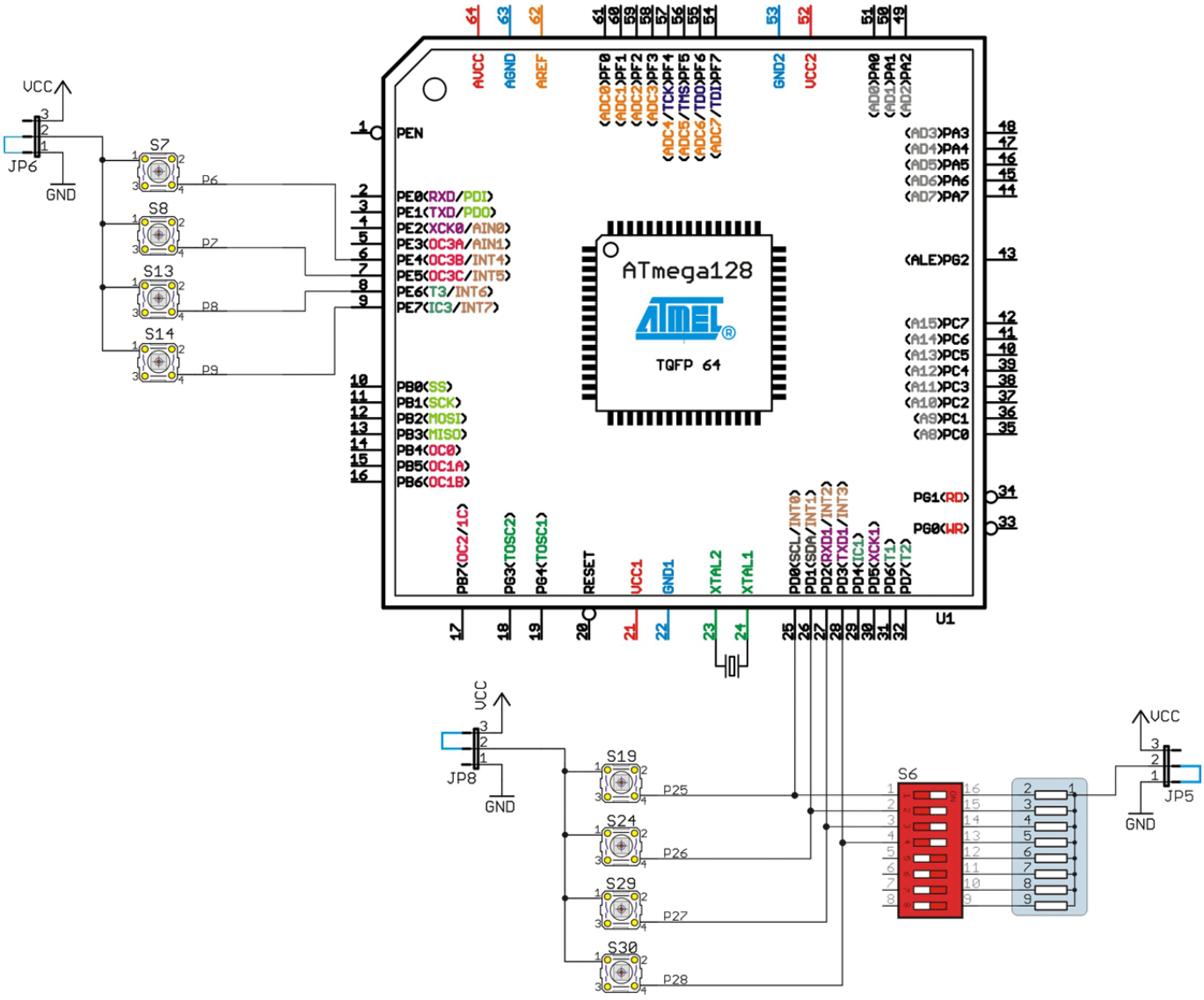
البرامج الفرعية

طباعة اسم المفتاح المضغوط على النافذة التسلسلية

والعودة إلى حلقة البرنامج الرئيسية

برنامج تشغيل الدارة في حالة JP6, موصلة إلى GND وJP8 موصلة إلى VCC:

في هذه الحالة فإنه يجب إلغاء تفعيل مقاومات الرفع الداخلية للبوابة PortD وتفعيل مقاومات سحب إلى النقطة الأرضية باستخدام مفاتيح توصيل المقاومات الخارجية S6.




```
$regfile = "m128def.dat"
$crystal = 1000000
$baud = 4800
'-----------------------
Config Debounce = 150

Config Pind.0 = Input
Config Pind.1 = Input
Config Pind.2 = Input
Config Pind.3 = Input

Config Pine.4 = Input
Config Pine.5 = Input
Config Pine.6 = Input
Config Pine.7 = Input
```

التوجيهات.

تحديد التأخير الزمني للقراءة الجديدة لحالة المفتاح

تعريف الأقطاب.

```
Portd.0 = 0 : Portd.1 = 0
Portd.2 = 0 : Portd.3 = 0
Porte.4 = 1 : Porte.5 = 1
Porte.6 = 1 : Porte.7 = 1
'-----------------------
Key1 Alias Pind.0
Key2 Alias Pind.1
Key3 Alias Pind.2
Key4 Alias Pind.3
Key5 Alias Pine.4
Key6 Alias Pine.5
Key7 Alias Pine.6
Key8 Alias Pine.7
'-----------------------
Do
   Debounce Key1 , 0 , Sw1 , Sub
   Debounce Key2 , 0 , Sw2 , Sub
   Debounce Key3 , 0 , Sw3 , Sub
   Debounce Key4 , 0 , Sw4 , Sub
   Debounce Key5 , 0 , Sw5 , Sub
   Debounce Key6 , 0 , Sw6 , Sub
   Debounce Key7 , 0 , Sw7 , Sub
   Debounce Key8 , 0 , Sw8 , Sub
Waitms 200
Loop
End
'-----------------------
Sw1:
   Print "Switch(1) is pressed!"
Return
'/------------
Sw2:
   Print "Switch(2) is pressed!"
Return
'/------------
Sw3:
   Print "Switch(3) is pressed!"
Return
'/------------
Sw4:
   Print "Switch(4) is pressed!"
Return
'/------------
Sw5:
   Print "Switch(5) is pressed!"
Return
'/------------
Sw6:
   Print "Switch(6) is pressed!"
Return
'/------------
Sw7:
   Print "Switch(7) is pressed!"
Return
'/------------
Sw8:
   Print "Switch(8) is pressed!"
Return
```

إلغاء تفعيل المقاومات الداخلية للبوابة Portd

تفعيل المقاومات الداخلية للبوابة PortE

الإشارة إلى الأسماء الفيزيائية للبوابات أثناء البرنامج بأسماء حقيقية.

حلقة البرنامج الرئيسي

قراءة حالة المفاتيح الثمانية

البرامج الفرعية

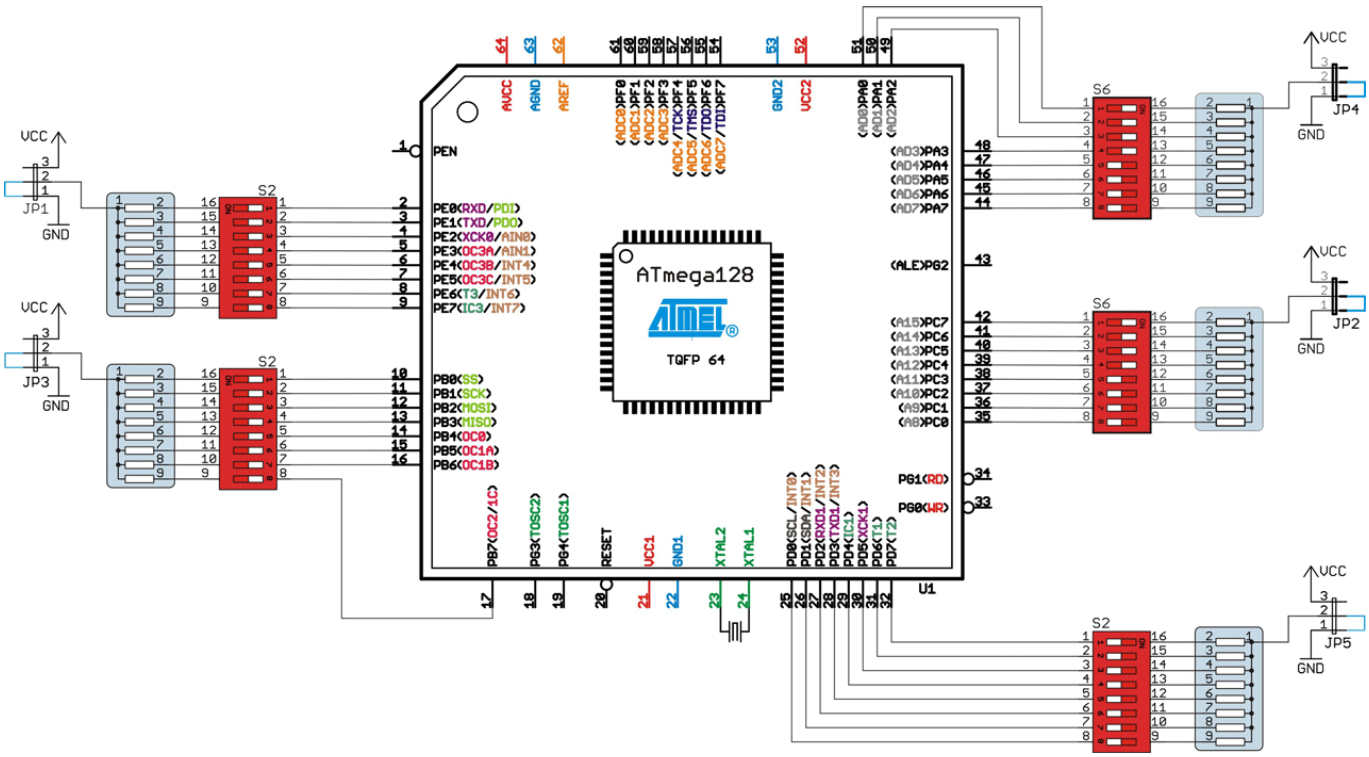
طباعة اسم المفتاح المضغوط على النافذة التسلسلية والعودة إلى حلقة البرنامج الرئيسية

التجربة الثالثة: استثمار توسعات البوابات (دخل/خرج) Exp.3: Expansion Connectors

الغاية من التجربة:

استثمار توسعات البوابات من أجل أغراض الدخل – الخرج الرقمي وتفعيل/إلغاء مقاومات الرفع خارجية.

مخطط التوصيل:



متطلبات توصيل:

يجب قصر الوصلات JP1~ JP6 حسب حالة المقاومة الخارجية للبوابة (رفع – سحب).

شرح عمل الدارة:

سوف نقوم بكتابة برنامج نعرف فيه جميع البوابات أعلاه كبوابات دخل رقمي ومن ثم سنقوم بقراءة القيم على مداخل البوابات عند تغيير قيمة الدخل باستخدام مفاتيح توصيل المقاومات الخارجية (S) في حالتين:

1- المقاومات الخارجية موصلة إلى النقطة VCC (مقاومات رفع للبوابات).

2- المقاومات الخارجية موصلة إلى النقطة GND (مقاومات سحب للبوابات).

- برنامج تشغيل الدارة في حالة المقاومات الخارجية موصلة إلى النقطة GND (مقاومات سحب للبوابات):

```
$regfile = "m128def.dat"
$crystal = 1000000
$baud = 4800
'-----------------------
Config Porte = Input
Config Portb = Input
Config Portd = Input
Config Portc = Input
Config Porta = Input
'-----------------------
Porta = 255
Portb = 255
Portc = 255
Portd = 255
Porte = 255
'-----------------------
Pa_value Alias Pina
Pb_value Alias Pinb
Pc_value Alias Pinc
Pd_value Alias Pind
Pe_value Alias Pine
'-----------------------
Do
   Print "PA= " ; Pa_value
      Print "PB= " ; Pb_value
         Print "PC= " ; Pc_value
            Print "PD= " ; Pd_value
               Print "PE= " ; Pe_value
                  Print "------------"
   Wait 1
Loop
End
'-----------------------
```

التوجيهات.

تعريف الأقطاب.

تفعيل المقاومات الداخلية للبوابات

الإشارة إلى الأسماء الفيزيائية للبوابات أثناء البرنامج بأسماء حقيقية.

حلقة البرنامج الرئيسي

قراءة قيمة الدخل على البوابات وطباعتها.

- برنامج تشغيل الدارة في حالة المقاومات الخارجية موصلة إلى النقطة VCC (مقاومات رفع للبوابات):

التوجيهات.	`$regfile = "m128def.dat"`
	`$crystal = 1000000`
	`$baud1 = 4800`

تعريف الأقطاب.	`'-----------------------`
	`Config Porte = Input`
	`Config Portb = Input`
	`Config Portd = Input`
	`Config Portc = Input`
	`Config Porta = Input`

إلغاء تفعيل المقاومات الداخلية للبوابات	`'-----------------------`
	`Porta = 0`
	`Portb = 0`
	`Portc = 0`
	`Portd = 0`
	`Porte = 0`

الإشارة إلى الأسماء الفيزيائية للبوابات أثناء البرنامج بأسماء حقيقية.	`'-----------------------`
	`Pa_value Alias Pina`
	`Pb_value Alias Pinb`
	`Pc_value Alias Pinc`
	`Pd_value Alias Pind`
	`Pe_value Alias Pine`

حلقة البرنامج الرئيسي
قراءة قيمة الدخل على البوابات وطباعتها.
	`'-----------------------`
	`Open "com2:" For Binary As #1`
	
	`Do`
	`Print #1 , "PA= " ; Pa_value`
	`Print #1 , "PB= " ; Pb_value`
	`Print #1 , "PC= " ; Pc_value`
	`Print #1 , "PD= " ; Pd_value`
	`Print #1 , "PE= " ; Pe_value`
	`Print #1 , "-----------"`
	`Wait 1`
	`Loop`
	
	`Close #1`
	`End`
	`'-----------------------`

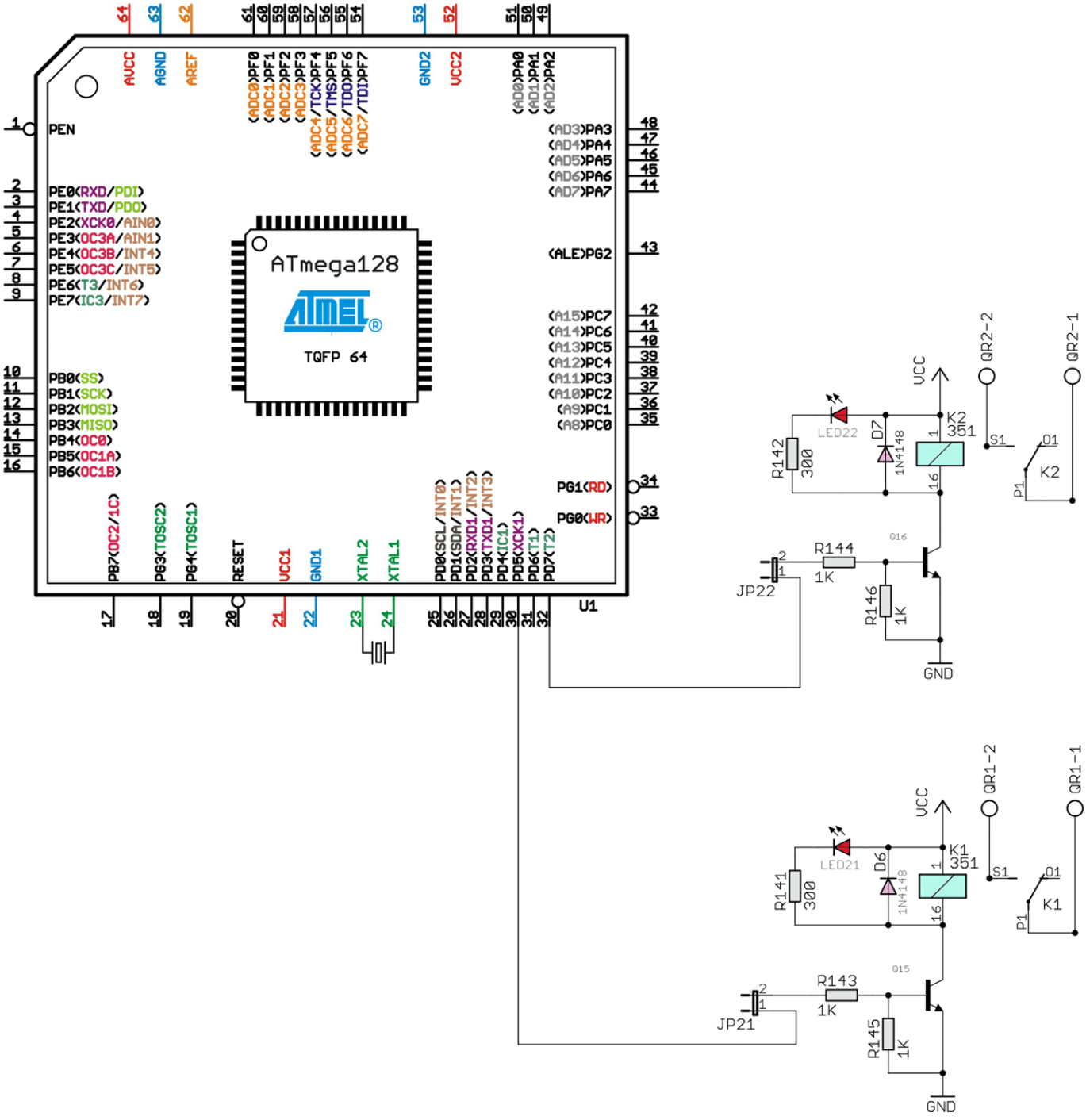
Exp.4: Controlling Relay (On/Of)

التجربة الرابعة: تشغيل زمني لمخارج تحكم ريليه

الغاية من التجربة:

استثمار مخارج بوابات المتحكم الرقمي للتحكم بعمل ريليه استطاعية.

مخطط التوصيل:



متطلبات توصيل:

يجب قصر الوصلات JP21 ~JP22.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج نعرف فيه البوابة Portd.5 والبوابة Portd.7 كبوابات خرج رقمي للتحكم بعمل الريليه K1 والريليه K2 على التوالي.

برنامج تشغيل الدارة:

التوجيهات.	`$regfile = "m128def.dat"` `$crystal = 1000000`

تعريف الأقطاب.	`'-----------------------`
	`Config Portd.5 = Output`
	`Config Portd.7 = Output`

الإشارة إلى الأسماء الفيزيائية للبوابات أثناء البرنامج بأسماء حقيقية.	`'-----------------------`
	`Relay1 Alias Portd.5`
	`Relay2 Alias Portd.7`

تعريف المتحولات	`'-----------------------` `Dim I As Byte`

حلقة البرنامج الرئيسي.	`'-----------------------`
	`For I = 0 To 9`
	`Set Relay1`
	`Wait 2`
	`Set Relay2`
	
	`Wait 2`
	
	`Reset Relay1`
	`Wait 2`
	`Reset Relay2`
	`Next I`
	`End`
	`'-----------------------`

برمجة المتحكمات المصغرة

التجارب العملية

الجلسة الثالثة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

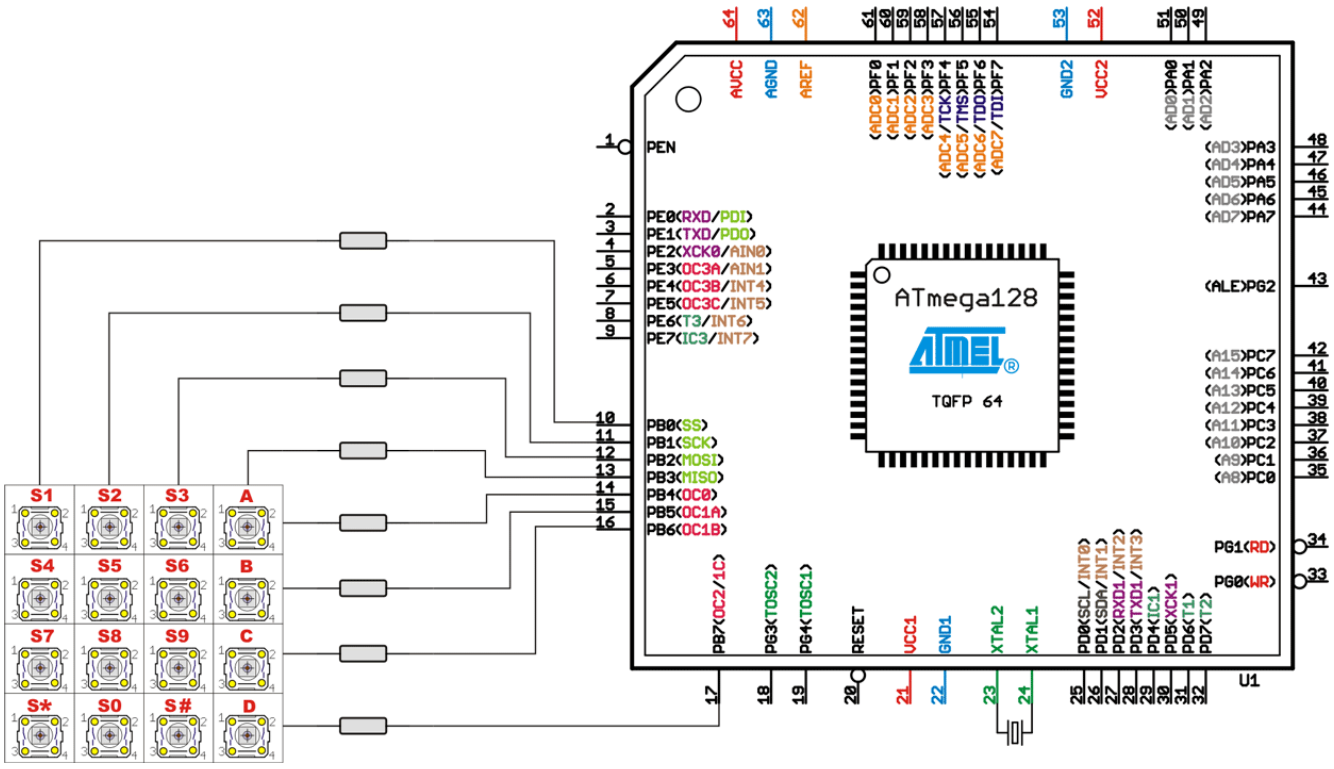
WALID BALID, Tuesday, December 15, 2009

التجربة الخامسة: لوحة مفاتيح ست عشرية مصفوفية Exp.5: Hexadecimal Array Keypad

الغاية من التجربة:

توصيل وبرمجة لوحة مفاتيح ست عشرية.

مخطط التوصيل:



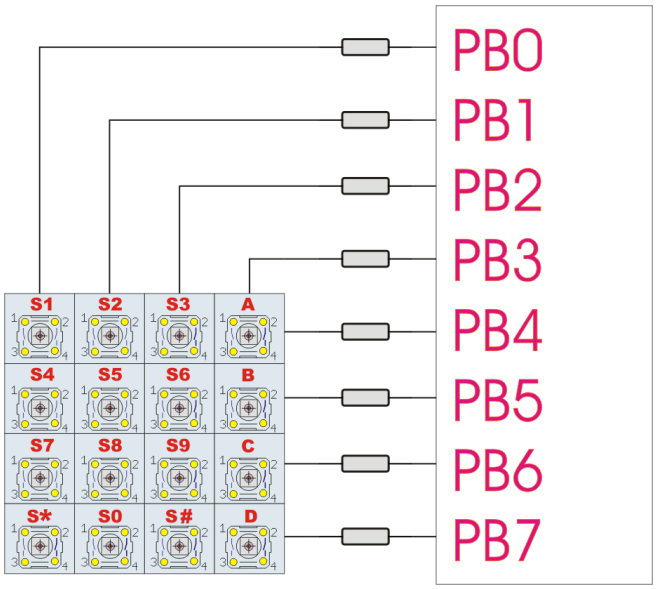
شرح عمل الدارة:

سوف نقوم بكتابة برنامج لقراءة حالة المفاتيح ومعرفة المفتاح المضغوط وطباعة اسم المفتاح المضغوط على النافذة التسلسلية.

التعليمات الجديدة:

من أجل قراءة لوحة مفاتيح ست عشرية فإننا نحتاج إلى تعليميتين أساسيتين:

شرح التعليمة	التعليمة البرمجية
تعريف البوابة الموصول معها لوحة المفاتيح وتعريف زمن التأخير لتفادي أثر العطالة الميكانيكية للمفاتيح.	`Config Kbd = Portb , Delay = 150`
قراءة حالة المفاتيح حيث أنه: 1- ستعود هذه التعليمة بالقيمة 16 = Var إذا لم يكن هناك أي مفتاح مضغوط. 2- ستعود هذه التعليمة بقيمة المفتاح المضغوط على الشكل التالي:	`Var = Getkbd()`



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

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 1000000
$baud = 4800
'-----------------------
Config Kbd = Portb , Delay = 150
'-----------------------
Dim Var As Byte
'-----------------------
Do
   Var = Getkbd()
      If Var < 16 Then
         Gosub Check_number
      End If
Loop

End
'-----------------------
Check_number:
Select Case Var
   Case 0 : Print "Key Pressed is (1)"
   Case 1 : Print "Key Pressed is (2)"
   Case 2 : Print "Key Pressed is (3)"
   Case 3 : Print "Key Pressed is (A)"
   Case 4 : Print "Key Pressed is (4)"
   Case 5 : Print "Key Pressed is (5)"
   Case 6 : Print "Key Pressed is (6)"
   Case 7 : Print "Key Pressed is (B)"
   Case 8 : Print "Key Pressed is (7)"
   Case 9 : Print "Key Pressed is (8)"
   Case 10 : Print "Key Pressed is (9)"
   Case 11 : Print "Key Pressed is (C)"
   Case 12 : Print "Key Pressed is (*)"
   Case 13 : Print "Key Pressed is (0)"
   Case 14 : Print "Key Pressed is (#)"
   Case 15 : Print "Key Pressed is (D)"
End Select
Return
'-----------------------
```

التوجيهات.

تعريف البوابة الموصلة معها لوحة المفاتيح.

تعريف المتحولات

حلقة البرنامج الرئيسي يتم فيها قراءة حالة المفاتيح.

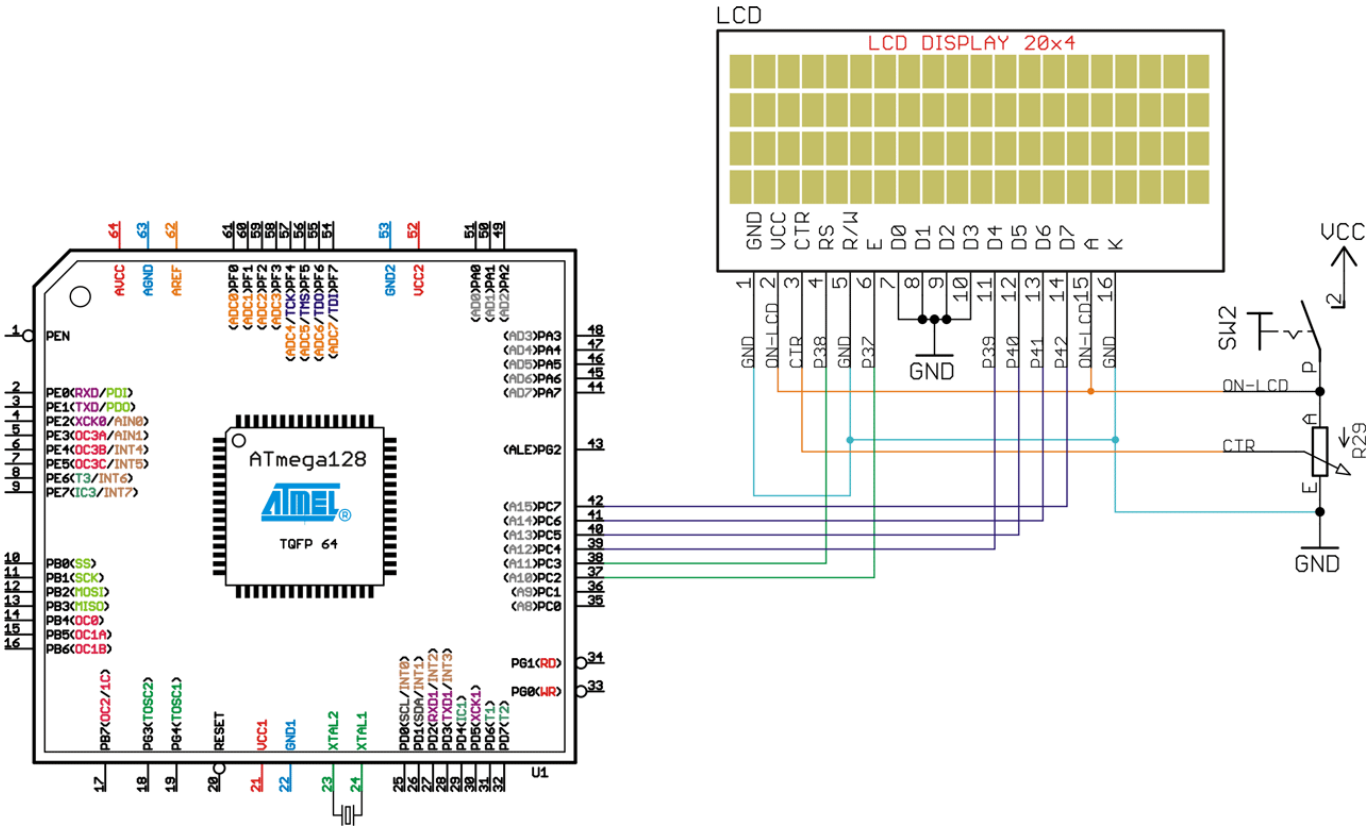
البرنامج لفرعي طباعة تعريف بالمفتاح المضغوط ثم العودة إلى البرنامج الرئيسي.

التجربة السادسة: برمجة شاشة الإظهار الكريستالية
Exp.6: Programming LCD

الغاية من التجربة:

توصيل وبرمجة شاشة إظهار محرفية 20x4.

مخطط التوصيل:



متطلبات توصيل:

يجب إغلاق المفتاح SW2.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لإظهار محارف نصية على شاشة إظهار كريستالية بالإضافة إلى استعراض الحركات الخاصة بالشاشة الكريستالية.

التعليمات الجديدة:

شرح التعليمة	التعليمة البرمجية
تعريف أقطاب شاشة الإظهار الموصلة مع المعالج	`Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 Db7 = Portc.7 , E = Portc.2 , Rs = Portc.3`
تعريف أبعاد شاشة الإظهار (أعمدة ❖ أسطر)	`Config Lcd = 20 * 4`
تعريف محرف إضافي باستخدام الأداة LCD Designer	`Deflcdchar 0 , 14 , 17 , 14 , 17 , 17 , 14 , 17 , 14`
مسح شاشة الإظهار	`Cls`

Lcd var	طباعة قيمة/محرف على شاشة الإظهار
Lowerline	تحريك مؤشر الكتابة إلى السطر التالي
Shiftlcd Right	إزاحة جميع المعلومات المظهرة على الشاشة عموداً إلى اليمين
Shiftlcd Left	إزاحة جميع المعلومات المظهرة على الشاشة عموداً إلى اليسار
Locate X , Y	توضع مؤشر الكتابة في موقع (سطر/عمود)
Shiftcursor Right	إزاحة مؤشر الكتابة عموداً واحداً إلى اليمين
Shiftcursor Left	إزاحة مؤشر الكتابة عموداً واحداً إلى اليسار
Home Upper	الانتقال إلى السطر/العمود الأول (نقطة البداية)
Cursor Off Noblink	إلغاء مؤشر الكتابة.
Cursor On Blink	تشغيل مؤشر الكتابة مع خفقان
Display Off	إطفاء شاشة الإظهار
Display On	تشغيل شاشة الإظهار
Thirdline	الانتقال إلى السطر الثالث
Fourthline	الانتقال إلى السطر الرابع
Home Third	وضع مؤشر الكتابة في بداية السطر الثالث
Home Fourth	وضع مؤشر الكتابة في بداية السطر الرابع
Lcd Chr(x)	طباعة المحرف الإضافي

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 8000000
'-----------------------
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 ,
Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.2 , Rs = Portc.3

Config Lcd = 20 * 4

Deflcdchar 0 , 14 , 17 , 14 , 17 , 17 , 14 , 17 , 14
Deflcdchar 1 , 17 , 14 , 17 , 14 , 14 , 17 , 14 , 17
'-----------------------

Dim A As Byte
'-----------------------
Do
   Cls
   Lcd "hello world!"
   Wait 1

   Lowerline
   Wait 1

   Lcd "LCD Test"
   Wait 1

   For A = 1 To 8
      Shiftlcd Right
```

التوجيهات.

تعريف أقطاب شاشة الإظهار الموصلة مع المتحكم.

تعريف موديل شاشة الظهار

تعريف محارف إضافية باستخدام الأداة LCD Designer

تعريف المتحولات

حلقة البرنامج الرئيسي يتم فيها الإظهار على الشاشة وعرض الحركات لأساسية وإظهار المحارف الإضافية.

```
      Waitms 500
   Next A

   For A = 1 To 8
      Shiftlcd Left
      Waitms 500
   Next A

   Locate 2 , 9
   Lcd "-L2,C9"
   Wait 1

   Shiftcursor Right
   Lcd "-*-"
   Wait 1

   Shiftcursor Left
   Lcd "*-*"
   Wait 1

   Home Upper
   Lcd "HELLO WORLD"
   Wait 1

   Cursor Off Noblink
   Wait 1

   Cursor On Blink
   Wait 1

   Display Off
   Wait 1

   Display On
   Wait 1

   Thirdline
   Lcd "   /Line 3"
   Wait 1

   Fourthline
   Lcd "   /Line 4"
   Wait 1

   Home Third
   Lcd ":)"
   Wait 1

   Home Fourth
   Lcd ":("
   Wait 1

   Cls
   Locate 1 , 9 : Lcd Chr(0)
   Wait 1
   Locate 1 , 11 : Lcd Chr(1)
Loop

End
```

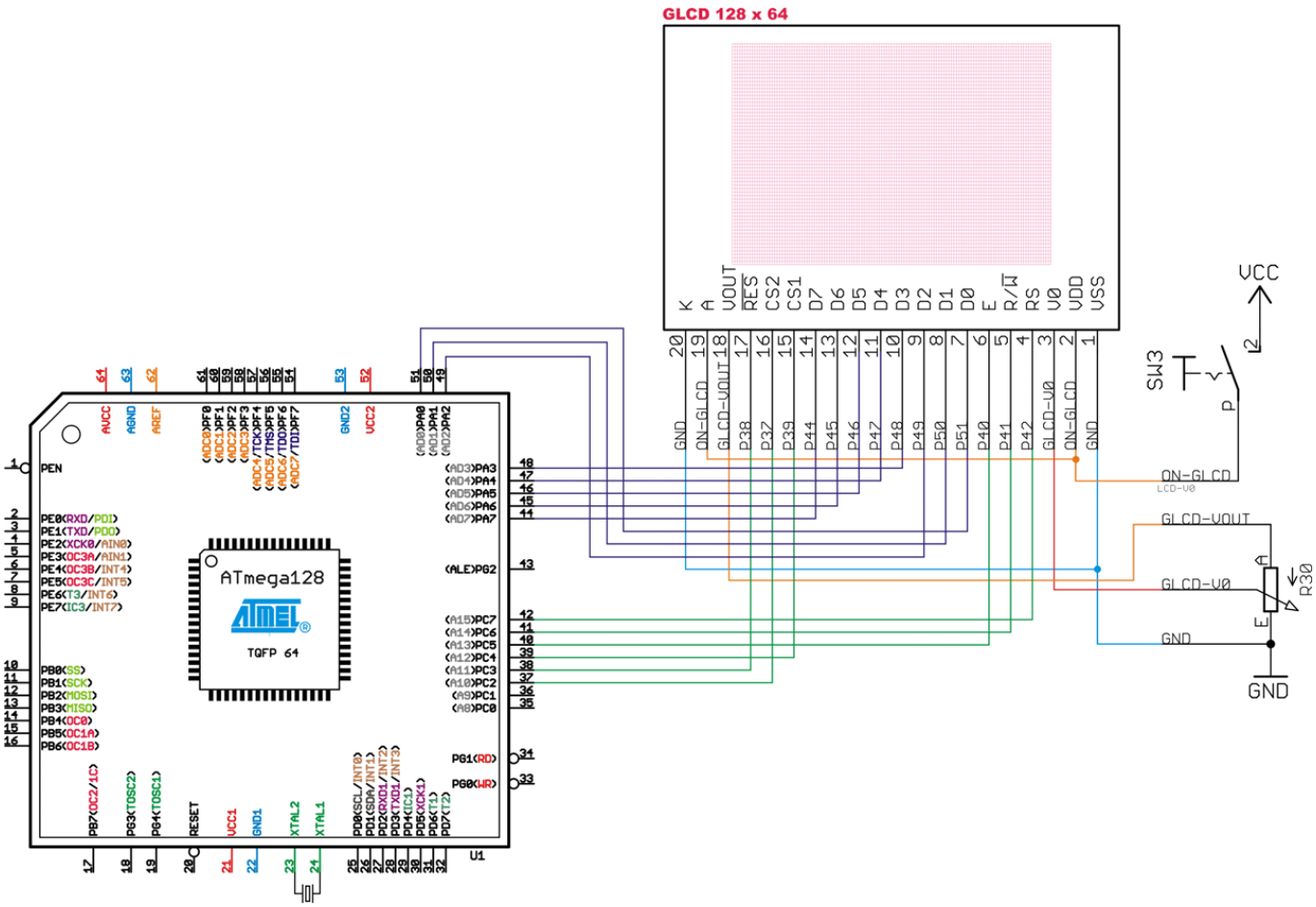
Exp.7: Programming Graphic LCD

التجربة السابعة: برمجة شاشة الإظهار الرسومية

الغاية من التجربة:

توصيل وبرمجة شاشة إظهار رسومية 128x64.

مخطط التوصيل:



متطلبات توصيل:

يجب إغلاق المفتاح SW3.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لإظهار محارف نصية وصور وإنشاءات رسومية على شاشة إظهار رسومية بالإضافة إلى استعراض الحركات الخاصة بالشاشة الرسومية.

التعليمات الجديدة:

شرح التعليمة	التعليمة البرمجية
تعريف الأقطاب الموصلة مع شاشة الإظهار الرسومية.	`Config Graphlcd = 128 * 64sed , Dataport= Porta, Controlport= Portc , Ce = 2 , Ce2 = 4 , Cd = 7 , Rd = 6 , Reset = 3 , Enable = 5`
تعريف مكتبة شاشة الإظهار الرسومية	`$lib "glcdKS108.lib"`

تعيين الخط المراد استخدامه	`Setfont Font`
مسح النصوص والرسوم	`Cls`
مسح النصوص فقط	`Cls Text`
مسح الرسوم فقط	`Cls Graph`
طباعة متحول أو نص على الشاشة في موقع محدد.	`Lcdat y , x , var [, inv]`
رسم خط استناداً إلى نقطة بداية ونقطة نهاية	`Line(X0 , Y0) - (X1 , Y1) , Color`
التحكم بإظهار أو إخفاء نقطة (SET/RST Pixel)	`Pset X , Y , Color`
رسم دائرة في نقطة مرجعية وبقطر r	`Circle(X , Y) , r , Color`
إظهار صورة في موقع محدد موجودة عند لافتة محددة	`Showpic X , Y , label`
تضمين ملف الخطوط من ملف خطوط خارجي	`$include "Font5x8.font"`
تضمين الصورة من ملف bgf خارجي	`$bgf "Smiley1.bgf"`

برنامج تشغيل الدارة:

التوجيهات.	`$regfile = "m128def.dat"`
	`$crystal = 8000000`
	`$lib "glcdKS108.lib"`

تعريف البوابة الموصلة معها شاشة الإظهار الرسومية.	`'-------------------------` `Config Graphlcd = 128 * 64sed , Dataport = Porta , Controlport = Portc , Ce = 2 , Ce2 = 4 , Cd = 7 , Rd = 6 , Reset = 3 , Enable = 5`
تعريف المتحولات	`'-------------------------` `Dim X As Byte , Y As Byte`
مسح شاشة الإظهار	`'-------------------------` `Cls : Cls Text : Cls Graph`

طباعة نصوص باستخدام خط من الحجم 8x8	`'-------------------------`
	`Setfont Font8x8`
	`Lcdat 3 , 11 , "Font 5x8 Test"`
	`Lcdat 4 , 11 , "Font 6x8 Test"`
	`Lcdat 5 , 11 , "Font 8x8 Test"`
	`Wait 2`

طباعة نصوص باستخدام خط من الحجم 5x8	`Cls`
	`Setfont Font5x8`
	`Lcdat 1 , 1 , "LINE1" : Wait 1`
	`Lcdat 2 , 1 , "LINE2" : Wait 1`
	`Lcdat 3 , 1 , "LINE3" : Wait 1`

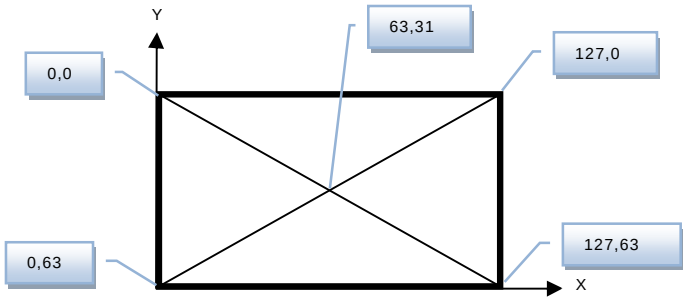
طباعة نصوص باستخدام خط من الحجم 6x8	`Setfont Font6x8`
	`Lcdat 4 , 1 , "LINE4" : Wait 1`
	`Lcdat 5 , 1 , "LINE5" : Wait 1`
	`Lcdat 6 , 1 , "LINE6" : Wait 1`

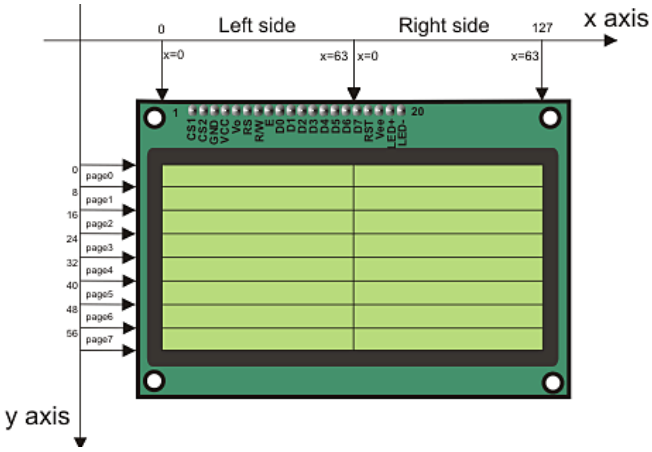
طباعة نصوص باستخدام خط من الحجم 8x8	`Setfont Font8x8`
	`Lcdat 7 , 1 , "LINE7" : Wait 1`
	`Lcdat 8 , 1 , "LINE8" : Wait 1`

طباعة نصوص باستخدام خط من الحجم 16x16	`'/------------/`
	`Cls`
	`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
'/------------/
Setfont Font8x8
   Lcdat 4 , 11 , "Drawing Lines" :
Wait 2
   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
      Cls Graph
'/------------/
   Lcdat 4 , 11 , "SET/RST Pixel" :
Wait 2
   Cls Text
      For X = 0 To 127
         Pset X , 20 , 255
         Pset X , 43 , 255
         Waitms 200
      Next X

      For Y = 0 To 63
         Pset 42 , Y , 255
         Pset 86 , Y , 255
         Waitms 200
      Next Y

      For X = 127 To 0 Step -1
         Pset X , 20 , 0
         Pset X , 43 , 0
         Waitms 200
      Next X

      For Y = 63 To 0 Step -1
         Pset 42 , Y , 0
         Pset 86 , Y , 0
         Waitms 200
      Next X
      Cls Graph
'/------------/
   Lcdat 4 , 11 , "Drawing Circle" :
Wait 2
   Cls Graph
   For X = 1 To 31
      Circle(63 , 31) , X , 255
      Waitms 200
      Circle(63 , 31) , X , 0
   Next X
```

رسم خطوط وفق التقسيم التالي لشاشة الإظهار





رسم خطوط نقطة بنقطة

ومن ثم مسحها نقطة بنقطة

رسم دائرة

```
   Cls Graph
'/------------/
   Showpic 0 , 0 , Smiley1
   Wait 2
   Cls : Cls Text : Cls Graph

   Showpic 0 , 0 , Smiley2
   Wait 2
   Cls Graph

   Showpic 0 , 0 , Smiley3
   Wait 2
   Cls Graph

   Showpic 0 , 0 , Smiley4
   Wait 2
'/------------/
  Glcdcmd &H3E , 1 : Glcdcmd &H3E , 2
  Wait 1
  Glcdcmd &H3F , 1 : Glcdcmd &H3F , 2
'/------------/
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"

Smiley4:
$bgf "Smiley4.bgf"
```

إظهار الصور المضمنة من ملفات خارجية

- إطفاء قسمي شاشة الإظهار الرسومية اليميني واليساري
- إعادة تشغيل القسمين

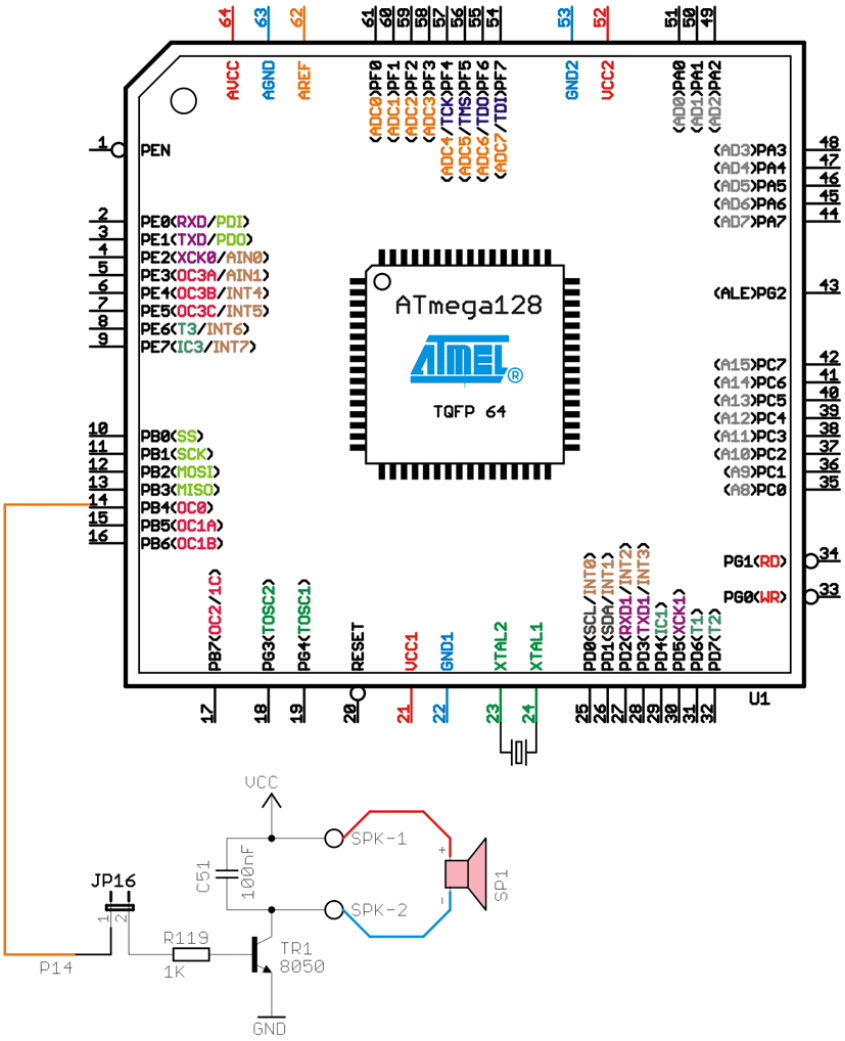
تعليمة تضمين ملفات النصوص من ملفات برمجية خارجية

تضمين ملفات الصور البرمجية من ملفات خارجية

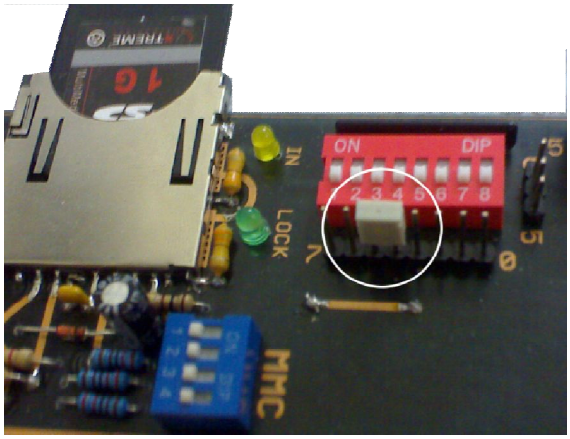
الغاية من التجربة:

توليد نغمات صوتية لأغراض التنبيه الصوتي.

مخطط التوصيل:



متطلبات توصيل:



يجب إغلاق نقطة الوصل JP16 باستخدام Jumper. ووضع نقطة وصل بين PB4,PB5 كما في الصورة.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لتوليد نغمات تنبيه صوتية تظهر على مجهار صوتي خارجي. يمكن تشكيل أي نغمة من خلال دمج عدة تعليمات sound التي تعتمد على توليد نبضة وتغيير عرض ودور النبضة وبالتالي يتغير التردد على قاعدة المفتاح الترانزستوري الذي يتحكم بالمجهار الخارجي (Speaker)

التعليمات الجديدة:

شرح التعليمة	التعليمة البرمجية
توليد صوت على القطب (Speaker) وبعرض نبضة محدد بالمتحول (Pulses) وزمن النبضة محدد بالمتحول (Periods).	`Sound Speaker , Pulses , Periods`

برنامج تشغيل الدارة:

التوجيهات.	`$regfile = "m128def.dat"` `$crystal = 8000000`
تعريف القطب الموصل في خرجه المكبر الصوتي	`Config Portb.4 = Output` `Speaker Alias Portb.4`

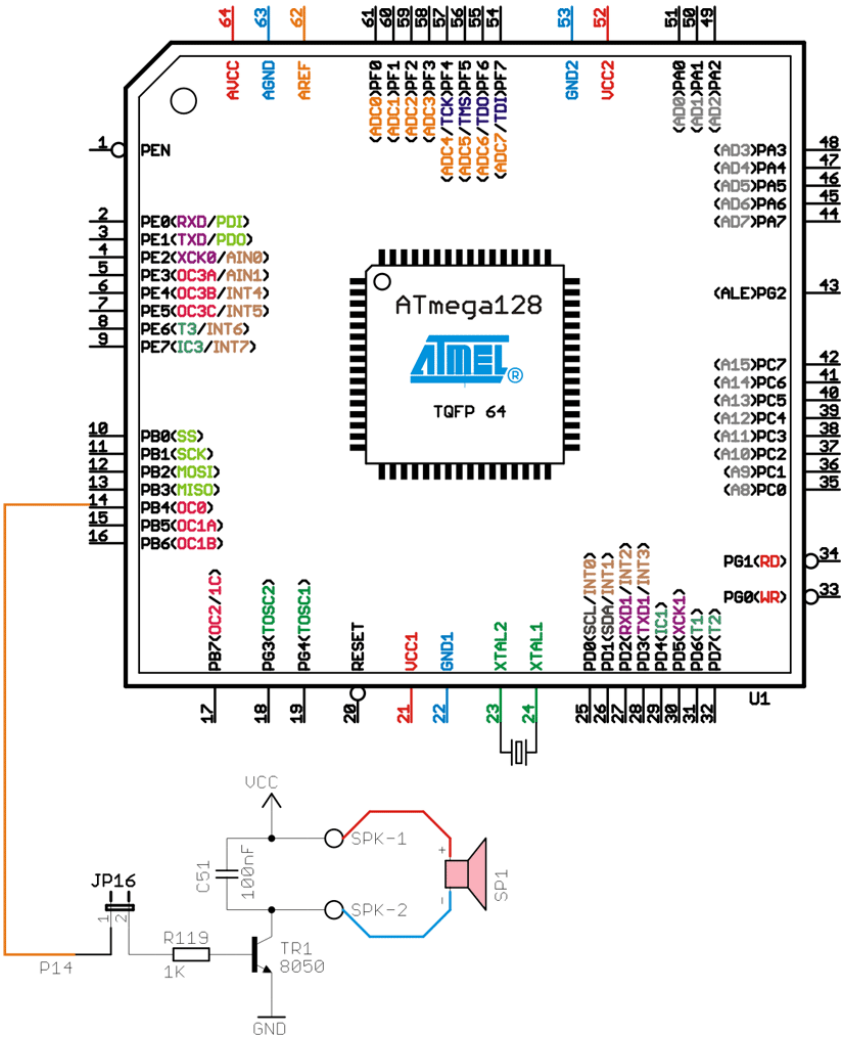
تعريف المتحولات وإسناد القيم الابتدائية لها	`Dim Pulses As Word , Periods As Word`
	`Dim I As Byte`
	
	`Pulses = 50 : Periods = 100`

حلقة البرنامج الرئيسي يتم فيها توليد أصوات نغمات مختلفة.	`Sound Speaker , 100 , 200 : Wait 1`
	
	`Do`
	`  Sound Speaker , Pulses , Periods`
	`  Waitms 1`
	`  Periods = Periods + 5`
	`Loop Until Periods = 1000`
	
	`For I = 0 To 6`
	`  Sound Speaker , 150 , 800`
	`  Waitms 1000`
	`Next I`
	
	`Sound Speaker , 100 , 200`
	`End`

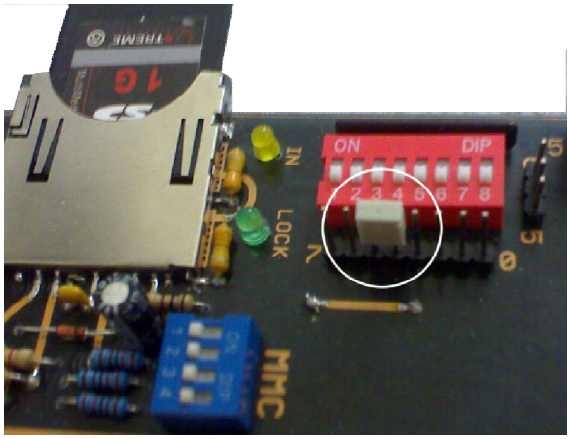
الغاية من التجربة:

توليد نغمات صوتية DTMF (Dual Tone Multi Frequency) لأغراض التنبيه الصوتي.

مخطط التوصيل:



متطلبات توصيل:



يجب إغلاق نقطة الوصل JP16 باستخدام Jumper. ووضع نقطة وصل بين PB4,PB5 كما في الصورة.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لتوليد نغمات تنبيه DTMF تظهر على مجهار صوتي خارجي موصل مع القطب OC1A.

التعليمات الجديدة:

شرح التعليمة	التعليمة البرمجية
توليد صوت على القطب (OC1A). **Number:** أحد أرقام لوحة مفاتيح الهاتف الستة عشر. **Duration**: زمن توليد النغمة بالميلي ثانية. **String**: مجموعة الأرقام التي تشكل رقم الاتصال كاملاً.	`Dtmfout Number , Duration` `Dtmfout String , Duration`

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"                                    التوجيهات.
$crystal = 8000000
'-------------------
Enable Interrupts                                           تفعيل المقاطعات لأن التعليمة DTMF تعتمد على
                                                            المؤقت1 في عملها
'-------------------
Dim Var As Byte                                             تعريف المتحولات وإسناد القيم الابتدائية لها
Dim Btmp As Byte , Sdtmf As String * 10
Sdtmf = "1234567890"
'-------------------
Do                                                          حلقة البرنامج الرئيسي يتم فيها توليد أصوات نغمات
   Dtmfout Sdtmf , 50                                       مختلفة.
      Waitms 1000

   For Btmp = 0 To 15
      Dtmfout Btmp , 100
      Waitms 500
   Next Btmp
Loop
```

ملاحظة1: إن التعليمة **Dtmfout** تعتمد على المؤقت1 في توليد النغمات لذا يجب تفعيل شعاع المقاطعة كما أنها ستقوم بإصدار النغمات على القطب OC1A فقط.

ملاحظة2: إن تردد الهزاز الكريستالي يجب أن يكون بين 4MHZ ~ 8MHZ.

ملاحظة3: الرجاء مراجعة الوثيقة الفنية Atmel application note (AN314) لفهم العلاقات الرياضية لهذه التعليمة.

برمجة المتحكمات المصغرة

التجارب العملية

الجلسة الرابعة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

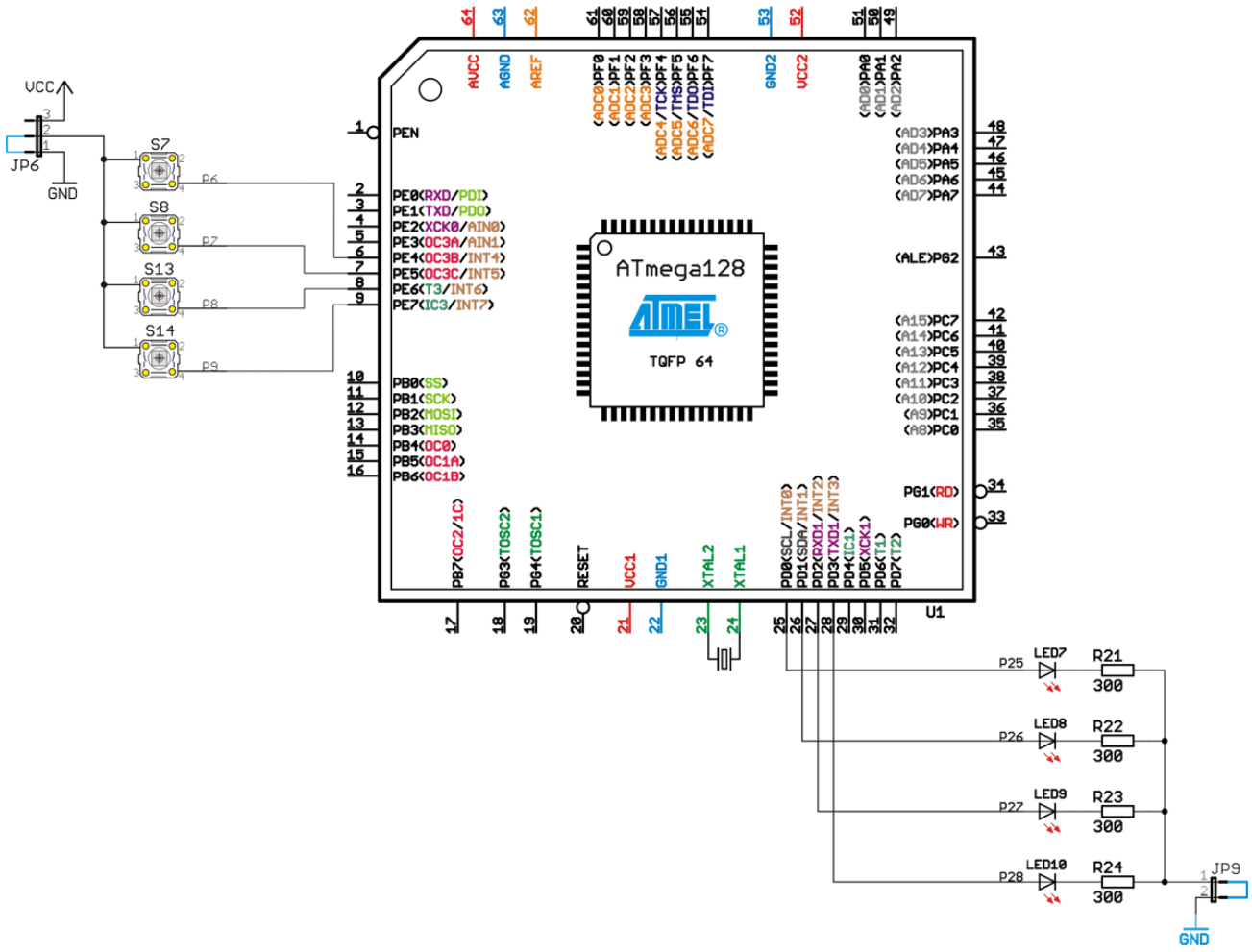
WALID BALID, Tuesday, December 15, 2009

التجربة العاشرة: تعليمات الإزاحة والدوران
Exp.10: Shift & Rotate Instructions

الغاية من التجربة:

التعرف على تعليمات الإزاحة (Shift) والدوران (Rotate).

مخطط التوصيل:



شرح عمل الدارة:

سوف نقوم بكتابة برنامج بحيث يقوم المفتاح S6 بالإزاحة للأمام (Shift Right)، (إزاحة بت يظهر على الثنائيات الضوئية) والمفتاح S7 بالإزاحة للخلف (Shift Left).

كذلك المفتاح S13 يقوم بعملية دوران نحو اليمين (Rotate Right) والمفتاح S14 يقوم بعملية دوران نحو اليسار (Rotate Right).

متطلبات التوصيل:

يجب إغلاق نقطة الوصل JP6 و JP9 باستخدام Jumper.

التعليمات الجديدة:

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

برنامج تشغيل الدارة:

التوجيهات.

```
$regfile = "m128def.dat"
$crystal = 8000000
'---------------------
```

تعريف البوابات الموصلة معها المفاتيح والثنائيات الضوئية.

```
Config Pine.4 = Input
Config Pine.5 = Input
Config Pine.6 = Input
Config Pine.7 = Input

Pine.4 = 1 : Pine.5 = 1
Pine.6 = 1 : Pine.7 = 1

Shift_r Alias Pine.4
Shift_l Alias Pine.5
Rotat_r Alias Pine.6
Rotat_l Alias Pine.7

Config Portd = Output
Leds Alias Portd
Portd = &B00000001
'---------------------
```

حلقة البرنامج الرئيسي يتم فيها فحص حالة المفاتيح.

```
Do
   Debounce Shift_r , 0 , Sr , Sub
   Debounce Shift_l , 0 , Sl , Sub
   Debounce Rotat_r , 0 , Rr , Sub
   Debounce Rotat_l , 0 , Rl , Sub
Loop
End
'---------------------
```

البرامج الفرعية.

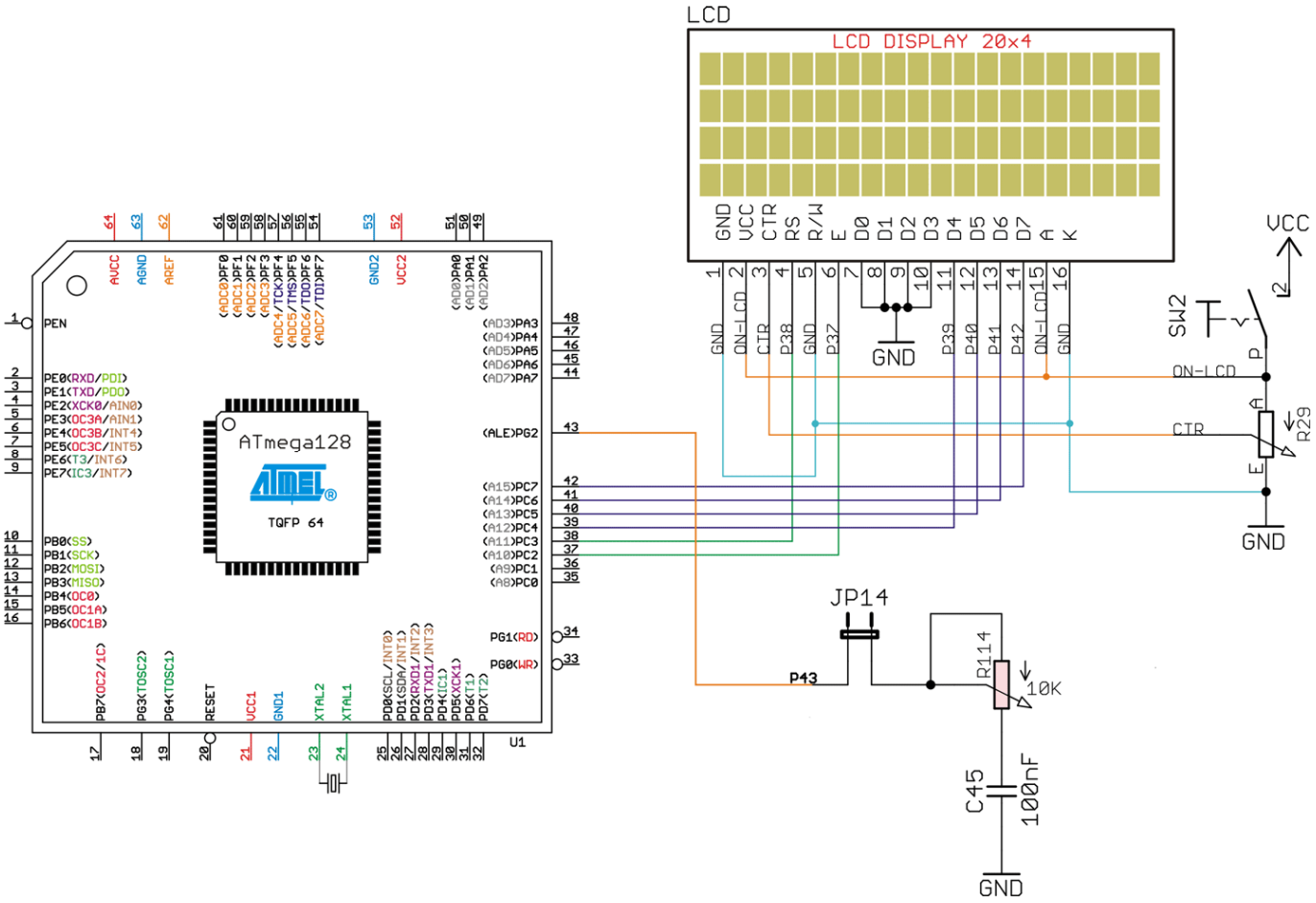
```
Sr:
   If Leds > 1 Then Shift Leds , Right , 1
Return
'---------------------
Sl:
   If Leds < 128 Then Shift Leds , Left , 1
Return
'---------------------
Rr:
   Rotate Leds , Right , 1
Return
'---------------------
Rl:
   Rotate Leds , Left , 1
Return
'---------------------
```

التجربة الحادية عشرة: قراءة قيمة مقاومة أو مكثف
Exp.11: Get RC Value

الغاية من التجربة:

قراءة قيمة تغير الثابت الزمني لشحن المقاومة والمكثف معاً.

مخطط التوصيل:



شرح عمل الدارة:

سوف نقوم بتوصيل مقاومة متغيرة ومكثف متغير مع أحد أقطاب المتحكم المصغر وقراءة قيمة تغير الثابت الزمني لشحن المقاومة والمكثف معاً باستخدام التعليمة **Getrc**.

متطلبات التوصيل:

يجب إغلاق نقطة الوصل JP14 باستخدام Jumper. وكذلك إغلاق المفتاح SW2 لتشغيل شاشة الإظهار.

التعليمات الجديدة:

شرح التعليمة	التعليمة البرمجية
تقوم هذه التعليمة بشحن مكثف موصول على القطب **y** من البوابة **pinx** ثم تقوم بتشغيل مؤقت زمني وقياس زمن تفريغ المكثف والناتج هو ثابت زمني تتعلق قيمته بقيمة المقاومة والمكثف معاً!	`var = Getrc(pinx , y)`

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 8000000
'-----------------------
Config Lcdpin = Pin , Db4 = Portc.4 ,
Db5 = Portc.5 , Db6 = Portc.6 , Db7 =
Portc.7 , E = Portc.2 , Rs = Portc.3
Config Lcd = 20 * 4
'-----------------------
Dim Rc_var As Word
'-----------------------
Cls : Cursor Off
Lcd "RC Val= "
Do
   Rc_var = Getrc(ping , 2)
   Locate 1 , 9
   Lcd Rc_var
   Waitms 1000
   Locate 1 , 8 : Lcd Spc(7)
Loop

End
```

التوجيهات.

تعريف البوابة الموصلة معها شاشة الإظهار الكريستالية.

تعريف المتحولات

حلقة البرنامج الرئيسي.

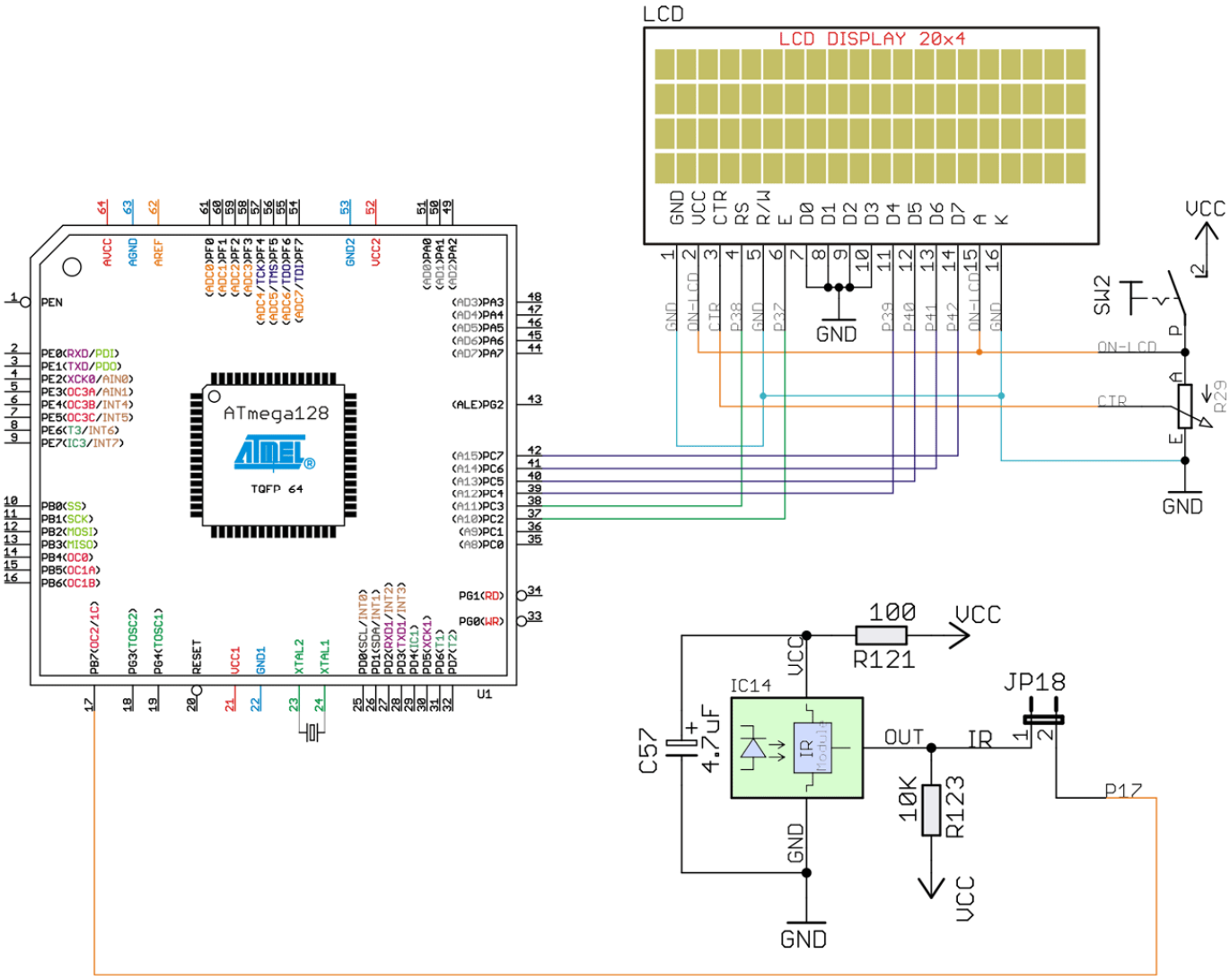
التجربة الثانية عشرة: دارة استقبال IR وفق المعيار RC5

Exp.12: IR receiver, RC5-code

الغاية من التجربة:

توصيل و برمجة مستقبل أشعة تحت الحمراء (CLRM-2038S) وفق البروتوكولات RC5, RC5-Extended.

مخطط التوصيل:



شرح عمل الدارة:

سوف نقوم بكتابة برنامج لدارة استقبال أوامر مرسلة من أجهزة التحكم بالأشعة تحت الحمراء التي تعمل وفقت البروتوكول RC5.

متطلبات التوصيل:

يجب إغلاق نقطة الوصل JP18 باستخدام Jumper. وكذلك إغلاق المفتاح SW2 لتشغيل شاشة الإظهار.

التعليمات الجديدة:

التعليمة البرمجية	شرح التعليمة
`Config Rc5 = Pinb.7`	تعريف القطب الموصول مع خرج مستقبل IR
`Getrc5(address , Command)`	فحص العنوان والأمر من الجهاز المرسل

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 8000000
'---------------------
Config Lcdpin = Pin , Db4 = Portc.4 ,
Db5 = Portc.5 , Db6 = Portc.6 , Db7 =
Portc.7 , E = Portc.2 , Rs = Portc.3
Config Lcd = 20 * 4
'---------------------
Config Rc5 = Pinb.7
'---------------------

Dim Address As Byte , Command As Byte
Enable Interrupts
'---------------------
Cls
Do
   Gosub Remote_control
Loop
'---------------------

Remote_control:
Getrc5(address , Command)
   If Address = 0 Then
      Command = Command And &B01111111
      Cls
      Lcd "Command is: " ; Command
      Waitms 50
   End If
Return
'---------------------
```

التوجيهات.

تعريف البوابة الموصل معها شاشة الإظهار الكريستالية.

تعريف القطب الموصل معه خرج المستقبل

تعريف المتحولات

تفعيل علم المقاطعة العامة

حلقة البرنامج الرئيسي يتم فيها استدعاء البرنامج الفرعي لقراءة حالة المفتاح المضغوط في جهاز التحكم

البرنامج لفرعي:

فحص العنوان إذا ما كان هو عنوان جهاز TV

فحص وطباعة تعريف بالمفتاح المضغوط في جهاز التحكم ثم العودة إلى البرنامج الرئيسي.

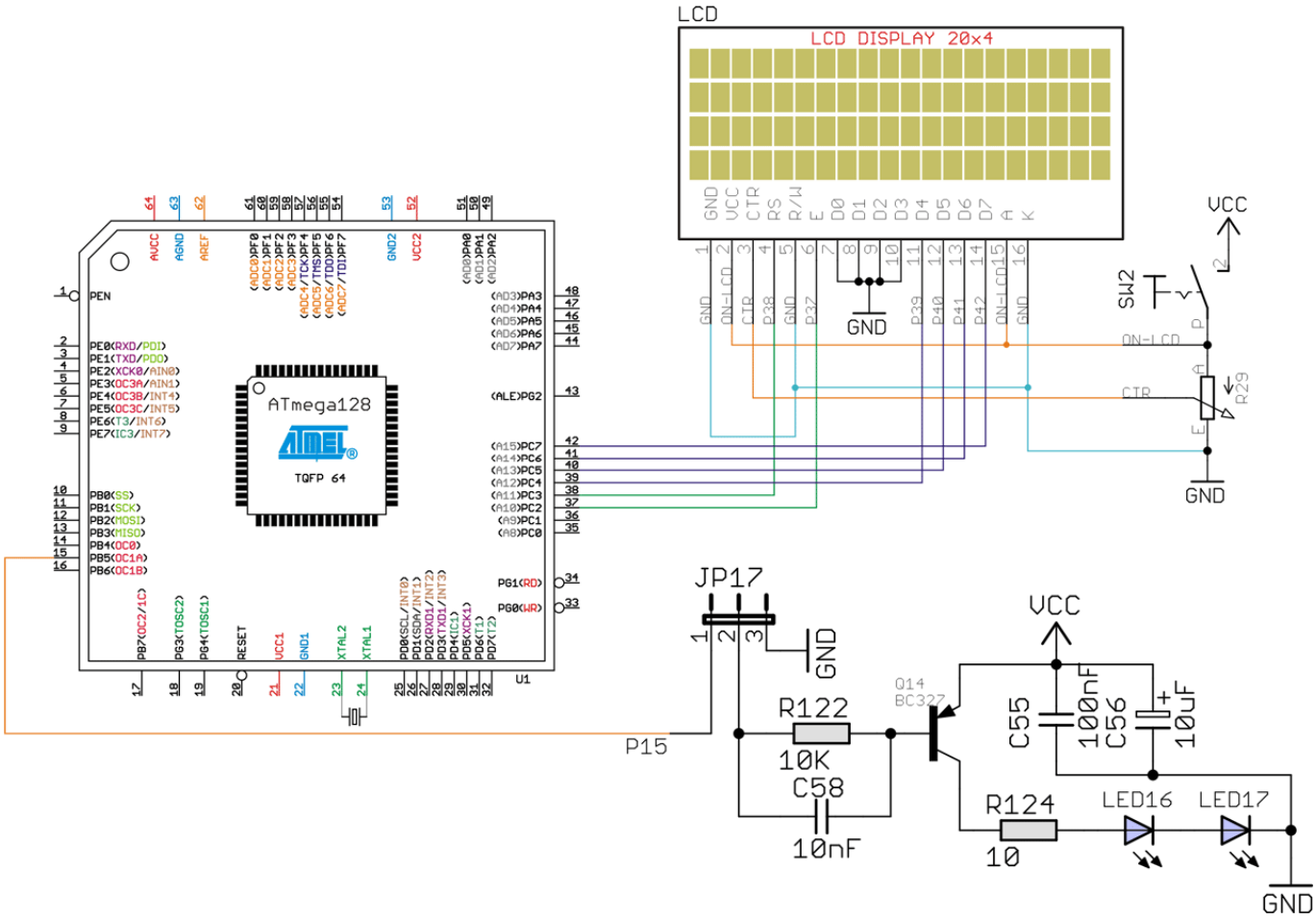
التجربة الثالثة عشر: دارة إرسال IR وفق المعيار RC5

Exp.13: IR transmitter, RC5-code

الغاية من التجربة:

توصيل وبناء دارة إرسال أشعة تحت الحمراء وفق المعيار RC5.

مخطط التوصيل:



شرح عمل الدارة:

سوف نقوم بتوصيل وبرمجة مرسل أشعة تحت الحمراء لأغراض التحكم عن بعد وفق البروتوكولات RC5, RC5-Extended. أي أننا في النهاية يمكن أن نصمم جهاز تحكم خاص.

متطلبات التوصيل:

يجب إغلاق نقطة الوصل JP17 باستخدام Jumper. وكذلك إغلاق المفتاح SW2 لتشغيل شاشة الإظهار.

التعليمات الجديدة:

من أجل قراءة لوحة مفاتيح ست عشرية فإننا نحتاج إلى تعليميتين أساسيتين:

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

برنامج تشغيل الدارة:

`$regfile = "m128def.dat"` `$crystal = 8000000`	التوجيهات.

`'----------------------`
`Config Debounce = 200`
 	
`Config Pine.4 = Input`
`Porte.4 = 1`
	تعريف القطب الموصل مع مفتاح الإرسال.

`'----------------------`
`Dim Togbit As Byte , Command As Byte , Address As Byte`
 	
`Command = 12`
`Togbit = 0`
`Address = 0`
	تعريف المتحولات
	إسناد القيم للمتحولات

`'----------------------`
`Do`
`  Debounce Pine.4 , 0 , Pwr_cmd , Sub`
`Loop`
 	
`End`
	حلقة البرنامج الرئيسي يتم فيها إرسال الأمر على القطب `OC1A pin` كلما تم ضغط المفتاح.

`'----------------------`
 	
`Pwr_cmd:`
`  Rc5send Togbit , Address , Command`
`  Waitms 200`
`Return`
`'\-------------\`
	البرنامج لفرعي يتم فيه إرسال الأمر والعنوان ثم العودة إلى البرنامج الرئيسي.

ملحق للتجربة الثانية والثالثة عشرة: Appendix: Exp.12 and Exp.13

شرح مفصل عن جهاز التحكم بالأشعة تحت الحمراء والمعيار RC5-code

Detailed Article about Remote Control and IR Module

ما هي الأشعة تحت الحمراء؟ What is infrared?

هي عبارة عن طاقة إشعاع ضوئي غير مرئي يقع تحت حزمة الترددات المرئية لأعيينا. في الحقيقة إن الأشعة تحت الحمراء هي ضوء طبيعي يبلغ طول الموجه لهذه الأشعة 950nm وهي موجة قصيرة جداً لهذا لا يمكن للعين أن ترى الضوء المنبعث من مرسل الأشعة تحت الحمراء.

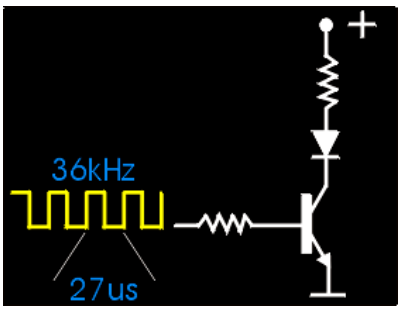
تعتبر الأشعة تحت الحمراء من أرخص الطرق وأسهلها للتحكم عن بعد بالأجهزة وذلك ضمن مجال مرئي، وتستخدم بكثرة في الأجهزة الكهربائية المنزلية وأجهزة التسجيل الرقمي والعرض المرئي. بالإضافة إلى سهولة توليدها، كما أنها لا تعاني من التدخل الكهرومغناطيسي، ولكنها في نفس الوقت يمكن أن تتصادم مع إشعاعات تحت حمراء أخرى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

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

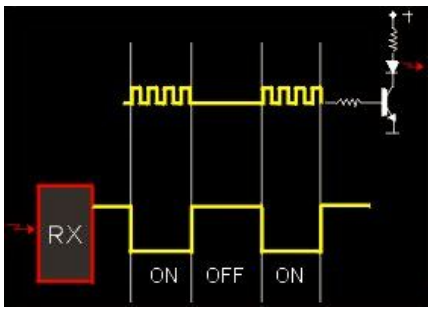
إن حزمة ترددات الأشعة تحت الحمراء تتراوح بين 30KHZ – 60KHZ ومجال الأشعة الأفضل هو ضمن 36KHZ والحزم التي حوله (38KHZ). لذلك تستخدم أجهزة التحكم بالأشعة تحت الحمراء الحزمتين 36KHZ, 38KHZ لإرسال المعلومات وهذا يعني أن الثنائي المرسل للأشعة تحت الحمراء سوف يتذبذب 36~38 ألف مرة خلال دور قدره واحد ثانية من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

إن مسألة إرسال تردد 36KHZ, 38KHZ هي مسألة سهلة، لكن الصعوبة تكمن في استقبال هذه الترددات وخصوصاً أن هذه الترددات انتقلت عبر الهواء وتراكبت معها ترددات الضجيج المحيط، لهذا السبب تقوم بعض الشركات بإنتاج مستقبلات الأشعة تحت الحمراء التي تحوي في بنيتها على مرشحات الحزمة ودارات فك

التشفير ودارات القص للحزم الغير مرغوبة، وهذا بدوره يساعد على استخلاص الإشارة الحقيقية. الشكل التالي يبين دارة إرسال بسيطة من أجل إرسال تردد 36KHZ، وذلك بتطبيق إشارة مربعة 27uS على قاعدة الترانزستور الشكل 1. إن المستقبل سيقوم باستلام الإشارة المرسلة وتعديلها كما في الشكل 2.



الشكل 1



الشكل 2

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

ماهي معايير الإرسال باستخدام الأشعة تحت الحمراء؟ What is IR Transmiting protocols?

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

معيار الإرسال RC5 RC5 Transmitting protocols?

إن اهتمامنا ينصب بشكل كلي على معيار RC5 الذي طورته شركة فيلبس ويتلخص بإرسال قطار من 14 نبضة في كل مرة يتم فيها الضغط على أحد أزرار جهاز التحكم وبزمن 1.728mS لكل نبضة، وهذا القطار من النبضات يتكرر كل 130mS إذا أبقيت المفتاح مضغوطاً.

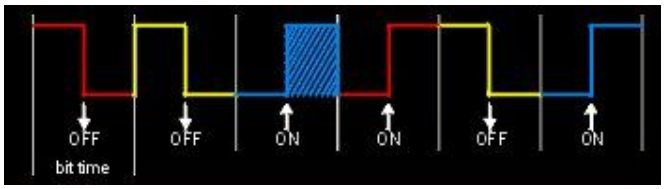
ولفهم مبدأ عمل هذا البروتوكول يجب التعرف إلى البارامترات التالية:

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

إن هذه البارامترات تختلف حسب نوع المستقبل.

إن كل نبضة من قطار النبضات هي بت واحد منقسم إلى قسمين: له نصف يميني ونصف يساري، ولكل منهما مستوى منطقي معاكس للآخر دائماً. فإذا كان البت المرسل من طرف الإرسال هو واحد منطقي، فإن القسم

اليميني من البت سيكون واحد منطقي، بينما القسم اليساري سيكون صفر منطقي، وإذا كان البت المرسل هو صفر منطقي، فستكون عكس الحالة السابقة تماماً.

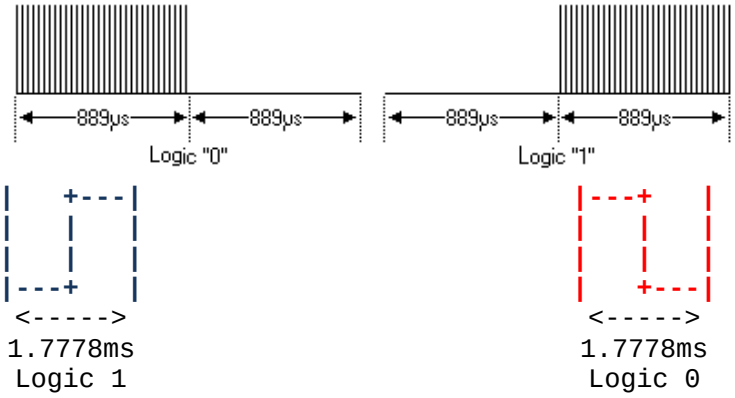


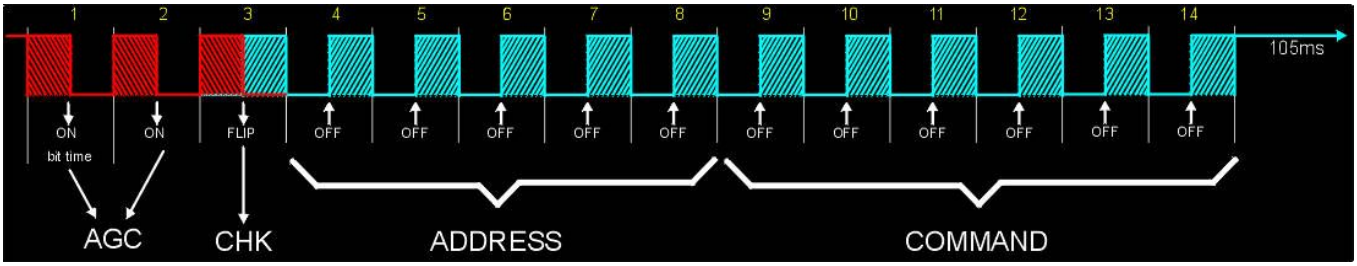
الشكل يبين المنطق الحقيقي الذي سوف تستقبله

بمعنى آخر، يمكنك أن تستنتج أن القسم اليميني من البت المستقبل، سيكون له نفس المستوى المنطقي للبت المرسل، من الشكل السابق تجد النبضة الزرقاء لها مستوى واحد منطقي، وهذا يعني أن البت المرسل هو واحد منطقي أيضا، ولكن القسم اليساري سيكون عكسه.

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

بينما من أجل إرسال "1" سيكون لدينا الحالة المعاكسة تماماً، 32 x 27µs of silence ثم يليها 32 نبضة مربعة دور كل منها 27µs.





الشكل يبين بروتوكول الإرسال لـ RC5

يتكون بروتوكول RC5 من 14Bits ثنائي على الشكل التالي:

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

- Bits1-2: هي بتات بداية الإرسال (Start Bits or AGC "Automatic Gain Control") وهي دائماً تملك القيمة "1". وهي تساعد في معايرة التحكم الآلي بربح مستقبل الأشعة.
- Bit3: هو بت التحكم CHECK bit (Control Bit or Toggle Bit)، هذا البت تتغير قيمته بين الصفر والواحد منطقي في كل مرة يتم فيها ضغط أحد أزرار التحكم. هذا يفيد جهاز التحكم ليفهم إذا مازلت تضغط على أحد الأزرار ويتكرر الأمر – تصور انك تضغط الرقم واحد وتستمر بالضغط، فلولا هذا البت فإن الجهاز سيفهم انك تريد اختيار القناة 11 بدلاً من القناة واحد لأنه سيرسل قطارين من النبضات لهما القيمة نفسها.
- Bits4-8: هي بتات العنوان، هذه البتات الخمسة تسمح لي باختيار نوع الجهاز الذي يجب أن يستجيب للأوامر، وهي تحقق لي عنونة لـ 32 جهاز (2^5=32) وهي على الشكل التالي:

SYSTEM ADDRESS	EQUIPMENT
0	TV SET 1
1	TV SET 2
2	VIDEOTEXT
3	EXPANSION FOR TV 1 AND 2
4	LASER VIDEO PLAYER
5	VIDEO RECORDER 1 (VCR 1)
6	VIDEO RECORDER 2 (VCR 2)
7	RESERVED
8	SAT 1
9	EXPANSION FOR VCR 1 OR 2
10	SAT 2
11	RESERVED
12	CD VIDEO
13	RESERVED
14	CD PHOTO
15	RESERVED
16	AUDIO PREAMPLIFIER 1
17	RECEIVER / TUNER
18	TAPE / CASSETE RECORDER
19	AUDIO PREAMPLIFIER 2
20	CD
21	AUDIO RACK
22	AUDIO SAT RECEIVER
23	DCC RECORDER
24	RESERVED
25	RESERVED
26	WRITABLE CD
26-31	RESERVED

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

COMMAND (in decimal)	DESCRIPTION of FUNCTION
0-9	NUMERIC KEYS 0 - 9
12	STANDBY
13	MUTE
14	PRESETS
16	VOLUME UP
17	VOLUME DOWN
18	BRIGHTNESS +
19	BRIGHTNESS -
20	COLOR SATURATION +
21	COLOR SATURATION -
22	BASS UP
23	BASS DOWN
24	TREBLE +
25	TREBLE -
26	BALANCE RIGHT
27	BALANCE LEFT
48	PAUSE
50	FAST REVERSE
52	FAST FORWARD-
53	PLAY
54	STOP
55	RECORD
63	SYSTEM SELECT
71	DIM LOCAL DISPLAY
77	LINEAR FUNCTION (+)
78	LINEAR FUNCTION (-)
80	STEP UP
81	STEP DOWN
82	MENU ON
83	MENU OFF
84	DISPLAY A/V SYS STATUS
85	STEP LEFT
86	STEP RIGHT
87	ACKNOWLEDGE
88	PIP ON/OFF
89	PIP SHIFT
90	PIP MAIN SWAP
91	STROBE ON/OFF
92	MULTI STROBE
93	MAIN FROZEN
94	3/9 MULTI SCAN
95	PIP SELECT
96	MOSAIC MULTI PIP
97	PICTURE DNR
98	MAIN STORED
99	PIP STROBE
100	RECALL MAIN PICTURE
101	PIP FREEZE
102	PIP STEP UP
103	PIP STEP DOWN
118	SUB MODE
119	OPTIONS BUS MODE
123	CONNECT
124	DISCONNECT

ربط مستقبل IR إلى معالج مصغر Interfacing IR Receiver to uC.

توضح هذه الفقرة بعض الأمور التي يجب مراعاتها عند وصل مستقبل أشعت تحت الحمراء مع uC أو uP.

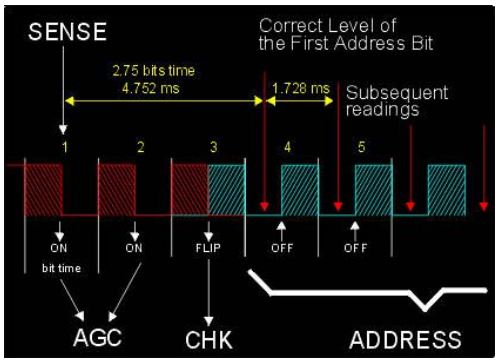
1- انتبه أن مستقبل الأشعة تحت الحمراء سوف يعكس المستوى المنطقي للنبضات — "1" = off | "0"=On.

2- في حال عدم الإرسال (inactivity) فإن خرج المستقبل سيكون على المستوى "1".

3- يمكن ربط خرج المستقبل إلى أي قطب من أقطاب المايكرو أو إلى قطب مقاطعة خارجية ومراقبة حالة القطب حتى تتغير حالته إلى المستوى المنخفض دلالةً على وجود حالة إرسال، حينها تبدأ باستقبال الشيفرة المؤلفة من 14 بت.

4- إن البتات الثلاثة الأولى مهمة فقط لإعلامك ببدء عملية الإرسال، ولكنها غير مهمة بعد ذلك لذلك لا تستقبلها وتقوم بفكها وتحليلها، فقط استقبل البتات الأحد عشر التالية التي تحوي على عنوان الجهاز والأمر المطلوب تبعاً للزر المضغوط على الجهاز، ويكون ذلك بانتظار زمن قدره 4.752mS بعدها يمكنك أن تبدأ عملية القراءة للبتات الأحد عشر مع مراعاة أن الفواصل الزمنية بين كل بت وآخر هي 1.728mS.

الشكل التالي يوضح هذا البند.



مستقبل الأشعة تحت الحمراء CLRM-2038S CLRM-2038S IR Module

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

1- مستقبل أشعة تحت الحمراء ومضخم إشارة في نفس الوقت.

2- مرشح تمرير داخل غلاف المستقبل من أجل ترددات PCM.

3- مناعة عالية ضد التأثر بالأضواء المحيطة.

4- درع مطور للمناعة ضد اضطرابات الحقل الكهربائي.

5- استهلاك طاقة منخفض ضمن مجال العمل 2.7V~5.5V.

6- متوافق مع متطلبات المستوى المنطقي TTL, CMOS.

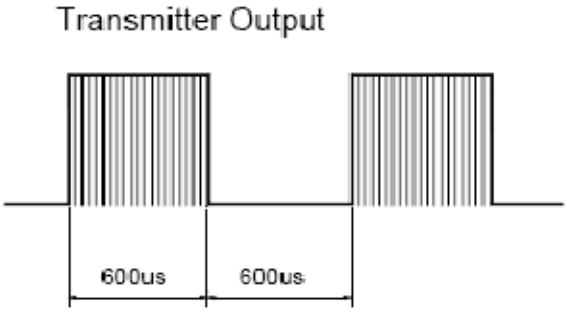
7- متوافق مع معايير NEC code,RC5 code.

8- تردد الحامل 38KHZ.

9- مسافة الاستقبال حتى 12m.

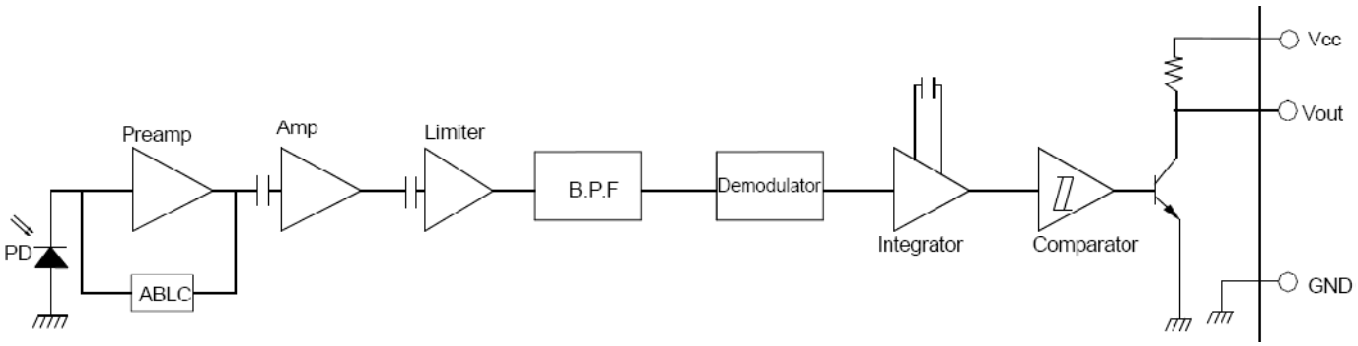
10- يمكن استخدامه من أجل التطبيقات التالية:

- مفتاح ضوئي (Optical switch).
- تطبيقات التحكم بالأجهزة مثل: Audio, TV, VCR, CD, MD, DVD, etc.
- التحكم بالأجهزة المنزلية مثل: Air-conditioner, Fan, CATV, etc

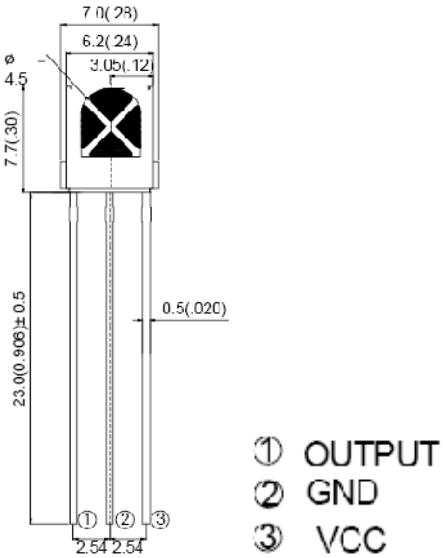


توزع الأقطاب والبنية الداخلية لـ CLRM-2038S Pins Confg & Internal Diagram

الشكل التالي يوضح البنية الداخلية للمستقبل CLRM-2038S.

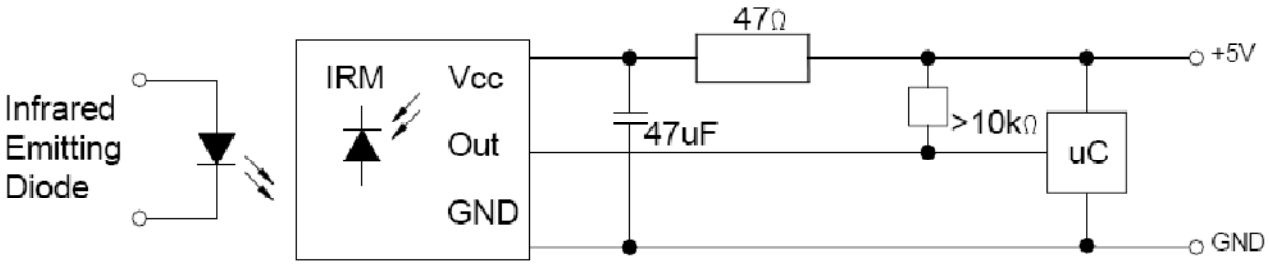


الأشكال التالية توضح الشكل الحقيقي وتوزع الأقطاب للمستقبل CLRM-2038S.



Rc5 receiver interface — دارة الملائمة لمستقبل الأشعة تحت الحمراء

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



IRSAT Remote Control — جهاز التحكم IRSAT

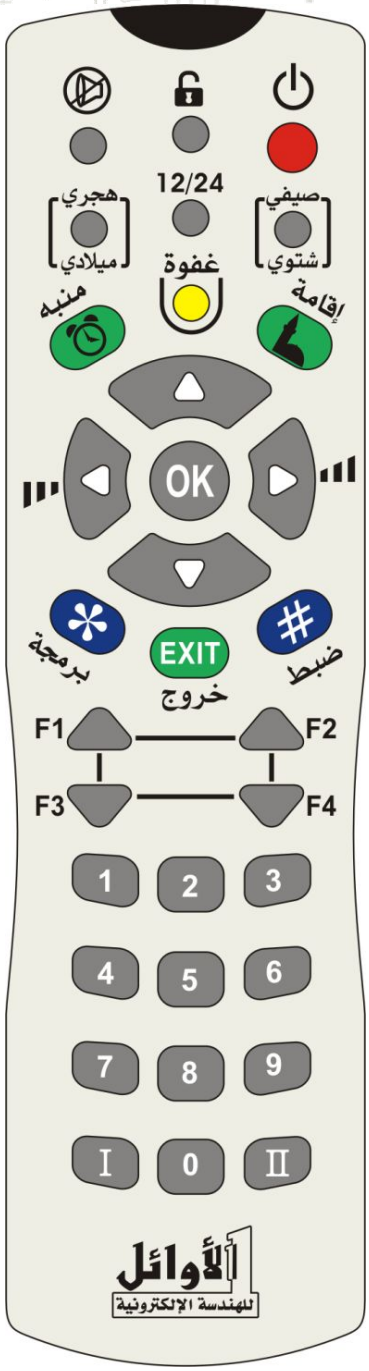
مواصفات الجهاز:

- النموذج: IRSAT RC5 TV Remote Control with 33 Keys
- بلد المنشأ: China (mainland)
- نوع البطاريات: AA x 2
- مجال الإرسال: 10 meters
- Standby current: 3μA.
- الأبعاد: 17x4.7x2.5cm

IRSAT Remote Control Key Commands — عناوين المفاتيح لجهاز التحكم IRSAT

بالنسبة لجهاز التحكم الموضح على الشكل أدناه فإنه مصنع خصيصاً لشركة الأوائل للهندسة الإلكترونية وفق دلائل وظيفية خاصة وموضحة في الجدول المدرج فيما يلي:

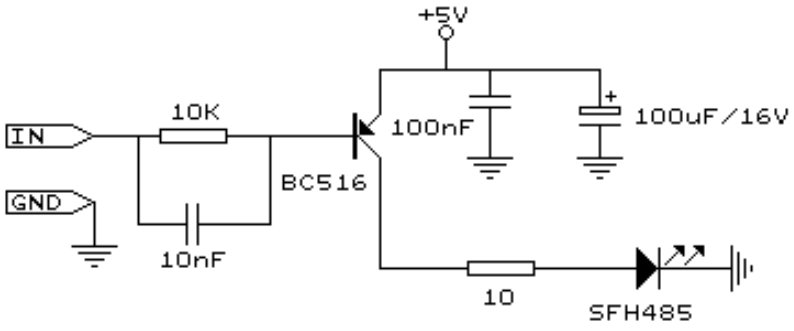
COMMAND (in Hex)	DESCRIPTION of FUNCTION
0-9	NUMERIC KEYS 0 - 9
12	
13	
14	
16	VOLUME UP
17	VOLUME DOWN
18	
19	
20	12/24
21	
22	BASS UP
23	BASS DOWN
24	
25	
26	
27	
48	
50	
52	F1
53	F2
54	F3
55	F4
63	
71	



دارة إرسال بروتوكول RC5

RC5 Sender

المخطط النظري للدارة: قم بتوصيل مرسل أشعة تحت الحمراء إلى القطب OC1(A) مع مراعاة الشكل التالي:



البرنامج باستخدام بيئة Bascom-AVR :

سوف يقوم البرنامج التالي باستدعاء مكتبة RC5 الموجودة في البيئة البرمجية والتي تحوي على بروتوكول الإرسال.

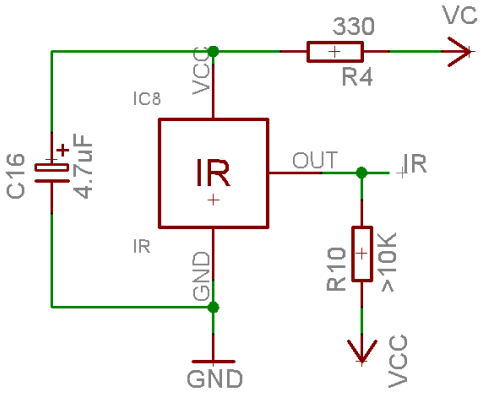
يتم إرسال البروتوكول باستخدام التعليمة RC5SEND والتي تقوم بتشغيل المؤقت Timer1 لحساب زمن النبضات بشكل آلي.

```
$regfile = "m8def.dat"                          ' specify the used micro
$crystal = 4000000                              ' used crystal frequency

Dim Togbit As Byte , Command As Byte , Address As Byte

Command = 12                                    ' power on off
Togbit = 0                                      ' make it 0 or 32 to set the toggle bit
Address = 0

Do
   Waitms 200
   Rc5send Togbit , Address , Command
Loop

End
```

دارة استقبال بروتوكول RC5 RC5 Receiver

المخطط النظري للدارة:

قم بتوصيل خرج مستقبل الأشعة تحت الحمراء إلى أحد أقطاب المايكرو مع مراعاة الشكل التالي:

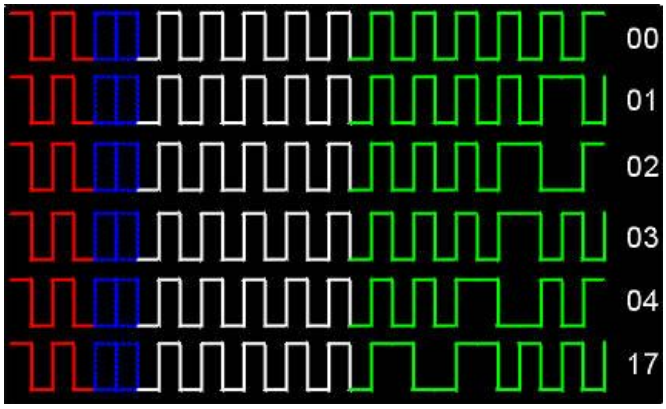


البرنامج باستخدام بيئة Bascom-AVR.

سوف يقوم البرنامج التالي باستدعاء مكتبة RC5 الموجودة في البيئة البرمجية والتي تحوي على بروتوكول الاستقبال.

يتم فحص حالة المستقبل باستخدام التعليمة getrc5 والتي تقوم بتشغيل المؤقت Timer0 لعد النبضات بشكل آلي.

```
$regfile = "m8def.dat"                                    ' specify the used micro
$crystal = 4000000                                        ' used crystal frequency
$baud = 19200                                             ' use baud rate
$lib "mcsbyte.lbx"

Config Rc5 = Pind.2                          ' pin we want to use for the receiver input

Enable Interrupts                             ' enable the interrupts
Dim Address As Byte , Command As Byte        'reserve space for variables

Print "Waiting for RC5..."

Do
  Getrc5(address , Command)                  ' check if a key on the remote is pressed

  If Address = 0 Then                        'we check for the TV address and that is
     Command = Command And &B01111111        'clear the toggle bit (bit7)
     Print Address ; "  " ; Command
  End If
Loop
End
```



Red: AGC pulses (ON)
Blue: Check bit (flipping)
White: Address (00)
Green: Command

برمجة المتحكمات المصغرة

التجارب العملية

الجلسة الخامسة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

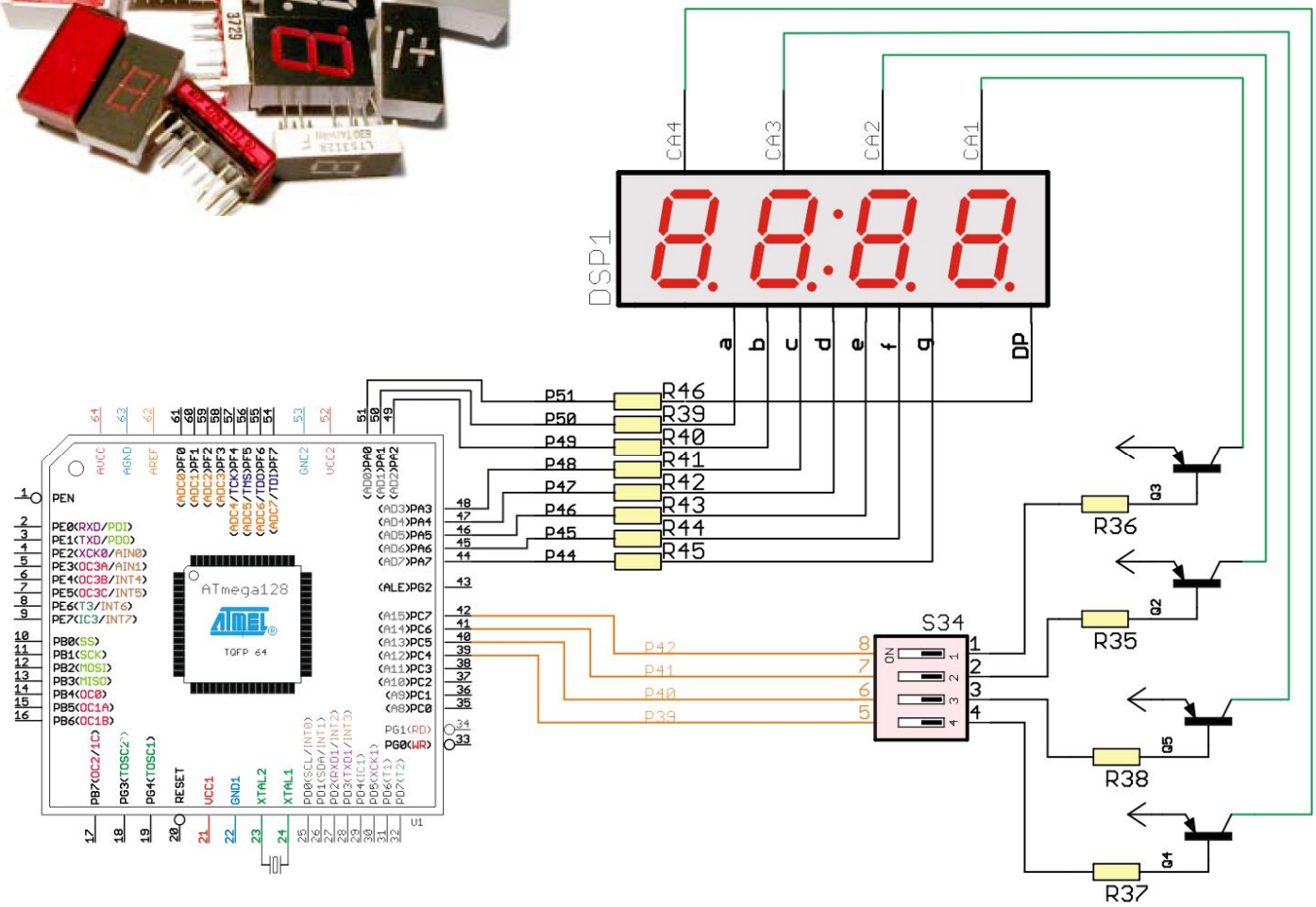
WALID BALID, Tuesday, December 15, 2009

التجربة الرابعة عشرة: لوحات الإظهار ذات السبع قطع
Exp.14: 7-Segment Single Display

الغاية من التجربة:

توصيل وبرمجة لوحة إظهار سباعية (7-Segment).

مخطط التوصيل:



شرح عمل الدارة:

سوف نقوم بكتابة برنامج التشغيل للوحات الإظهار السباعية على ثلاث مراحل:

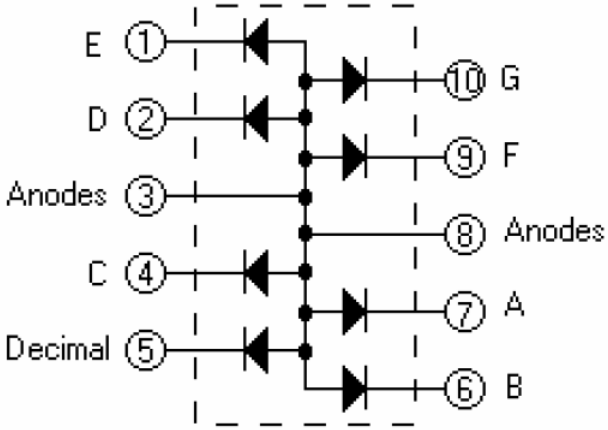
1- تشغيل لوحة إظهار فردية كعداد عشري 9 – 0.

2- تشغيل لوحة إظهار ثنائية كعداد عشري 99 – 0.

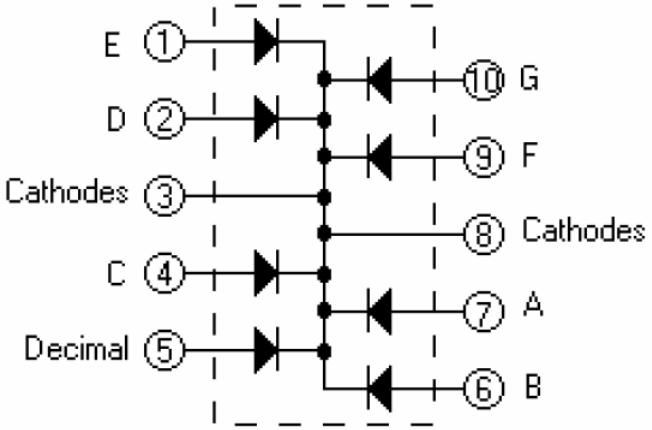
3- تشغيل لوحة إظهار رباعية كعداد عشري 9999 – 0.

إن عملية تشغيل لوحة الإظهار ذات السبع قطع بما في ذلك إظهار الأرقام على اللوحة، يتعلق مباشرة بقيمة تشفير كل رقم من أرقم لوحة الإظهار، ومن أجل معرفة تشفير كل رقم من الأرقم فإنه يجب معرفة بنية لوحة الإظهار ذات السبع قطع.

الشكل التالي يبين المخطط الداخلي للوحات الإظهار ذات السبع قطع.



Common anode



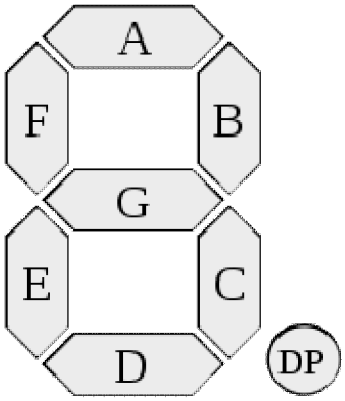
Common cathodes

كما هو واضح من الشكل أعلاه أن لوحة الإظهار تحوي على سبعة ثنائيات ضوئية (ثنائي ضوئي لكل قطعة)، بالإضافة إلى ثنائي ضوئي ثامن للفاصلة العشرية.

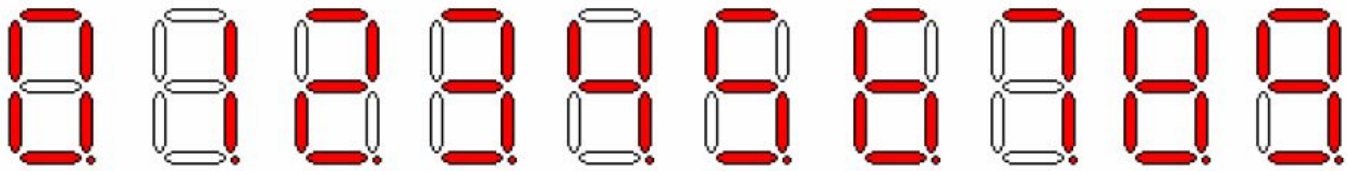
إن جميع الثنائيات الضوئية تكون موصولة بإحدى طريقتين، إما مصعد مشترك (CA) أو مهبط مشترك (CC).

الشكل اليميني يوضح وصلة المهبط المشترك، وأما اليساري فيوضح وصلة المصعد المشترك.

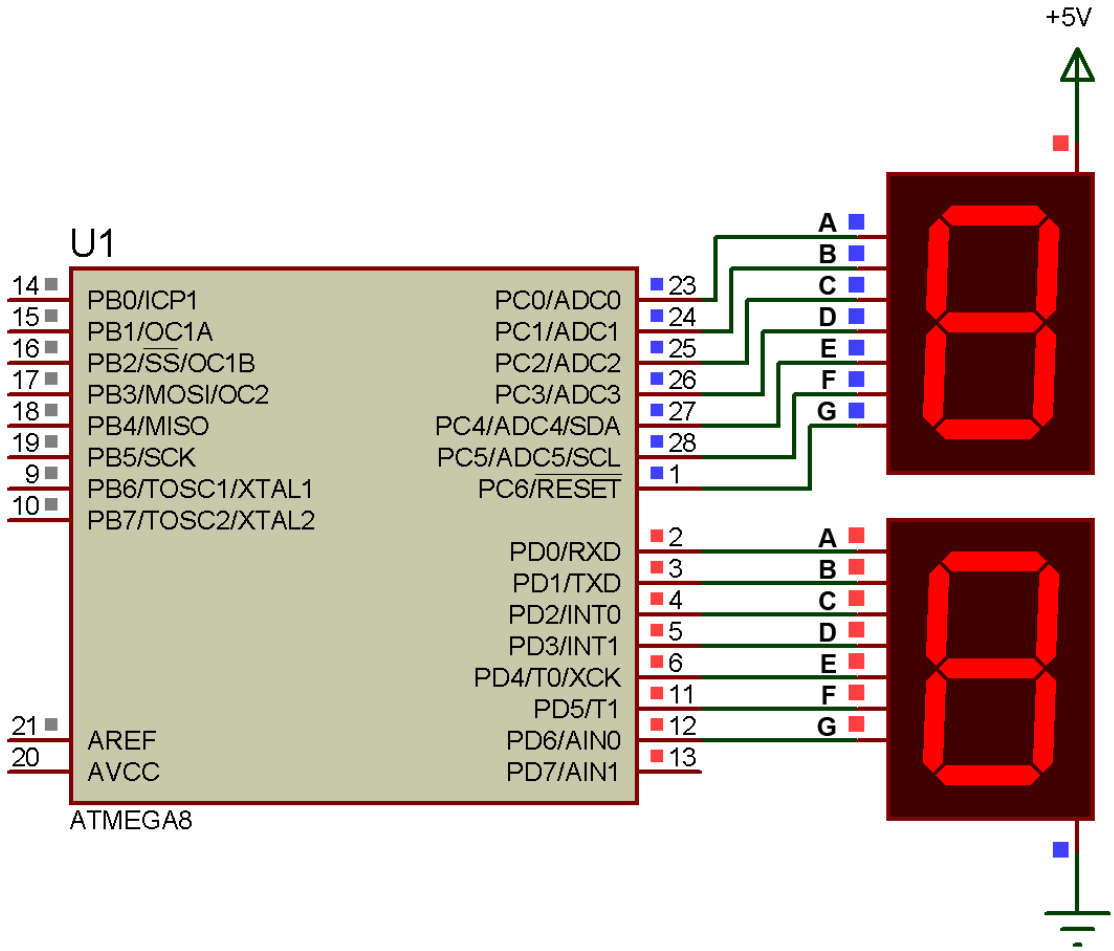
إن القطع السبعة في لوحة الإظهار بالإضافة إلى نقطة الفاصلة العشرية يتم الإشارة إلى كل واحدة منها بحرف (A,.....,H)، إن هذا الترميز لكل قطعة هو ترميز عالمي معتمد من قبل كل الشركات كما في الشكل:



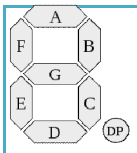
وبالتالي من أجل إظهار الرقم (1) على لوحة الإظهار السباعية، فإنه يجب تشغيل القطعتين B,C وهكذا بالنسبة لباقي الأرقام.



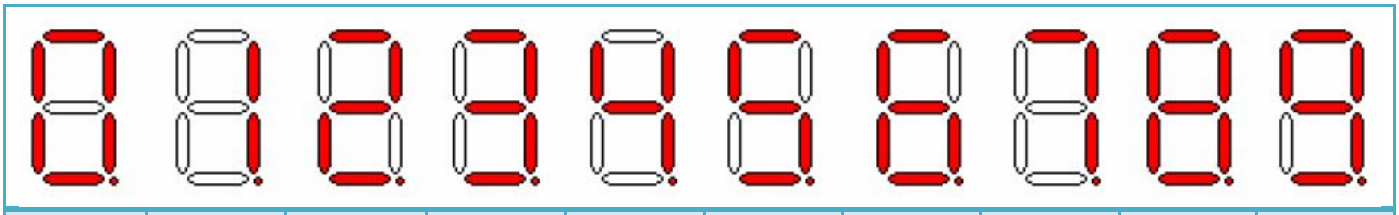
بفرض إنه لدينا شاشة إظهار سباعية ذات مصعد مشترك موصولة مع البوابة PortC لمتحكم مصغر وأخرى ذات مهبط مشترك موصولة مع البوابة PortD كما في الشكل التالي:



وبالتالي فإنه من أجل تشغيل أي قطعة في اللوحة العلوية (ذات المهبط المشترك)، ينبغي وضع صفر منطقي على قطب المتحكم الموصول مع القطعة.

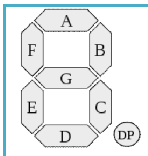
الرقم المطلوب إظهاره	القيمة على البوابة PortC								القطع التي هي في حالة عمل: 
	0-A	1-B	2-C	3-D	4-E	5-F	6-G	7-H	
0	0	0	0	0	0	0	1	1	A.B.C.D.E.F
1	1	0	0	1	1	1	1	1	B.C
2	0	0	1	0	0	1	0	1	A.B.D.E.G
3	0	0	0	0	1	1	0	1	A.B.C.D.G
4	1	0	0	1	1	0	0	1	B.C.F.G
5	0	1	0	0	1	0	0	1	A.C.D.F.G
6	0	1	0	0	0	0	0	1	A.C.D.E.F.G
7	0	0	0	1	1	1	1	1	A.B.C
8	0	0	0	0	0	0	0	1	A.B.C.D.E.F.G
9	0	0	0	0	1	0	0	1	A.B.C.D.F.G

وبالتالي بتحويل القيم الثنائي إلى عشرية أو ست عشرية يمكن أن نحدد مايلي:

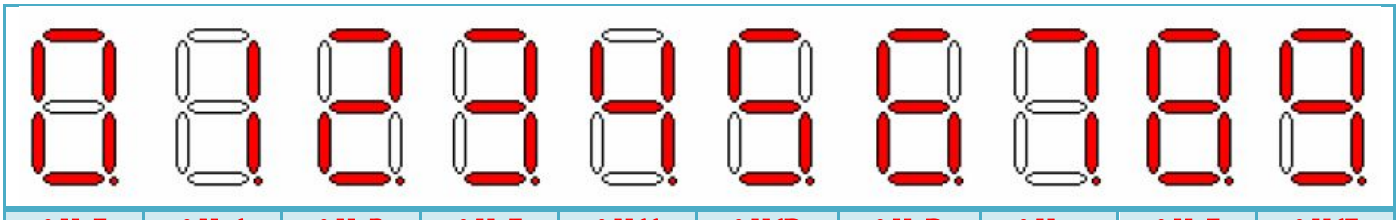


&HC0	&HF9	&HA4	&HB0	&H99	&H92	&H82	&HF8	&H80	&H90

وبالمثل فإنه يمكن حساب القيم من أجل تشغيل اللوحة السفلية (ذات المصعد المشترك)، ولكن الحالة معكوسة تماماً، لأنه ينبغي وضع واحد منطقي على قطب المتحكم الموصول مع القطعة المراد تشغيلها.

القطع التي هي في حالة عمل: 	القيمة على البوابة PortC								الرقم المطلوب إظهاره
	7-H	6-G	5-F	4-E	3-D	2-C	1-B	0-A	
A.B.C.D.E.F	0	0	1	1	1	1	1	1	0
B.C	0	0	0	0	0	1	1	0	1
A.B.D.E.G	0	1	0	1	1	0	1	1	2
A.B.C.D.G	0	1	0	0	1	1	1	1	3
B.C.F.G	0	1	1	0	0	1	1	0	4
A.C.D.F.G	0	1	1	0	1	1	0	1	5
A.C.D.E.F.G	0	1	1	1	1	1	0	1	6
A.B.C	0	0	0	0	0	1	1	1	7
A.B.C.D.E.F.G	0	1	1	1	1	1	1	1	8
A.B.C.D.F.G	0	1	1	0	1	1	1	1	9

وبالتالي بتحويل القيم الثنائي إلى عشرية أو ست عشرية يمكن أن نحدد مايلي:



&H3F	&H06	&H5B	&H4F	&H66	&H6D	&H7D	&H07	&H7F	&H6F

ملاحظة: عند تحويل قيم التشفير بين شاشة إظهار من نوع مصعد مشترك إلى شاشة إظهار من نوع مهبط مشترك فإنه يكفي طرح القيمة من القيمة &HFF (255 DEC).

التعليمات الجديدة:

إن قيم التشفير يمكن تخزينها في البرنامج ضمن ما يسمى بجدول بيانات (LUT: Look-Up Table)، وهناك تعليميتين أساسيتين وبعض التعليمات الأخرى للتخزين والقراءة من جداول البيانات.

التعليمة البرمجية	شرح التعليمة
`Data` var_1 , … , var_n	تخزين بيانات (رقمية، محرفية) في ذاكرة البرنامج عند لافتة معينة، حيث أن العنصر الأول له الدليل 0.
`var = Lookup(Index , label)`	جلب قيمة من LUT مخزن في `Data` عند اللافتة `label` وله الدليل `Index`.
`Var = Lookupstr(Index , Label)`	جلب محرف من LUT مخزن في `Data` عند اللافتة `label` وله الدليل `Index`.
`Var = Lookdown(value,Label,Entries)`	البحث عن دليل (`Index`) قيمة (`value`) مخزن في `Data` عند اللافتة `label` على أن يبتم البحث ضمن عدد محدد من العناصر (`Entries`).
`Restore Label`	تقوم بوضع مؤشر (عداد) بيانات إلى أول عنصر موجود في LUT مخزن في `Data` عند اللافتة `label` من أجل قراءة البيانات باستخدام تعليمة: `Read Var`
`Read Var`	قراءة القيم الموجودة في LUT مخزن في `Data` والتي تم الإشارة إليها باستخدام التعليمة `Restore` وإسنادهم إلى المتحول `Var`.

برنامج تشغيل الدارة (لوحة إظهار فردية كعداد عشري 9-0):

متطلبات التوصيل: يجب إغلاق نقطة الوصل 4 من المفتاح SW34.

```
$regfile = "m128def.dat"
$crystal = 8000000
'---------------------

Config Porta = Output
Segment Alias Porta
Porta = &H81

Config Portc.4 = Output
Ctr1 Alias Portc.4
Reset Ctr1
'---------------------
Dim I As Byte
'---------------------
Do
   For I = 0 To 9
      Segment = Lookup(i , Seg)
      Waitms 1000
   Next I
Loop

End
'---------------------

Seg:
Data &H81 , &HF3 , &H49 , &H61 , &H33 ,
&H25 , &H05 , &HF1 , &H01 , &H21
```

التوجيهات.

تعريف البوابة الموصل معها شاشة الإظهار.
تعيين القيمة الأولية على البوابة (&H81) وهي تظهر الرقم صفر على شاشة الإظهار

تعريف قطب التحكم بالمصعد المشترك للوحة الإظهار

تعريف المتحولات

حلقة البرنامج الرئيسي يتم فيها:
قراءة القيم المخزنة في (Data) عند اللافتة (Seg) باستخدام التعليمة (Lookup)، قيمة تلو لأخرى.

لافتة جدول تخزين البيانات.

برنامج تشغيل الدارة (تشغيل لوحة إظهار ثنائية كعداد عشري 99 – 0):

متطلبات التوصيل: يجب إغلاق نقطتي الوصل 4,5 من المفتاح SW34.

`$regfile = "m128def.dat"` `$crystal = 8000000`	التوجيهات.

`'--------------------`
`Config Porta = Output`
`Segment Alias Porta`
`Porta = &H81`
	تعريف البوابة الموصل معها خطوط البيانات لشاشات الإظهار.
	تعيين القيمة الأولية على البوابة (&H81) وهي تظهر الرقم صفر على شاشة الإظهار

`Config Portc.4 = Output`
`Display1 Alias Portc.4`
 	
`Config Portc.5 = Output`
`Display2 Alias Portc.5`
 	
`Set Display1 : Set Display2`
	تعريف قطب التحكم بالمصعد المشترك للوحة الإظهار الأولى والثانية.
	إطفاء كلا لوحتي الإظهار.

`'--------------------` `Dim I As Byte , J As Byte`	تعريف المتحولات

`'--------------------`
 	
`Do`
`For I = 0 To 9`
`For J = 0 To 9`
`Segment = Lookup(i , Seg)`
`Reset Display1 : Waitms 20`
`Set Display1`
 	
`Segment = Lookup(j , Seg)`
`Reset Display2 : Waitms 20`
`Set Display2`
`Next J`
`Next I`
`Loop`
 	
`End`
	حلقة البرنامج الرئيسي يتم فيها:
	قراءة القيم المخزنة في (Data) عند اللافتة (Seg)
	باستخدام التعليمة (Lookup)، قيمة تلو لأخرى.
	تشغيل لوحة الإظهار الأولى.
	إيقاف لوحة الإظهار الأولى وتشغيل الثانية.
	تكرار العملية...

`'--------------------`
`Seg:`
`Data &H81 , &HF3 , &H49 , &H61 , &H33 ,`
`&H25 , &H05 , &HF1 , &H01 , &H21`
	لافتة جدول تخزين البيانات.

برنامج تشغيل الدارة (تشغيل لوحة إظهار رباعية كعداد عشري 9999 – 0):

```
$regfile = "m128def.dat"
$crystal = 8000000
'---------------------
Config Porta = Output
Segment Alias Porta
Porta = &H81

Config Portc = &B11110000
Display1 Alias Portc.4
Display2 Alias Portc.5
Display3 Alias Portc.6
Display4 Alias Portc.7

Set Display1 : Set Display2
Set Display3 : Set Display4
'---------------------
Dim I As Byte , J As Byte
Dim K As Byte , L As Byte
'---------------------
Do
   For I = 0 To 9
      For J = 0 To 9
         For K = 0 To 9
            For L = 0 To 9
               Segment = Lookup(i , Seg)
               Reset Display1 : Waitms 7
               Set Display1

               Segment = Lookup(j , Seg)
               Reset Display2 : Waitms 7
               Set Display2

               Segment = Lookup(k , Seg)
               Reset Display3 : Waitms 7
               Set Display3

               Segment = Lookup(l , Seg)
               Reset Display4 : Waitms 7
               Set Display4
            Next L
         Next K
      Next J
   Next I
Loop

End
'---------------------
Seg:
Data &H81 , &HF3 , &H49 , &H61 , &H33 ,
&H25 , &H05 , &HF1 , &H01 , &H21
```

التوجيهات.

تعريف البوابة الموصل معها خطوط البيانات لشاشات الإظهار.

تعيين القيمة الأولية على البوابة (&H81) وهي تظهر الرقم صفر على شاشة الإظهار

تعريف أقطاب التحكم بالمصعد المشترك للوحات الإظهار الأربعة.

إطفاء كل لوحات الإظهار.

تعريف المتحولات

حلقة البرنامج الرئيسي يتم فيها:

قراءة القيم المخزنة في (Data) عند اللافتة (Seg) باستخدام التعليمة (Lookup)، قيمة تلو لأخرى.

تشغيل لوحة الإظهار الأولى.

إيقاف لوحة الإظهار الأولى وتشغيل الثانية.

وهكذا....

تكرار العملية...

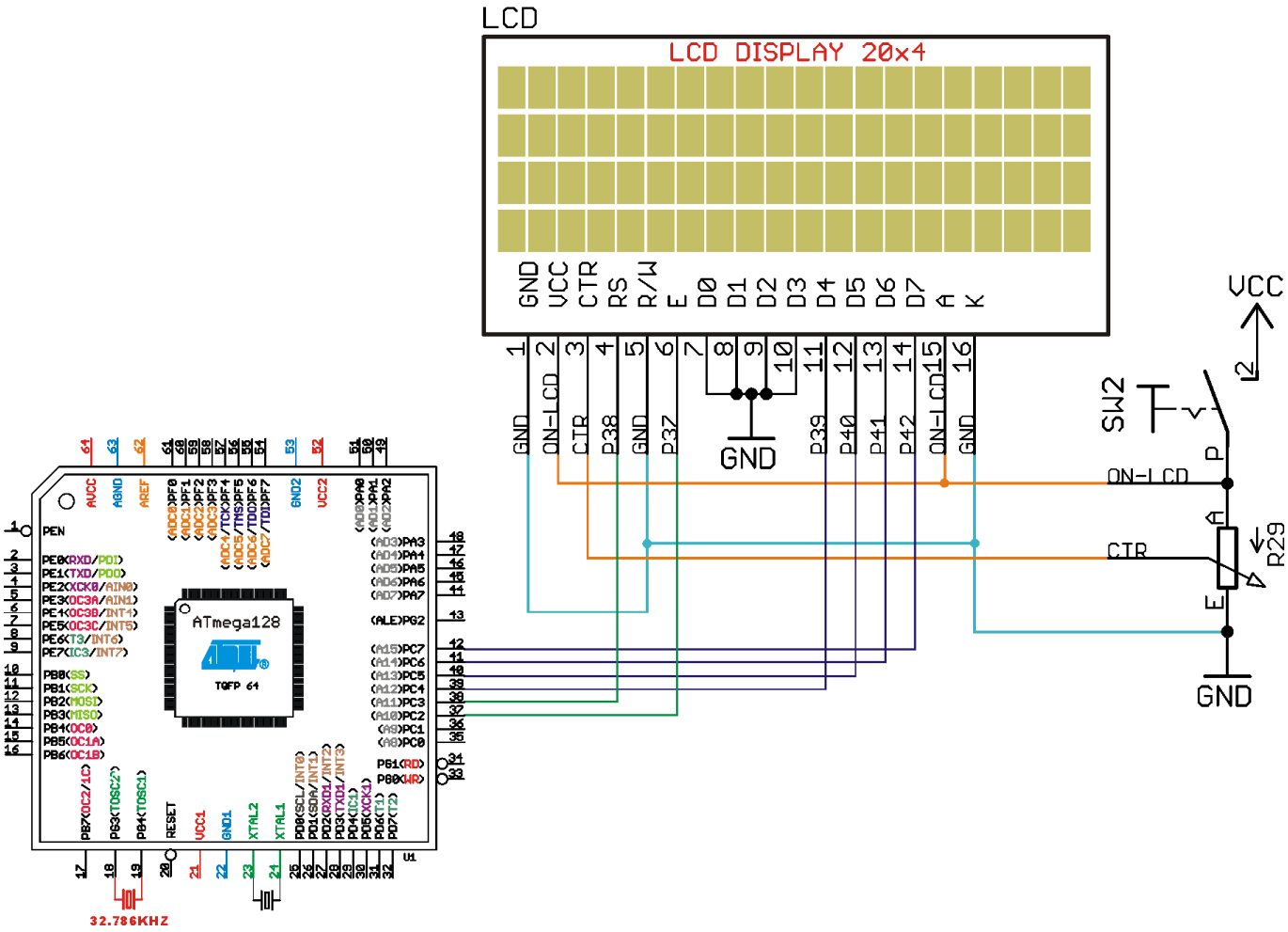
لافتة جدول تخزين البيانات.

التجربة الخامسة عشرة: ساعة توقيت الزمن الحقيقي Exp.15: Real Time Clock

الغاية من التجربة:

برمجة وتشغيل ساعة توقيت وتاريخ في الزمن الحقيقي.

مخطط التوصيل:



متطلبات توصيل:

يجب إغلاق المفتاح SW2.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لتشغيل ساعة توقيت وعرض قيمة التوقيت والتاريخ على شاشة الإظهار الكريستالية.

إنه من أجل تشغيل وحدة التوقيت في الزمن الحقيقي (RTC) الموجودة داخل لمعالج فإنه يجب وصل هزاز كريستالي قيمته 32.786 KHZ على الأقطاب TOSC1,2.

التعليمات الجديدة:

الجدول التالي يوضح التعليمات الجديدة المستخدمة في ساعة الزمن الحقيقي الداخلية.

التعليمة البرمجية	شرح التعليمة
`Config Clock = Soft | User`	تعريف ساعة الزمن الحقيقي (داخلية | خارجي).
`Config Date = Dmy , Separator = Char`	تعيين شكل عرض التاريخ والفواصل بين اليوم والشهر والسنة
`Time$ = "hh:mm:ss"`	تعيين قيمة افتراضية للتوقيت يبدأ منها التوقيت.
`Date$ = "mm/dd/yy"`	تعيين قيمة افتراضية للتاريخ يبدأ منها التاريخ.
`Var = Time$`	قراءة القيمة الحالية للوقت
`Var = Date$`	قراءة القيمة الحالية للتاريخ

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 8000000
'----------------------
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 =
Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E
= Portc.2 , Rs = Portc.3
Config Lcd = 20 * 4
'----------------------
Config Clock = Soft
Config Date = Dmy , Separator = /
Enable Interrupts

Time$ = "16:59:00"
Date$ = "14/04/08"
----------------------
Do
   Print Time$
   Print Date$

   Lcd Time$ : Locate 2 , 1 : Lcd Date$

   Wait 1
   Cls
Loop
```

Directives

Configure the LCD display

تعريف ساعة التوقيت الحقيقي الداخلية للمعالج.

تعريف نمط إظهار اليوم/الشهر/السنة

تعريف القيم الابتدائية (معايرة) للتاريخ والتوقيت

حلقة البرنامج الرئيسي يتم فيها:

إظهار التوقيت والتاريخ على النافذة التسلسلية للمعالج

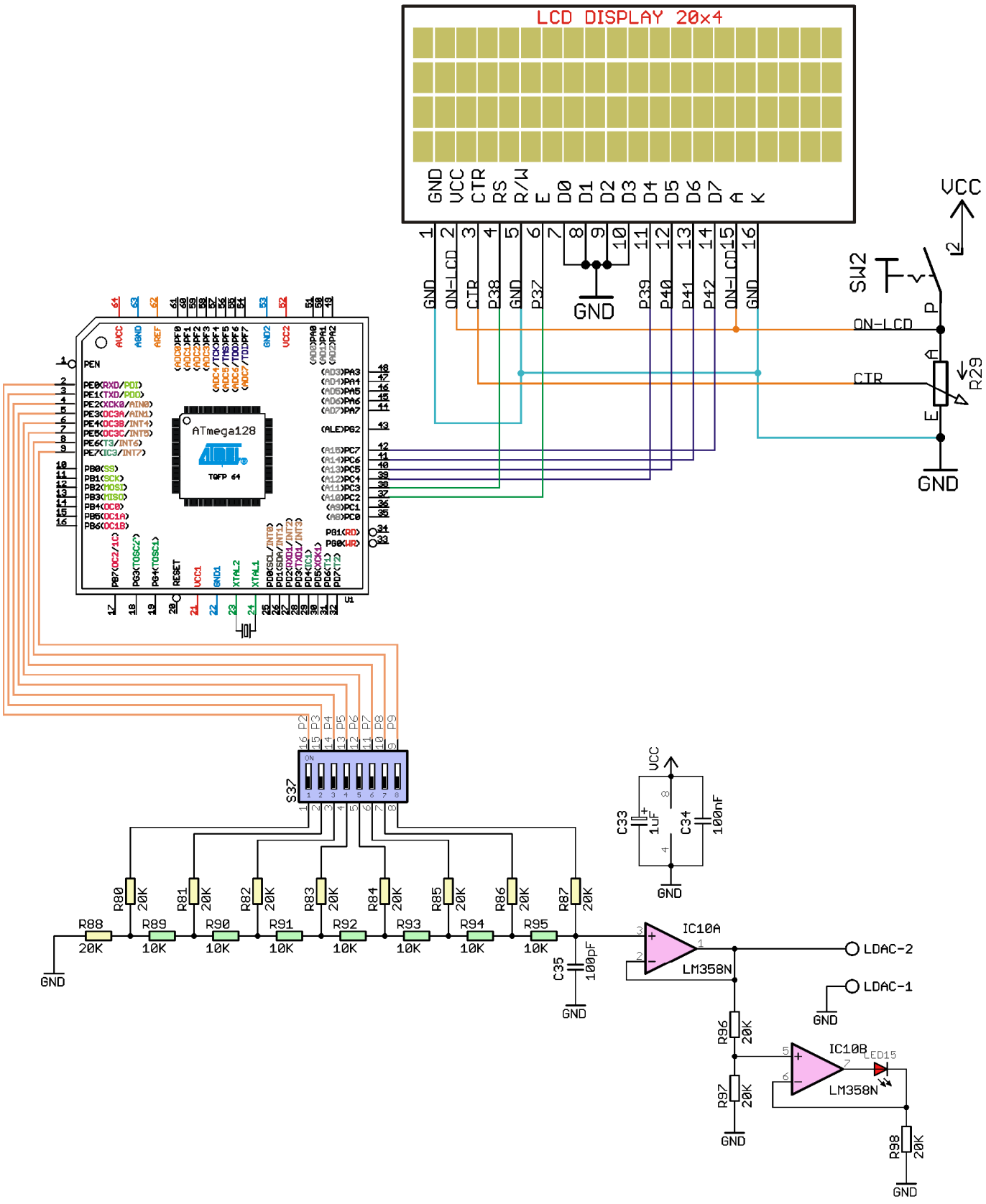
عرض التوقيت والتاريخ على شاشة الإظهار الكريستالية.

التجربة السادسة عشرة: التحويل الرقمي التشابهي
Exp.16: DAC using Ladder R-2R

الغاية من التجربة:

استخدام شبكة لادر R-2R لأغراض التحويل الرقمي التشابهي.

مخطط التوصيل:



متطلبات توصيل:

يجب إغلاق جميع نقاط المفتاح SW37 لوصل شبكة لادر مع المعالج.

يجب إغلاق المفتاح SW2 لتشغيل شاشة الإظهار الكريستالية.

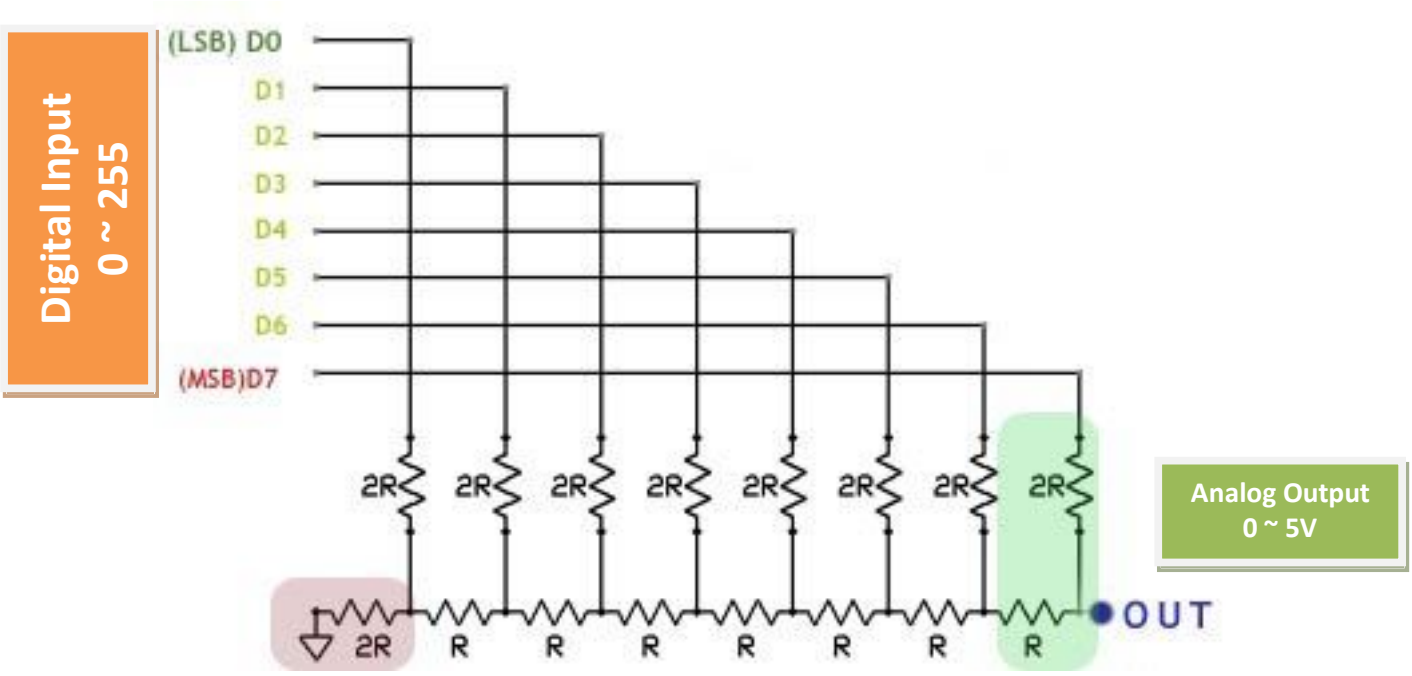
شرح عمل الدارة:

سوف نقوم بكتابة برنامج يقوم بتوليد قيمة رقمية تعبر عن جهد تشابهي يظهر على خرج شبكة لادر، وسوف نقوم بحساب قيمة هذا الجهد وعرضه على شاشة الإظهار الكريستالية.

إن الفكرة الأساسية لشبكة لادر هو إدخال قيم رقمية (0 or 1) على كل مدخل من مداخل الشبكة، والحصول في خرجها على قيمة تشابهية لها علاقة مباشرة بالقيم الرقمية المطبقة على مداخلها.

بشكل عام من أجل شبكة لادر بدقة 8-bit، فإن القيم الرقمية على مداخل الشبكة ستتراوح من 0-255 (00000000>>11111111) والذي بدوره ينتج جهداً تشابهياً خطياً في خرج الشبكة $0\text{-}5V_{DC}$.

$$V_{OUT} = V_{REF} \times VAL / 2^{N}$$



وبالتالي فإنه من أجل شبكة لادر بدقة 8-bit، فإن قيمة كل درجة يمكن حسابه بالعلاقة التالية:

$$V_{step} = V_M / 2^{N} = 5 / 255 = 0.019 \text{ Volt}$$

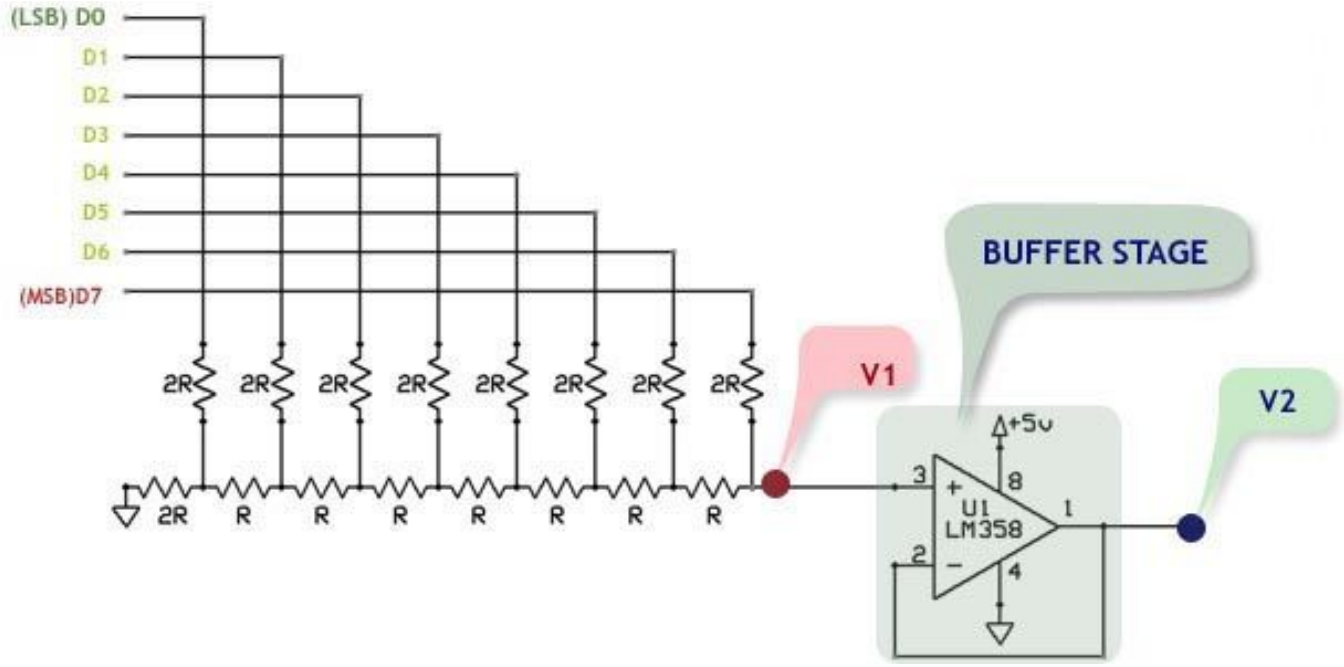
وبالتالي فإن جهد خرج الشبكة سيكون عبارة عن القيمة الرقمية على دخل الشبكة مضروباً بجهد الدرجة الواحد المحسوب أعلاه.

$$V_{out} = Dig_{val} \times V_{step}$$

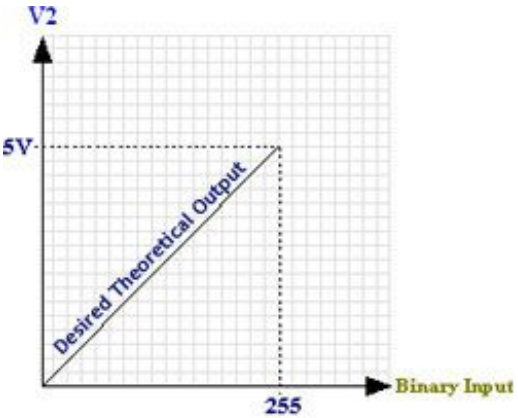
وبالتالي من أجل قيمة رقمية على دخل الشبكة ولتكن 129=0001 1000 فإن جهد الخرج يجب بالشكل:

$$129 \times 0.019 = 2.451V$$

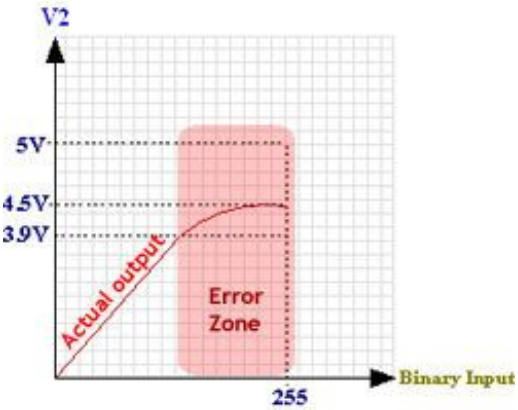
من أجل توصيل خرج شبكة لادر إلى تطبيقات حقيقية فإنه يجب إضافة دارة عزل في خرج الشبكة كما في الشكل التالي:



إن الغاية من دارة العزل هو الفصل بين الجهد في خرج الشبة (V1) والجهد الموصول مع الحمل (V2) بحيث أن استجرار التيار يكون من العازل ولا يكون من الشبكة كي لا يؤثر على عملها. ولكن سيظهر لدينا مشكلة حقيقية (عدم الخطية) نتيجة استخدام المضخم كعازل في الدارة السابقة.



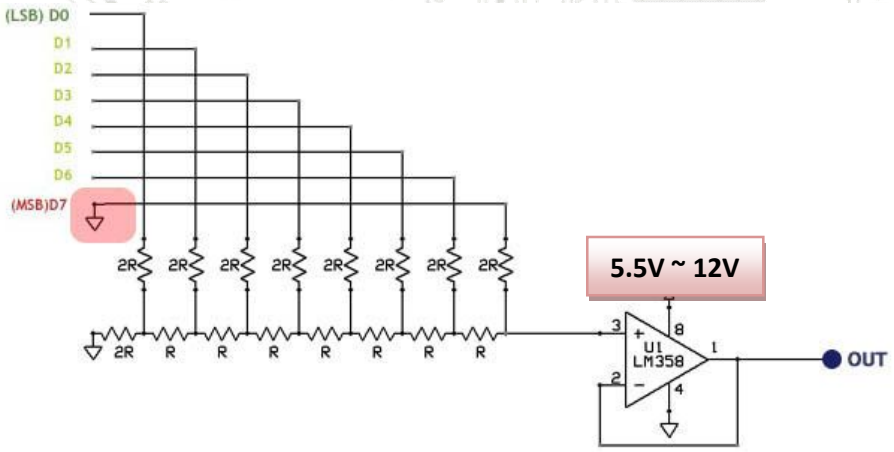
1- في حال كون جهد تغذية المضخم مساوٍ لمجال الجهد الأعظمي على دخله، فإن جهد خرج المضخم لن يكون على كامل مجال جهد الدخل للمضخم (0-to-Vcc)، وإنما سيكون أقل بقليل (Vcc - 0.5V) أي حوالي 4.5V.



2- كما أن خرج المضخم غير خطي على كامل مجال جهد الخرج للمضخم، وإنما سيكون خطياً حتى عتبة معينة تبعاً لمواصفات المضخم نفسه (في مثالنا هذا 3.9V).

يمكن التخلص من المشكلة الأولى بزيادة جهد التغذية للمضخم إلى 6.5V بدلاً من 5V.

أما المشكلة الثانية فيمكن حلها بوصل النقطة D7 إلى نقطة صفرية وبالتالي يصبح مجال الخرج 0 to 2.5V فقط.



التعليمات الجديدة:

ليس هناك تعليمات جديدة في هذه التجربة ولكن من الجيد ذكر بعض تعليمات التقريب الرياضية.

التعليمة البرمجية	شرح التعليمة

`Var = Round(x)`	إرجاع القيمة (x) مقربة إلى أقرب قيمة صحيحة. مثال:
	`Round(2.3) = 2 , Round(2.8) = 3`
	`Round(-2.3) = -2 , Round(-2.8) = -3`

`Var = Int(source)` Ex: Int(134.567)=134	إرجاع القسم الصحيح لعدد كسري
`Var = Frac(source)` Ex: Frac(134.567)=567	إرجاع القسم الكسري لعدد كسري
`Var = Fix(x)` EX: Fix(12.98)=12	تقريب عدد كسري (x) إلى القيمة الصحيحة الأصغر.
`Var = Sgn(x)`	تحديد إشارة العدد (x). إذا كانت **x<0** فإن **Var=-1** إذا كانت **x>0** فإن **Var=1**

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 8000000
'-----------------------
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 =
Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E
= Portc.2 , Rs = Portc.3
Config Lcd = 20 * 4
'-----------------------
Config Porte = Output
Digital Alias Porte
'-----------------------
Dim I As Byte , V_out As Single
Const Vlot_step = 5 / 255
'-----------------------
Cls
Do
   For I = 0 To 255
      Digital = I
      V_out = I * Vlot_step
      Lcd "DAC OUT= " ; V_out
      Wait 1 : Cls
   Next I
Loop
End
```

التوجيهات.

تعريف أقطاب شاشة الإظهار الموصلة مع المتحكم.
تعريف موديل شاشة الظهار

تعريف البوابة الموصولة مع شبكة لادر.

تعريف الثوابت والمتحولات.

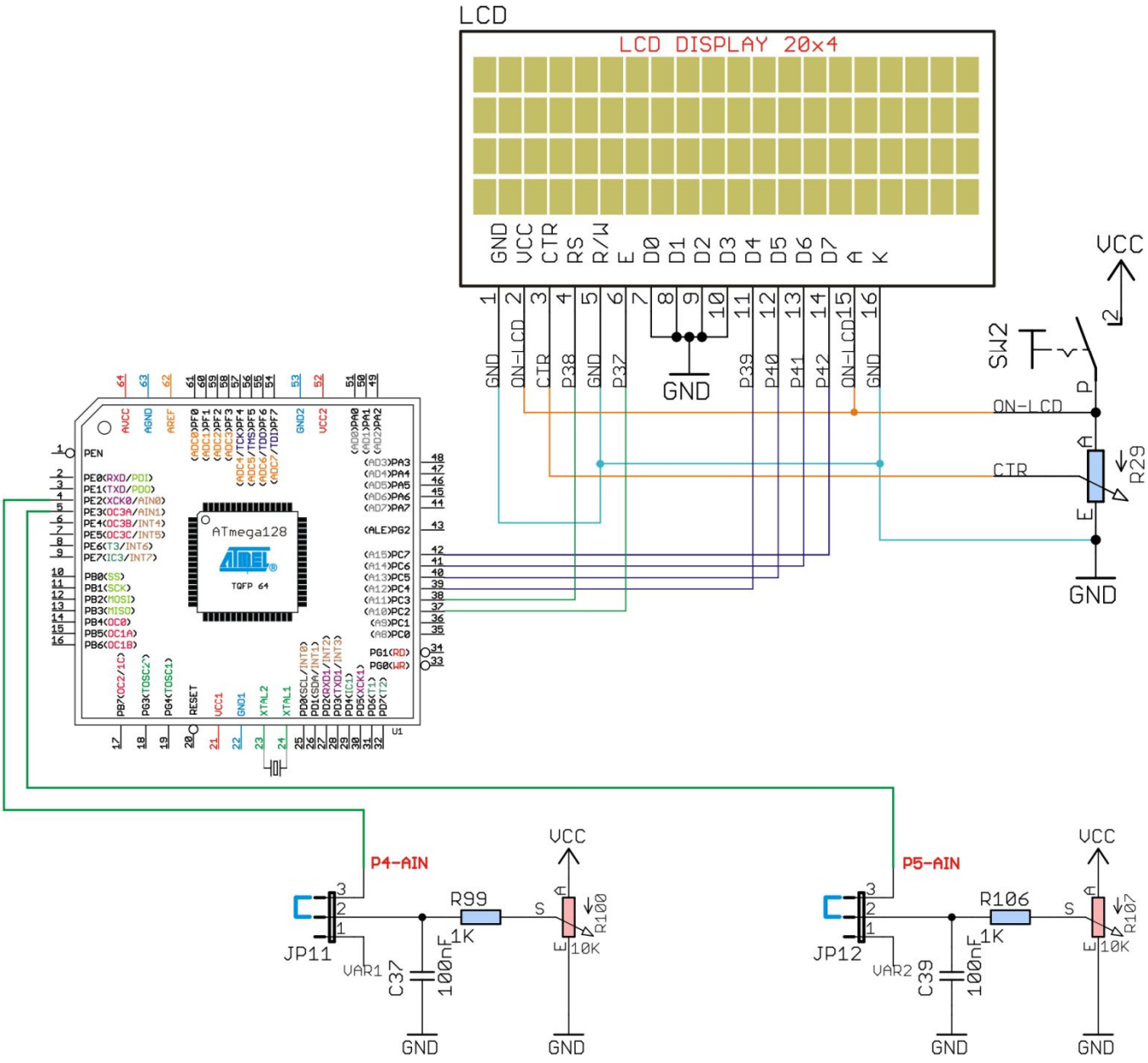
حلقة البرنامج الرئيسي يتم فيها:
توليد قيم رقمية على خرج البوابة (دخل شبكة لادر)
والتي ينشأ عنها جهد تشابهي على خرج الشبكة.
حساب قيمة جهد الخرج التشابهي وعرضها على شاشة
الإظهار الكريستالية.

التجربة السابعة عشرة: المقارن التشابهي
Exp.17: Analog Comparator

الغاية من التجربة:

استثمار المقارن التشابهي الداخلي للمعالج في أغراض مقارنة الإشارة التشابهية.

مخطط التوصيل:



متطلبات توصيل:

يجب إغلاق المفتاح SW2.

يجب وضع نقطتي الوصل (Jumpers) JP11, JP12 ليت وصلها مع أقطاب المقارن التشابهي.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لتشغيل المقارن التشابهي للمعالج من أجل مقارنة إشارتين تشابهيتين مختلفتين.

في حال كون الجهد التشابهي المطبق على القطب **ANI0** (المدخل الموجب للمقارن) أكبر من الجهد المطبق على القطب **AIN1** (المدخل السالب للمقارن) فإن خرج المقارن يصبح في المستوى **High**، وأما في حال كون الجهد على القطب **ANI1** أكبر من الجهد المطبق على القطب **AIN0** فإن خرج المقارن يصبح في المستوى **Low**.

يمكن استخدام هذا المقارن في تطبيقات قياس ارتفاع أو انخفاض مستوى السوائل في خزان من أجل تشغيل أو إيقاف مضخة المياه.

إن خرج المقارن يمكن أن يستخدم لقدح مقاطعة حادثة المسك للمؤقت/عداد1 كما أن المقارن التشابهي يملك مقاطعة خاصة يمكن إعدادها لتكون على الجبهة الصاعدة أو الهابطة أو تغير الجبهة أثناء المقارنة.

التعليمات الجديدة:

الجدول التالي يوضح التعليمات الجديدة المستخدمة في ساعة الزمن الحقيقي الداخلية.

التعليمة البرمجية	شرح التعليمة
`Config Aci= On|off , Compare= On|off , Trigger = TOGGLE|RISING|FALLING`	(**Aci=On|off**): وصل | فصل المقارن التشابهي عن أقطاب نافذة المعالج. (**Compare=On|off**): من أجل "" فإن المؤقت1 في نمط حادثة المسك سيقوم بقدح المقارن التشابهي. (**Trigger**): يوجد ثلاث حالات تتفعل عندها مقاطعة المقارن: `Rising: (Ain0 > Ain1)` سوف تحدث مقاطعة عندما يصبح الجهد على القطب "`Ain0`" أكبر من الجهد على القطب "`Ain1`". `Falling: (Ain0 < Ain1)` سوف تحدث مقاطعة عندما يصبح الجهد على القطب "`Ain0`" أصغر من الجهد على القطب "`Ain1`". `Toggle: (Ain0 > Ain1 and Ain0 < Ain1)`
`Enable Aci`	تفعيل مقاطعة المقارن التشابهي.
`On Aci label`	في حال حصلت مقاطعة المقارن التشابهي إقفز إلى برنامج المقاطعة الموجود عن اللافتة "`label`".
`Start Ac`	تغذية المقارن التشابهي وبدأ عمله.
`Stop Ac`	إيقاف المقارن التشابهي وفصله عن التغذية.
`Enable Interrupts`	تفعيل علم المقاطعات العام الذي لابد منه عند استخدام أي مقاطعة لأي عنصر في المعالج.
`nop`	هذه التعليمة من تعليمات لغة التجميع "Assembly" وهي تأخير زمني بمقدار دورة آلة واحدة.

برنامج تشغيل الدارة:

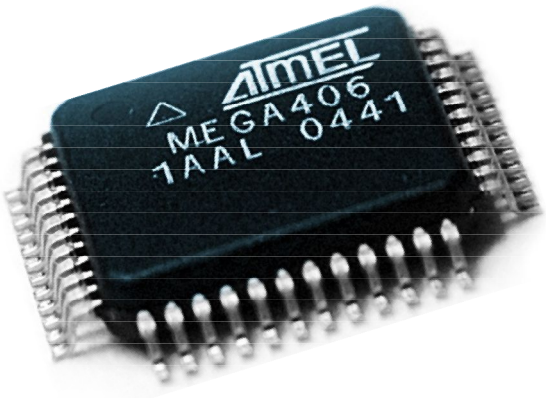
التوجيهات.	`$regfile = "m128def.dat"` `$crystal = 8000000`

تعريف أقطاب شاشة الإظهار الموصلة مع المتحكم.
تعريف موديل شاشة الظهار
	`'-----------------------`
	`Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.2 , Rs = Portc.3`
	`Config Lcd = 20 * 4`

إعداد المقارن التشابهي بحيث يقدح عن تغير الجهد على مدخليه.
القفز إلى برنامج خدمة المقاطعة للمقارن عند حصول المقاطعة.
تشغيل المقارن التشابهي
	`Config Aci = On , Compare = Off , Trigger = Rising`
	`On Aci Isr_aci`
	`Enable Aci`
	`Start Ac`
	`Enable Interrupts`

حلقة البرنامج الرئيسي وهي حلقة فارغة يمكن كتابة أي روتين مرغوب.	`'-----------------------`
	`Dim I As Byte`
	`'-----------------------`
	`Cls`
	`Do`
	`  nop`
	`Loop`
	`End`

برنامج خدمة مقاطعة المقارن التشابهي
يعد مقاطعات المقارن التشابهي ويعرضها على شاشة الإظهار
	`'-----------------------`
	`Isr_aci:`
	`  Cls`
	`  I = I + 1`
	`  Lcd "change: " ; I`
	`  Waitms 100`
	`Return`

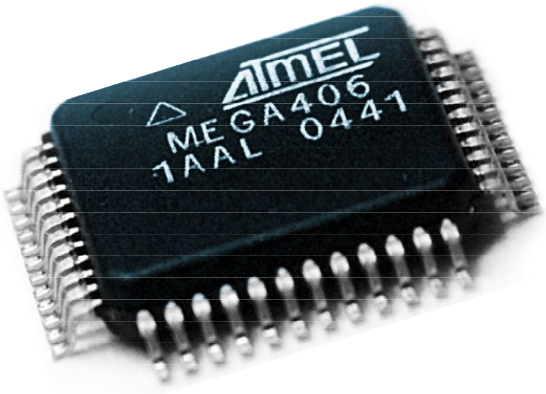


برمجة المتحكمات المصغرة

التجارب العملية

ربط مخارج المتحكم المصغر إلى العالم الخارجي

Interfacing with Microcontrollers



BASCOM-AVR IDE
MCS Electronics

Programming Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

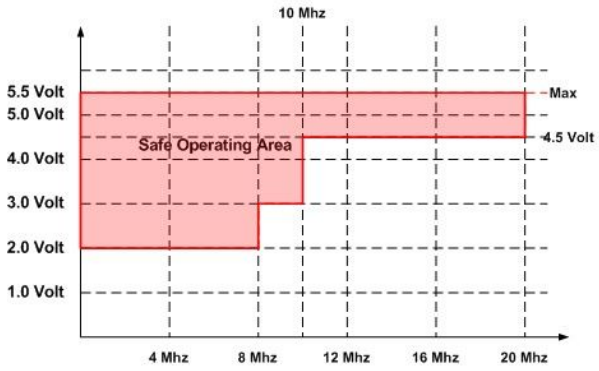
WALID BALID, Tuesday, December 15, 2009

تغذية المتحكم المصغر
Powering Microcontroller

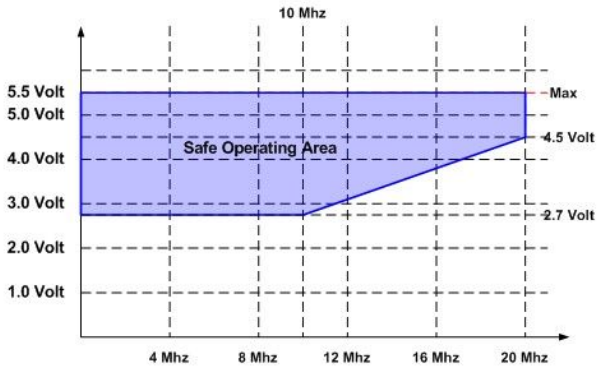
بقدر ما تكون التغذية الرئيسية – لأي دارة إلكترونية – مصممة بشكل جيد وفق اعتبارات تصميمية قياسية، بقدر ما يكون عمل العناصر الإلكترونية في الدارة مستقراً وقريباً من منحني العمل الأمثلي.

إن التغذية الكهربائية التي توصل للمتحكم المصغر هي بمثابة الروح التي تبث الحياة والحركة في المتحكم المصغر، كما أن استهلاك التغذية في المتحكم يتعلق مباشرة بسرعة عمل المتحكم المصغر، حيث أنه كلما ازداد تردد عمل المعالج، ازداد استهلاك التغذية في المعالج.

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



Typical Microchip PIC Microcontrollers Operating Voltage & Frequency
(12F683,16F690, 16F886)



Typical Atmel AVR Microcontrollers Operating Voltage & Frequency
(Tiny13, Tiny2313, Mega168P)

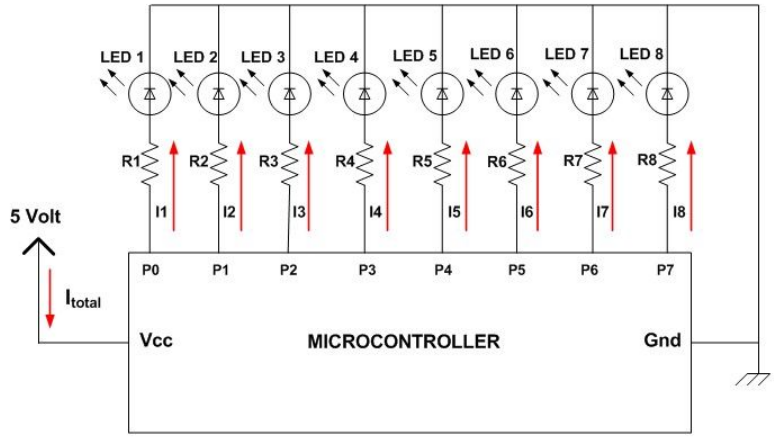
Absolute Maximum Ratings*

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

أحد أهم الاعتبارات التي يجب أن تؤخذ بعين الاعتبار عن ربط أقطاب المتحكم إلى الأحمال هو التيار الأعظمي المستهلك من قطب المتحكم (Vcc to Gnd). إن قيمة التيار التي يمكن سحبها أو تصريفها لقطب دخل\خرج من أقطاب المتحكم تتراوح عادة من 20~40mA حسب المواصفات الكهربائية للمتحكم المصغر. كما أن التيار الأعظمي الذي يمكن سحبه أو تصريفه عن طريق المتحكم بشكل كلي هو 200mA. الشكل جانباً يوضح المواصفات الكهربائية لمتحكمات العائلة AVR.

إن التيار الأعظمي الذي يمكن استجراره من المتحكم هو مجموع تيارات الأقطاب وتيار التشغيل للمتحكم، وإن زيادة التيار فوق الحدود العظمى سوف يؤدي إلى عطل دائم في المتحكم ويتوجب بعدها تغييره.

على الشكل التالي تم استخدام ثمانية أقطاب من متحكم مصغر كأقطاب خرج لتشغيل ثنائيات ضوئية.



Typical LEDs Display on the Microcontroller I/O Ports

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

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

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

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

كما أن تيار عمل المتحكم هو 2.4mA وبالتالي يمكن حساب التيار الكلي من العلاقة:

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

كما هو واضح فإن هذه القيمة تقترب من القيمة العظمة للتيار المسموح استجراره من متحكمات العائلة AVR والذي هو 200mA، بينما تفوق القيمة العظمة للتيار المسموح استجراره من متحكمات العائلة PIC والذي هو 90mA.

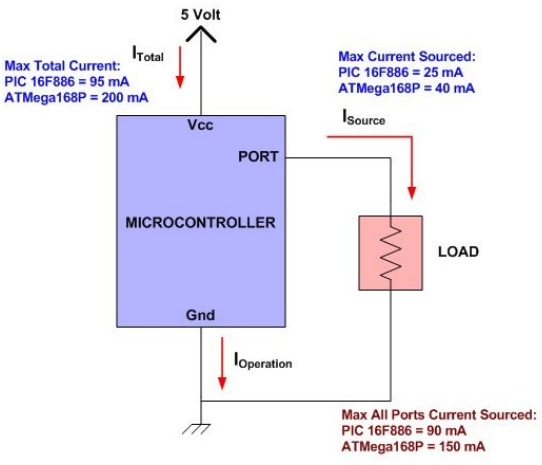
وبالتالي فإن حساب التيارات المسحوبة من أقطاب المتحكم المصغر يعتبر من أهم الأمور التي يجب دراستها في بداية أي مشروع يعتمد على المتحكم المصغر وهو ما سوف نناقشه فيما يأتي.

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

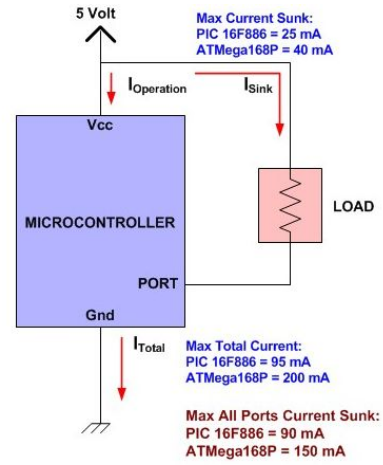
إن وصل الأحمال مع أقطاب المتحكم يكون بطريقتين:

- القطب يعمل كمنبع لتيار تشغيل الحمل (Source).
- القطب يعمل كمصرف لتيار تشغيل الحمل (Sink).

الشكل جانباً يوضح التوصيل.



A. Microcontroller's Port is used as a Current Source



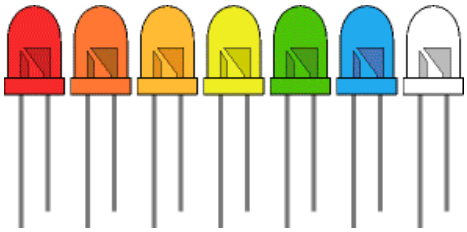
B. Microcontroller's Port is used to Sink Current

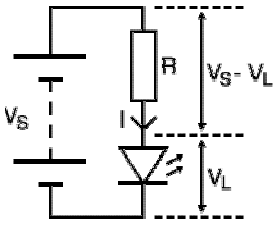
Calculating Current Resistor Value
حساب قيمة واستطاعة مقاومة تحديد التيار

إن قيمة مقاومة تحديد التيار للحمل تتعلق مباشرة بجهد تشغيل الحمل وتياره ومقاومته الأمامي. من أجل حساب قيمة مقاومة تحديد التيار لثنائي الانبعاث الضوئي (LED)، فإنه يجب معرفة تيار وجهد التشغيل للثنائي.

إن تيار وجهد العمل للثنائيات الضوئية يختلف حسب لون الثنائي الضوئي، الجدول التالي يوضح المواصفات الكهربائية:

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





IF max: التيار الأعظمي الأمامي المار في الثنائي.

VF typ: الجهد الأمامي النموذجي من اجل تشغيل الثنائي.

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

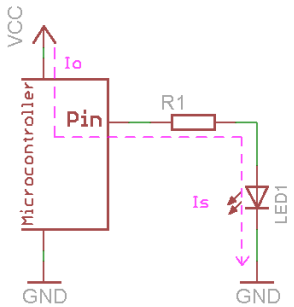
VR max: الجهد العكسي الأعظمي الذي يمكن للثنائي أن يتحمله.

Luminous intensity: شدة السطوع للثنائي.

Viewing angle: زاوية انعكاس الرؤية للإضاءة.

Wavelength: طول موجة الضوء الصادر.

وبالتالي من أجل ثنائي ضوئي ذو لون أحمر فإن جهد وتيار العمل هو 2V/20mA، وبالتالي يمكن حساب مقاومة تحديد التيار من العلاقة:



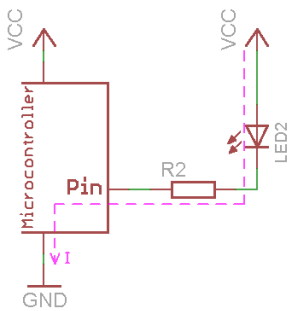
المتحكم هو منبع للتيار

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

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

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

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

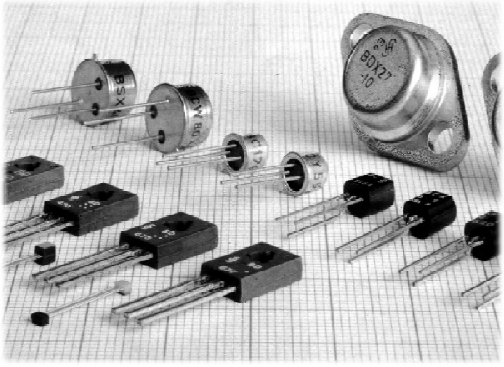


المتحكم هو مصرف للتيار

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

Using Transistors as Control Switches
استخدام مفاتيح التحكم الترانزستورية

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



بشكل عام يوجد نوعين من الترانزستورات:

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

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

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

إن مجال استخدام الترانزستورات في أنظمة التحكم الرقمي يقتصر على استخدام هذه الترانزستورات كمفاتيح إلكترونية تحكمية (On/Of)، وبالتالي فإن اختيار الترانزستور سيعتمد بالكلية على ثلاث عوامل أساسية:

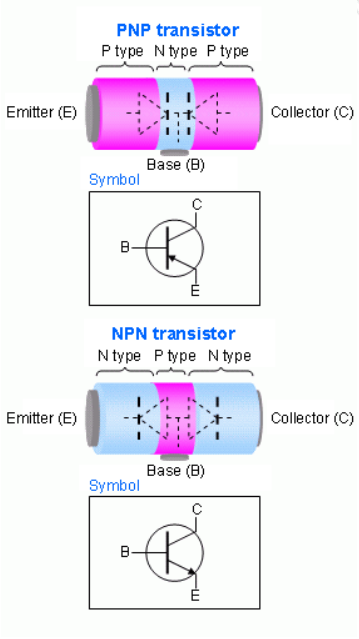
التيار المار في الترانزستور. الاستطاعة المبددة في الترانزستور. سرعة الفتح والإغلاق للترانزستور.

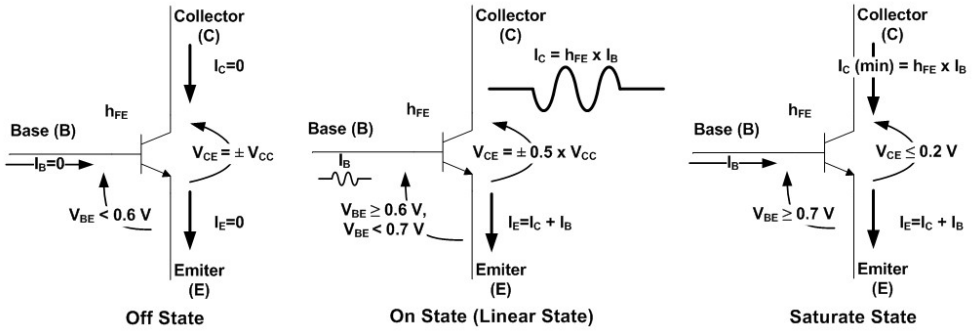
من أجل التحكم بأحمال ذات تيارات صغيرة، يتم استخدام الترانزستورات ثنائية القطبية.

من أجل التحكم بأحمال ذات تيارات وجهود متوسطة كبيرة، تستخدم الترانزستورات الحقلية.

من أجل استخدام الترانزستور كمفتاح إلكتروني فإنه يجب أن يعمل في منطقتي القطع والإشباع.

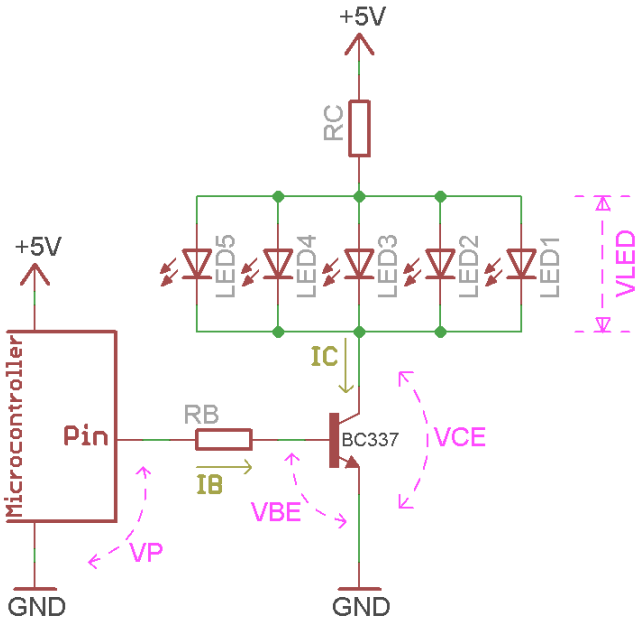
- في حالة القطع (Of state): يكون تيار القاعدة IB=0.
- في الحالة الفعالة (On active state): يكون فيها تيار المجمع $I_C = I_B \times h_{FE}$ وهي الحالة التي يستخدم فيها الترانزستور كمضخم إشارة فعال، وإن أي زيادة في تيار القاعدة ينتج عنه زيادة في تيار المجمع.
- في حالة الإشباع (On saturate state): في هذه الحالة يمرر الترانزستور كامل التيار.





الشكل أعلاه يوضح الحالات الثلاث المذكورة آنفاً لعمل الترانزستور، وما يقابل كل حالة من شروط للجهد والتيار.

الآن لنأخذ مثالاً عملياً ونحسب قيم المقاومات والتيارات كما في الشكل التالي:



إن الترانزستور الذي قمنا باختياره BC337 له المواصفات:

I_{C_max}=800mA, $V_{BE\_saturate}$=0.65V, $V_{CE\_saturate}$=0.2V,

h_{FE} = 100, V_{CE_max}=50V

لنحسب تيار الحمل (IC) مع العلم أن تيار كل ثنائي ضوئي هو: I_{LED}= 20mA وجهد عمل الثنائي هو: V_{LED}=2V.

$$I_C = 5 \times 20mA = 100mA$$

$$Rc = \frac{V_{cc} - V_{LED}}{I_C} = \frac{5-2}{20} = 150\Omega$$

$$P_{RC} = (V_{cc} - V_{LED}) \times I_C$$

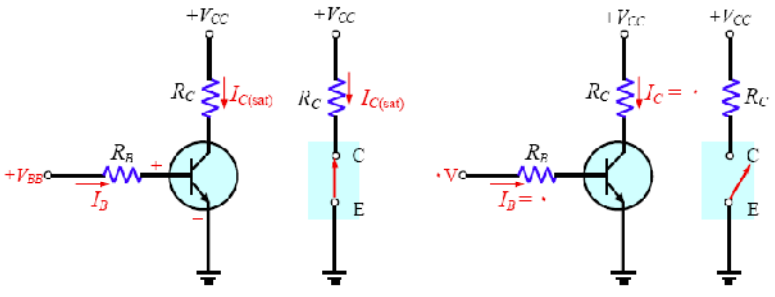
$$P_{RC} = (5-2) \times 100 = 300mW$$

لنحسب قيمة التيار الأصغري اللازم لقيادة الترانزستور عن طريق بوابة المتحكم:

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

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

بالتالي يمكننا حساب قيمة مقاومة القاعدة واستطاعتها من العلاقة التالية:



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

$$P_{RC} = (V_P - V_{BE}) \times I_B$$

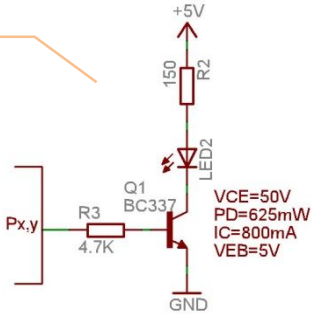
$$P_{RC} = (5-0.7) \times 1 = 4.3mW$$

أولاً: مفاتيح التحكم الترانزستورية باستخدام الترانزستورات ثنائية القطبية (BJT).

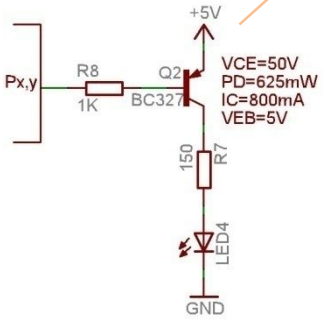
يمكن توصيل المفاتيح الترانزستورية بطريقتين:

- متحكم بها لتكون فعالة عند المنطق العالي "**1**": وبالتالي فإن الترانزستور سوف يعمل كمفتاح لوصل/فصل النقطة الأرضية (GND) للحمل، وفي هذه الحالة سوف يستخدم ترانزستور من نوع NPN.

الشكل (1)



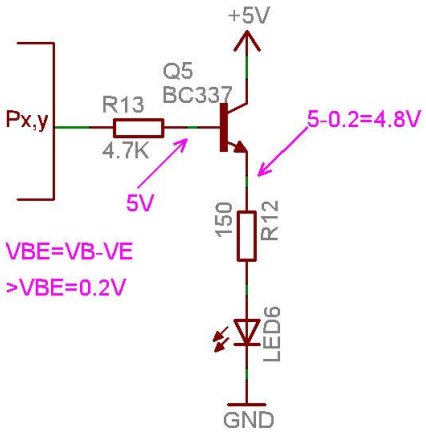
الشكل (2)



- متحكم بها لتكون فعالة عند المنطق المنخفض "**0**": وبالتالي فإن الترانزستور سوف يعمل كمفتاح لوصل/فصل نقطة التغذية (VCC) للحمل، وفي هذه الحالة سوف يستخدم ترانزستور من نوع PNP.

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

الشكل جانباً يبين تصميم خاطئ يستخدم ترانزستور من نوع NPN كمفتاح لوصل/فصل نقطة التغذية (VCC) للحمل. لنوضح الخطأ بالحسابات التالية:



حتى يفتح الترانزستور بشكل كامل (حالة الإشباع)، فيجب أن يكون التيار على قاعدته

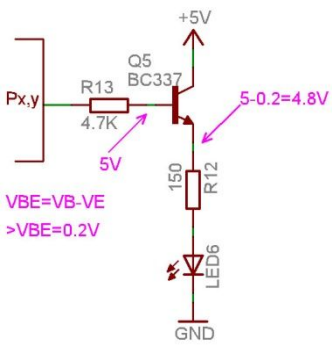
$V_{BE} = 0.7V$

إن الجهد الموجود على المشع (E) هو: $V_E = V_{CC} - V_{CE} = 5 - 0.2 = 4.8V$

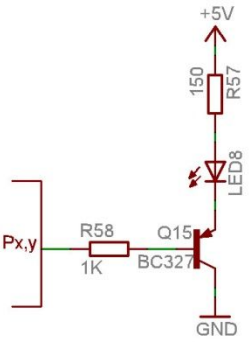
كما أن الجهد على قاعدة الترانزستور هو: $V_B = V_{PIN} = 5V$

وبالتالي فإن: $V_{BE} = V_B - V_E = 5 - 4.8 = 0.2V$

هذا يعني أن الترانزستور يعمل في المنطقة الفعالة ولن يكفي تيار مجمع الترانزستور (I_C) لتشغيل الحمل وسيعمل الثنائي الضوئي بشكل خافت.

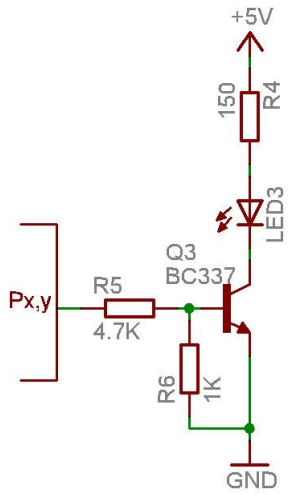


توصيل خاطئ!



إن الجهد المطبق على قاعدة الترانزستور في الدارة المبينة في الشكل (1) والشكل (2) يساوي 5V وهو نفسة جهد منطق بوابة المتحكم المصغر، بنفس الوقت من أجل الفتح الكامل للترانزستور فإنه يكفي تطبيق 0.7V، وإن هذا الجهد الزائد على القاعدة يؤدي إلى سحب تيار زائد، وبالتالي يمكن إضافة مقاومة مع مقاومة القاعدة ليتشكل لدينا مقسم كمون خرجه يتراوح بين 0.7~1V.

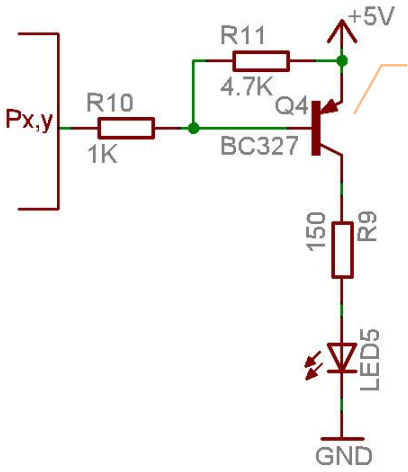
الشكل (3)



$$VB = VP \frac{R6}{R6 + R5}$$

$$VB = 5 \frac{1}{4.7 + 1} = 0.87V$$

الشكل (4)



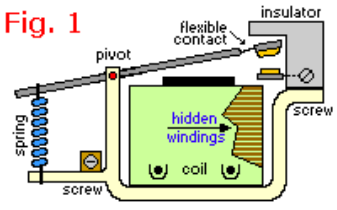
إن الدارات المدرجة أعلاه تقتصر على التحكم بعمل ثنائي ضوئي، بفس الوقت فإن الترانزستور يستخدم للتحكم بعمل ريليه الوصل الميكانيكي (Relay).

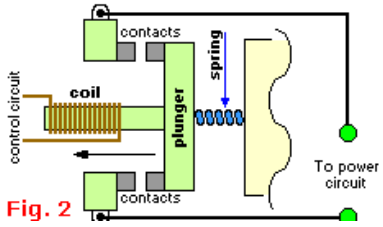
إن مجال جهود التحكم بالريليه واسع نسبياً، يتراوح:

3V, 5V, 6V, 9V, 12V, 15V, 24V, 36V, 48V, 60V

إن التيار الذي يستجره ملف تشغيل الريليه يتراوح من 30mA ~ 300mA، وذلك حسب حجم واستطاعة الريليه.

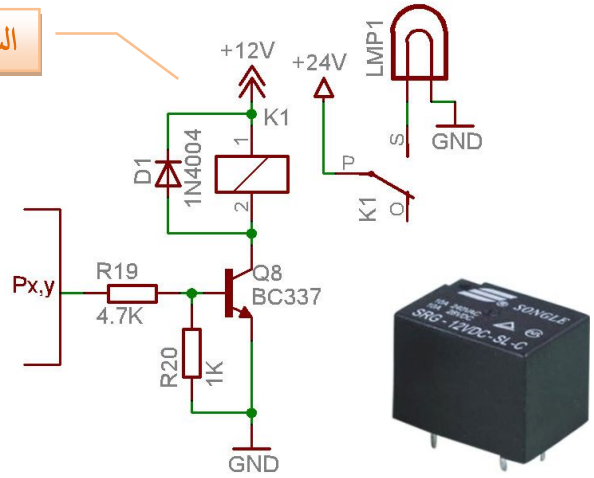
يوضح الشكل التالي رسماً تفصيلياً للبنية الداخلية للريليه حيث أنه عندما يتم تغذية ملف الريليه فإن الزراع الذي يحمل التماس المتحرك سوف ينجذب ويلامس التماس الثابت مؤدياً إلى وصل الدارة، وعندما يفقد الملف تهييجه تؤثر قوة النابض العكسية على الذراع وتعيده إلى وضعيته الأساسية.



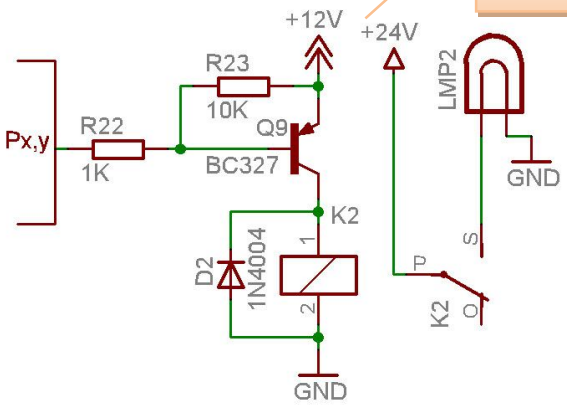


الدارات التالية يتم فيها استبدال الثنائيات الضوئية بـ Relay.

الشكل (5)



الشكل (6)



على الشكل (5) والشكل (6) تتحكم الريليه التي يعمل ملفها على 12V بعمل مصباح تيار مستمر جهد تشغيله 24V.

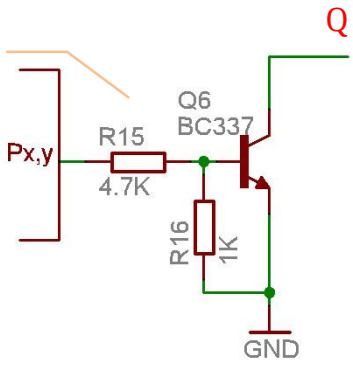
في الشكل (6) وعند تطبيق "1" على قطب بوابة المتحكم فإن الترانزستور Q8 سوف يغلق مؤدياً إلى وصل النقطة الأرضية إلى الطرف الثاني من ملف الريليه ويتهيج الملف مؤدياً بدوره إلى جذب تماس الريليه K1 وإغلاق النقطتين P, S، وعندها يضيء المصباح الكهربائي.

ملاحظة: من أجل حماية الترانزستور من أن يتم تدميره (حرقه) بسبب تيار التفريغ (Electromotive Force) لملف الريليه عند وصل وفصل الترانزستور، يتم إضافة ديود (ثنائي) على التوازي مع ملف الريليه (على الشكل أعلاه D1, D2) والذي بدوره يشكل حلقة مغلقة عند لتفريغ تيار الملف عند فصل الترانزستور.

مفاتيح التحكم الترانزستورية ذات المجمع المفتوح:

في بعض التطبيقات الخاصة يطلب أن يكون خرج الترانزستور المتحكم به ذو حالة منطقية وحيدة "1" or "0"، وهذا ما يسمى بالترانزستور ذو المجمع المفتوح.

الشكل (7)

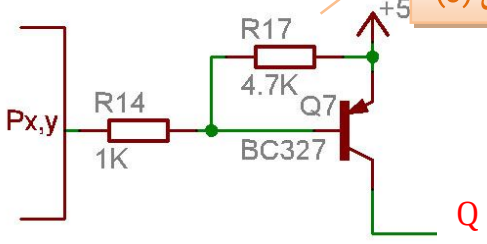


الشكل (7) يستخدم ترانزستور من النوع NPN.

في الحالة الافتراضية (الترانزستور مفتوح) يكون الخرج Q عائماً، وبالتالي يمكن استخدام مقاومة رفع (Pull-Up) إلى الجهد المطلوب وجوده عند هذه الحالة (5V, 12V, ..).

أمّا عند تطبيق "1" على بوابة الترانزستور، تصبح الحالة المنطقية على الخرج "0".

الشكل (8)



الشكل (8) يستخدم ترانزستور من النوع PNP.

في الحالة الافتراضية (الترانزستور مفتوح) يكون الخرج Q عائماً.

أمّا عند تطبيق "0" على بوابة الترانزستور، تصبح الحالة المنطقية على الخرج "1".

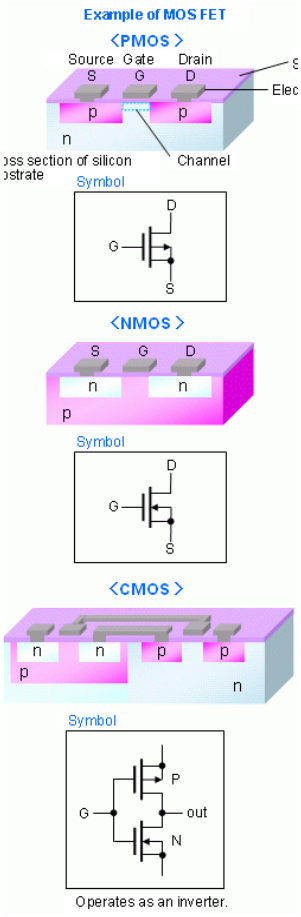
ثانياً: مفاتيح التحكم الترانزستورية باستخدام الترانزستورات أحادية القطبية (MOSFET & FET):

كما ذكرنا سابقاً، من أجل التحكم بالتطبيقات ذات الاستطاعات العالية، تستخدم الترانزستورات أحادية القطبية، التي تعتبر أغلى ثمناً من الترانزستورات ثنائية القطبية.

هنا ينبغي أن نوضح فكرة عامة وهي:

عندما نقول ترانزستور قناة نوع "**N**" فهذا يعني أن الترانزستور سيفتح القناة عند تطبيق جهد موجب على بوابته (G) لكي يفتح.

عندما نقول ترانزستور قناة نوع "**P**" فهذا يعني أن الترانزستور سيفتح القناة عند تطبيق جهد صفري أو سالب على بوابته (G) لكي يفتح.



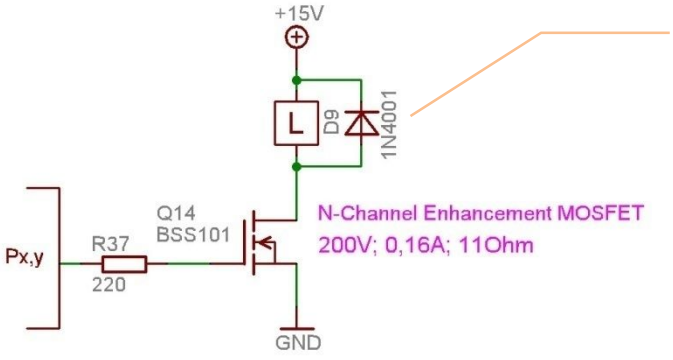
الشكل التالي عبارة عن دارة تحكم بحمل تحريضي (L) عن طريق ترانزستور MOSFET معزز بقناة نوع "N" له المواصفات التالية:

$$V_{DS\_max}=200V,\ I_{DS\_max}=160mA,\ R_{DS}=0.16\Omega,\ V_G=5V$$

يتحكم هذا الترانزستور بحمل تحريضي له المواصفات التالية:

$$V_{Load}=15V,\ I_{Load}=100mA$$

بما أن جهد بوابة الترانزستور $V_G=5V$ فيمكن قيادة هذا الترانزستور من بوابة المتحكم المصغر مباشرة كما في الشكل التالي:



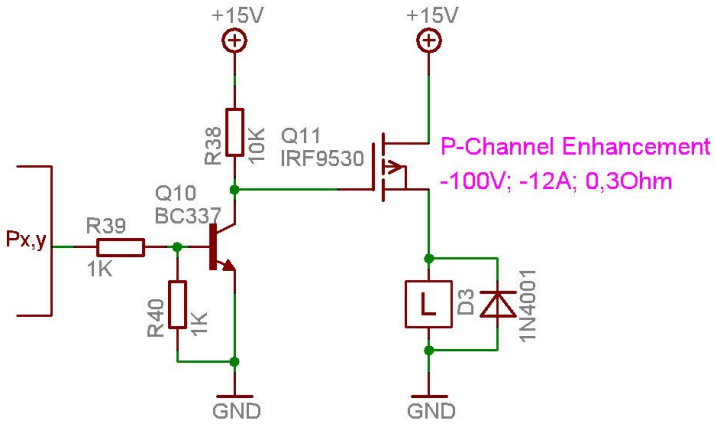
من أجل حماية الترانزستور من تيار التفريغ العكسي (EMF) للحمل التحريضي.

إن جهود التحكم ببوابة الترانزستورات أحادية القطبية تتراوح من 5V ~ 25V حسب استطاعة الترانزستور، وبالتالي لا يمكن قيادتها مباشرة من بوابة المتحكم المصغر، لذلك يستخدم الترانزستور ثنائي القطبية من أجل قيادة بوابة ترانزستور أحادي القطبية، حيث يعمل الترانزستور الثنائي القطبية كمفتاح إلكتروني لجهد تغذية بوابة الترانزستور ثنائي القطبية، ويقود الترانزستور أحادي القطبية الحمل مباشرة.

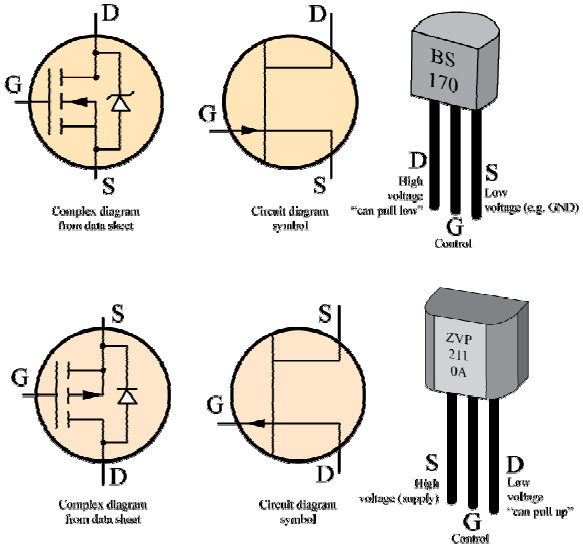
الشكل التالي عبارة عن دارة تحكم بحمل تحريضي (L) عن طريق ترانزستور MOSFET معزز بقناة نوع "P" له المواصفات التالية:

$$V_{DS\_max}=100V,\ I_{DS\_max}=12A,\ R_{DS}=0.3\Omega,\ V_G=15V$$

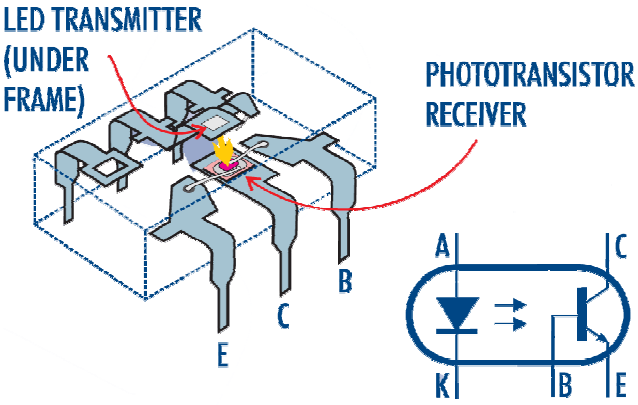
يتحكم هذا الترانزستور بحمل تحريضي له المواصفات التالية: $V_{Load} = 15V, I_{Load} = 10$.



يتم قدح الترانزستور الحقلي عن طريق الترانزستور الثنائي BC337، حيث أنه عند تطبيق "0" على قاعدة الترانزستور Q10 يفتح الترانزستور ويطبق "0" على بوابة الترانزستور الحقلي Q11 فيفتح الترانزستور ويمرر التيار إلى الحمل. أما عند تطبيق "1" على قاعدة الترانزستور Q10 يغلق مسبباً تطبيق جهد موجب (15V) على قاعدة الترانزستور Q11 وإغلاقه.



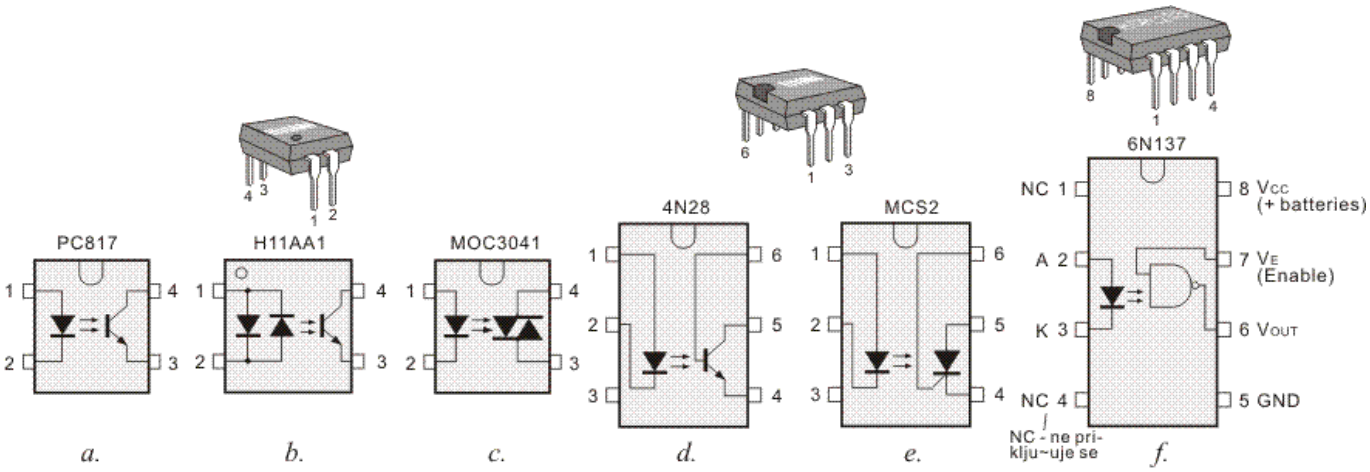
في كثير من التطبيقات يلزم عزل دارة التحكم التي يمكن أن تكون معالج أو متحكم مصغر والتي تعمل ضمن مستويات جهود منخفضة (TTL) عن دارة الحمل المتحكم به (مثل المحرك) والذي يعمل بجهود وتيارات عالية وذلك خصوصاً في التطبيقات التي يمكن أن تولد ضجيج (تحريضي، كهرطيسي، مغناطيسي)، وبالتالي يجب الفصل بين النقطة المشركة ومسارات التحكم لكلا الدارتين (التحكم والقيادة) بهدف منع انتشار الضجيج والتشويش التحريضي على مسارات دارة التحكم. والحماية من اختلال عملها وعمل المتحكم المصغر، وفي بعض الأحيان يمكن أن يؤدي الضجيج إلى محو الذواكر الموجودة.



من أجل حل هذه المشكلة تستخدم العوازل الضوئية وتدعى بـ (OPTO-ISOLATORS or PHOTO-COUPLERS)، وهي عبارة عن مرسل ضوئي ومستقبل ضوئي في دارة متكاملة واحدة، حيث أن المرسل يتصل مباشرة مع دارة التحكم والمستقبل الضوئي يتصل مع دارة القيادة، بمعنى آخر أن العازل الضوئي يقوم بالفصل الفيزيائي الكامل بين دارتين بحيث يبقى الارتباط بينهما ضوئياً بهدف نقل إشارة أو توليد أمر تحكم.

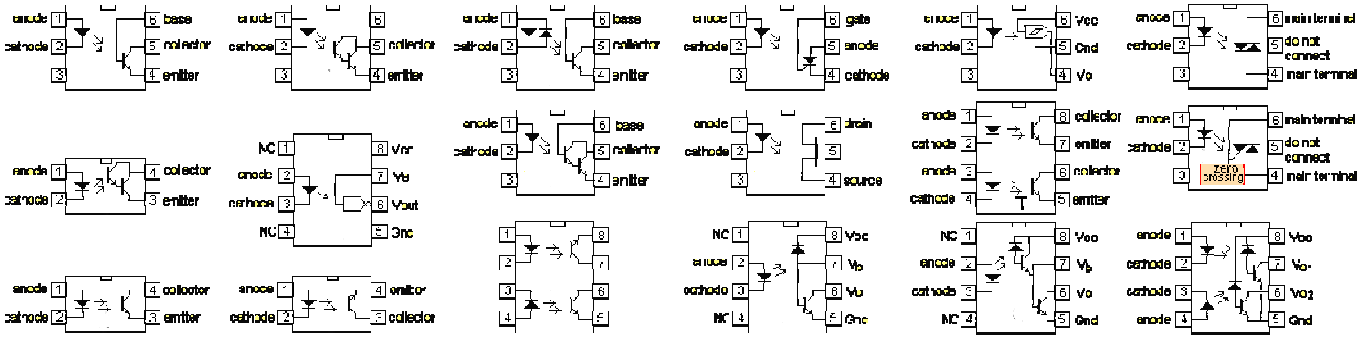
طبعاً ربما يتساءل البعض لماذا لا نستخدم الريليه بدلاً من العوازل الضوئية وهي تؤدي نفس الغرض!؟ هذا صحيح ولكن إذا قارنا الحجم الذي تأخذه الريليه الميكانيكية فسنجده كبير جداً مقارنةً مع حجم العوازل الضوئية بالإضافة إلى السرعة الكبيرة في نقل الإشارات باستخدام العوازل الضوئية، بنفس الوقت الذي تشكل فيه العطالة الميكانيكية عبئاً كبيراً على سرعة عمل الريليه، بالإضافة إلى ذلك كله تتميز العوازل الضوئية بوثوقية عمل عالية جداً.

تتوفر العوازل الضوئية على شكل دارات متكاملة ذات أغلفة مكونة من **4Pin, 6Pin, 8Pin**.



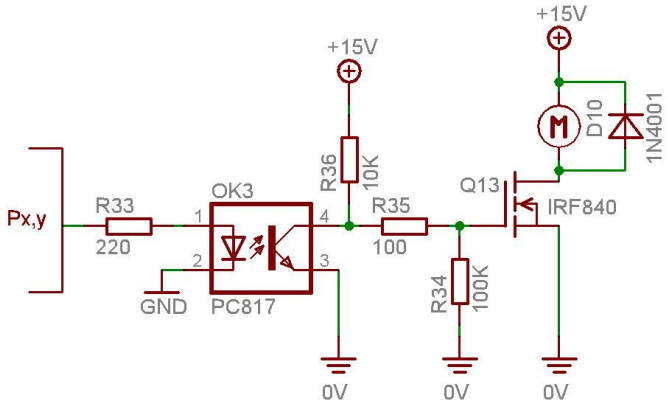
على الشكل أعلاه، تتوفر هذه العوازل بحيث يكون القيم المرسل هو عبارة عن ثنائي ضوئي يعمل بجهد 2V وتيار 15mA، أما بالنسبة لدارة الاستقبال فهي على عدة أنواع بما يتناسب مع الحمل المقاد.

تتوفر العوازل الضوئية بحيث يكون المستقبل الضوئي إما: ترانزستور ثنائي – ترانزستور حقلي – ترياك – ثايرستور – ترانزستور دارلنكتون – بوابة منطقية.



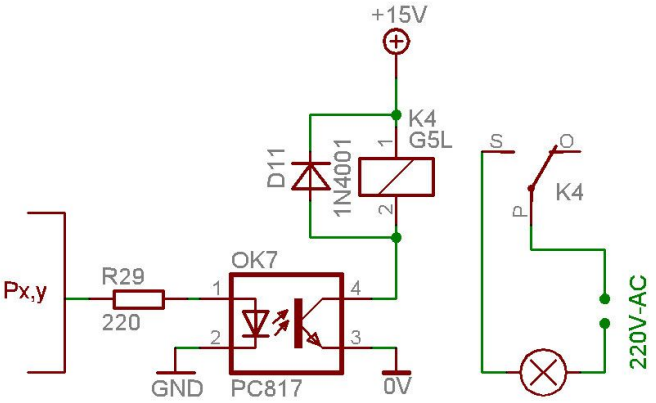
الشكل التالي هو عبارة عن دارة التحكم بسرعة محرك تيار مستمر عن طريق متحكم مصغر مقادة بشكل غير مباشر عن طريق عازل ضوئي ترانزستوري وترانزستور حقلي ذو بوابة معزولة قناة نوع P.

عند تطبيق "1" على دارة دخل العازل (الثنائي الضوئي) يتم قدح الترانزستور الضوئي داخل العازل فيفتح الترانزستور ويطبق "0" على بوابة الترانزستور الحقلي Q13 الذي هو قناة نوع P فيفتح الترانزستور ويمرر التيار إلى المحرك فيعمل. أما عند تطبيق "0" على قاعدة دخل العازل يغلق الترانزستور الضوئي داخل العازل مسبباً تطبيق جهد موجب (15V) على قاعدة الترانزستور Q13 وإغلاقه وبالتالي توقف المحرك عن العمل.

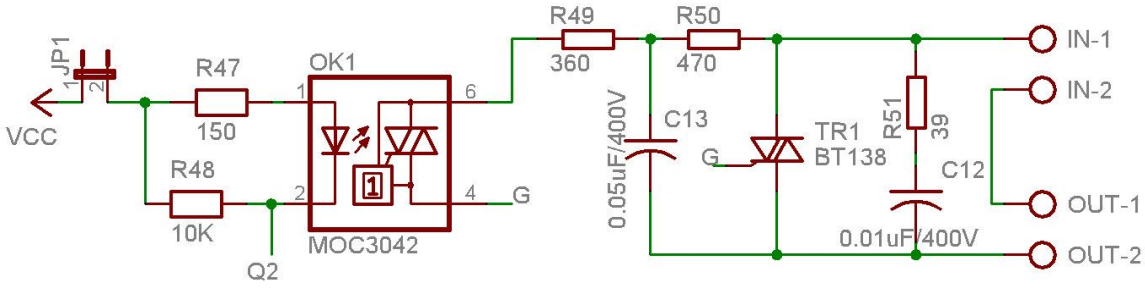


الشكل التالي هو عبارة عن دارة تحكم بمصباح تيار متناوب عن طريق متحكم مصغر مقاد بشكل غير مباشر عن طريق عازل ضوئي ترانزستوري وريليه ميكانيكية.

عند تطبيق "1" على دارة دخل العازل (الثنائي الضوئي) يتم قدح الترانزستور الضوئي داخل العازل فيفتح الترانزستور ويطبق "0" على الطرف الثاني لملف الريليه فيتهيج ملف الريليه ويؤدي إلى إغلاق تماسها فيمرر التيار إلى المصباح. أما عند تطبيق "0" على قاعدة دخل العازل يغلق الترانزستور الضوئي داخل العازل مسبباً قطع التغذية عن ملف الريليه فيفصل تماسها ويفتح دارة تغذية المصباح الكهربائي.



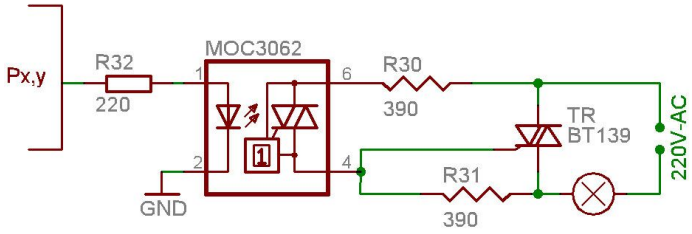
الشكل التالي هو عبارة عن دارة قيادة أحمال تحريضية ذات جهود وتيارات عالية عن طريق متحكم مصغر مقادة بشكل غير مباشر عن طريق عازل ضوئي من نوع ترياك وترياك استطاعي.



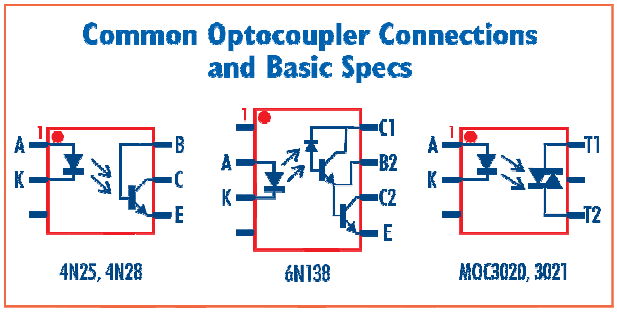
Control Signal ("0"= on | "1"= off)

إن هذه الدارة تختلف في بنيتها عن دارات العوازل السابقة حيث أن قسم الاستقبال في العازل الضوئي سوف يتحكم بشكل مباشر بمصدر التيار الكهربائي.

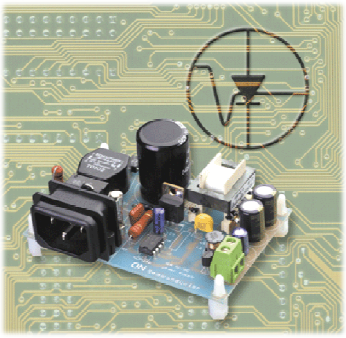
عند تطبيق "0" على دارة دخل العازل (الثنائي الضوئي) يتم قدح الترياك الضوئي داخل العازل فيفتح الترياك ويؤدي إلى مرور التيار الكهربائي من نقطة الدخل IN_1 (220V/50HZ) عبر دارة مقسم الجهد إلى قاعدة الترياك BT138، فيفتح الترياك ويؤدي إلى مرور التيار إلى خرج الحمل على النقطتين $OUT_{1,2}$. أما عند تطبيق "1" على دخل العازل يغلق الترياك الضوئي داخل العازل ويفصل الترياك الاستطاعي. إن الغاية من وجود دارة المرشح المؤلف من المقاومة R51 والمكثف C12 هو للحد من أثر تيار التسريب للحمل التحريضي. من أجل حمولات غير تحريضية يمكن استخدام الدارة التالية.



الجدول التالي يبين أكثر أنواع العوازل الضوئية استخداماً.



TYPE	ISOLATION (Viso)	INPUT LED IF(max)	OUTPUT VCE(max)	CTRmin (@ IF)	BANDWIDTH (kHz)
4N25	5300Vrms	80mA	7V	20% (10mA)	300
4N28	5300Vrms	80mA	7V	10% (10mA)	300
6N138	2500Vrms	20mA	7V	300% (1.6mA)	~20
MOC3020	7500Vpk	50mA	Voff =400V	(Trig. @ 30mA)	—
MOC3021	7500Vpk	50mA	Voff =400V	(Trig. @ 15mA)	—



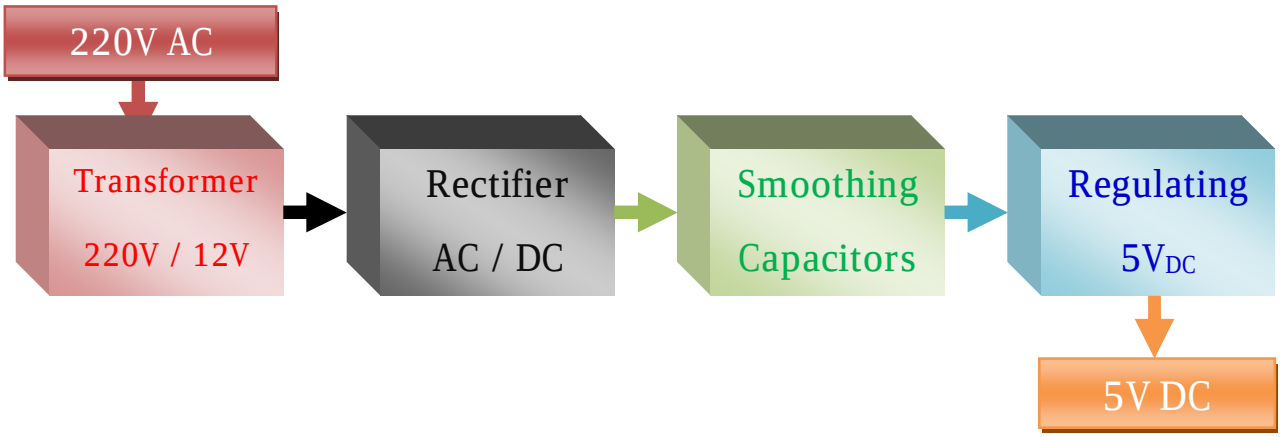
تؤثر جودة تصميم وحدة التغذية المستمرة بشكل كبير في أداء النظام ووثوقية عمله.

بشكل عام تنقسم وحدات التغذية المستمرة إلى نوعين أساسيين:

- وحدات التغذية المستمرة الخطية: وهي التي تعتمد على المحولات لتخفيض الجهد المتناوب ومن ثم تقويمه وتنظيمه.
- وحدات التغذية المستمرة التقطيعية: ولا تحوي على محولات تخفيض التيار وإنما يتم الحصول على التيار المطلوب عبر تقطيع تيار الدخل باستخدام دارات متكاملة تقطيعية. فعالية هذه الدارات كبير جداً يصل إلى 95%. كما أنا قابل للعمل على مجال واسع من جهد وتردد الدخل (90V~265V / 45HZ~55HZ).

الذي سوف ندرسه في هذا المبحث هو تصميم وحدة تغذية خطية ذات أداء وكفاءة عالية.

تتألف مراحل تصميم وحدة التغذية المستمرة الخطية من أربع مراحل أساسية موضحة على الشكل التالي:



المرحلة الأولى: تحويل الجهد المتناوب من 220V إلى جهد متناوب منخفض متناسب مع الجهد المستمر المطلوب وذلك باستخدام محولات تيار متناوب ذات تيار (استطاعة) متناسبة مع استطاعة الحمل.

تتكون هذه المحولات من ملف ابتدائي يوصل إلى الجهد العالي (220V)، وملف ثانوي يعطي الجهد المخفّض.

تتوفر هذه المحولات في الأسواق بجهود خرج مختلفة:

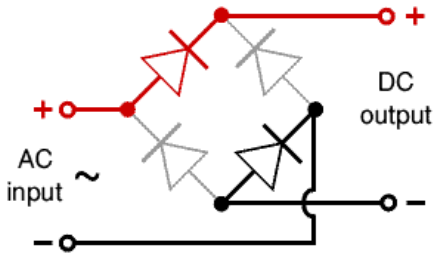
(5V, 6V, 9V, 12V, 15V, 28V, 24V, 36V, 48V).

واستطاعات عديدة:

(200mA, 300mA, 400mA, 500mA, 1000mA, 1500mA).

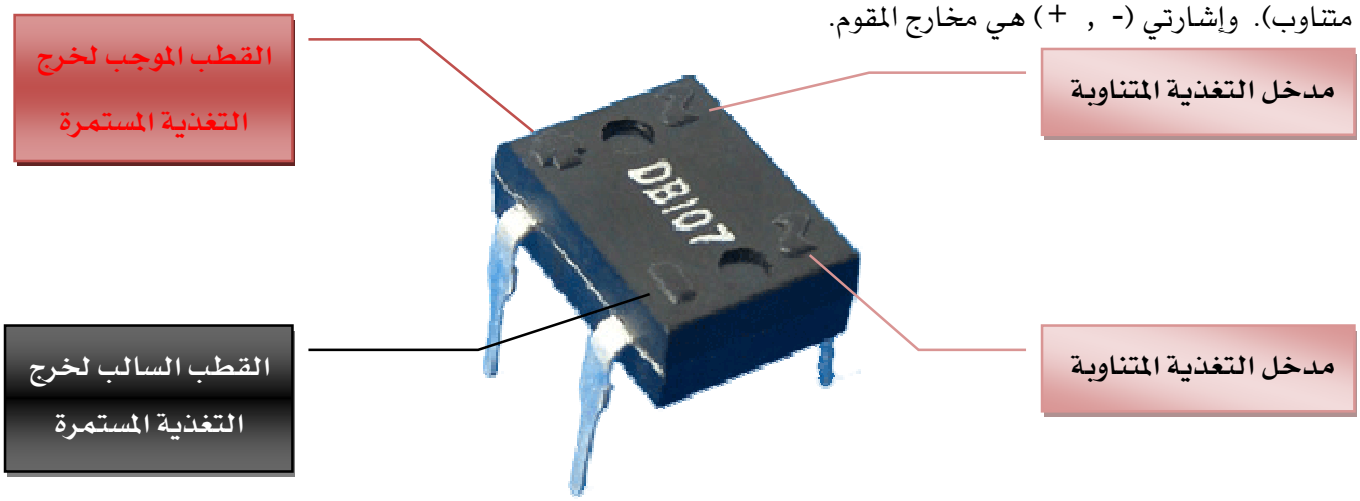
وتتوفر أيضاً بجهود مزدوجة (6x2, 8x2, 9x2, 12x2, 15x2, etc...) وتسمى المحولات ذات النقطة المشتركة.

المرحلة الثانية: تقويم الجهد المتناوب إلى جهد مستمر، ويتم ذلك باستخدام المقومات الجسرية.

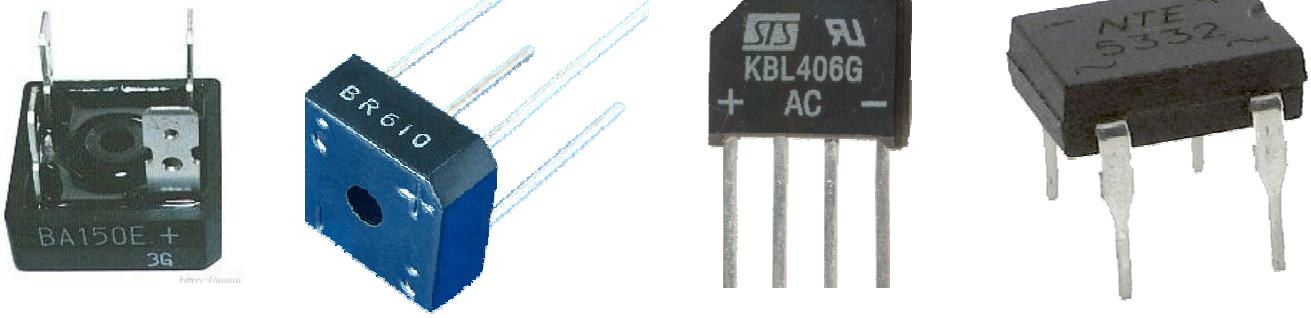


يتكون المقوم الجسري (جسر التقويم) من أربعة ثنائيات موصلة مع بعضها بشكل جسري، ويحتوي المقوم الجسري على أربعة أرجل يكون مشاراً عليها بالرموز التالية (~ , - , +).

حيث أنّ إشارة (~) الموجودة على القطبين هي مدخل المقوم الجسري (تيار متناوب). وإشارتي (- , +) هي مخارج المقوم.



تتوفر هذه المقومات الجسرية باستطاعات مختلفة (1A, 2A, 4A, 6A, 10A, 14A, 25A, 40A) لتلبي كافة التطبيقات البسيطة منها والصناعية.

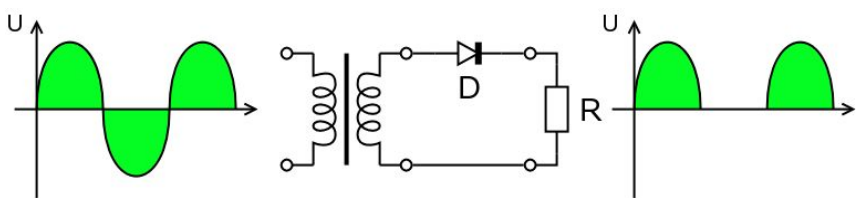


14A / 400V | 6 Amp | 4 Amp | 1 Amp

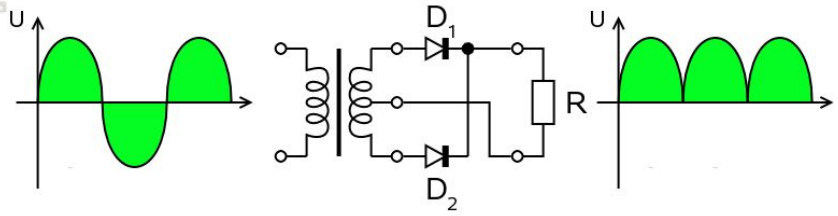
إن طرق توصيل دارات التقويم لا تقتصر فقط على طريقة تقويم الموجة الكاملة التي تستخدم المقوم الجسري، وإنما على هناك دارات تقويم مثل: نصف الموجة، الموجة الكاملة ذات النقطة المشتركة.

إن الاختلاف الرئيسي بين الوصلات الثلاث هو في القيمة الوسطية لجهد وتيار الخرج المقوم.

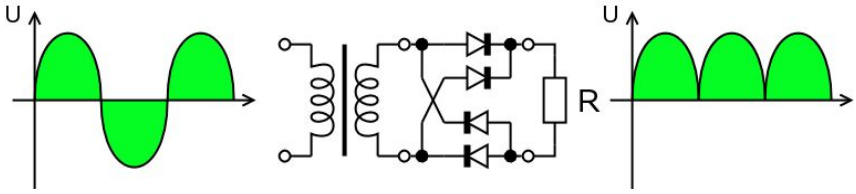
الأشكال التالية تبين الاختلاف في التوصيل والبنية للوصلات الثلاث.



دارة تقويم نصف موجة



دارة تقويم موجة كاملة ذات نقطة مشتركة

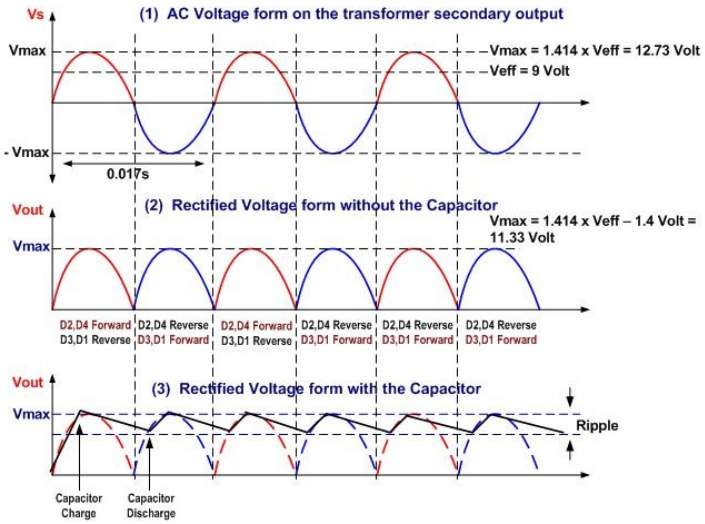


دارة تقويم موجة كاملة جسرية

تعتبر دارة تقويم الموجة الكاملة الجسرية من أكثر الدارات انتشاراً واستخداماً وكفاءةً، وهي ماسوف نعتمده في تصميم وحدة التغذية في هذا الفصل.

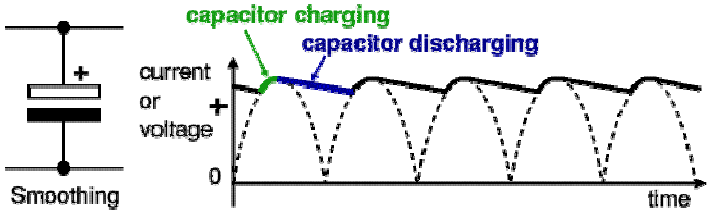
الجدول التالي يوضح العلاقات الحسابية للجهود والتيارات من أجل الوصلات الثلاث.

دارة تقويم نصف موجة	دارة تقويم موجة كاملة نقطة مشتركة	دارة تقويم موجة كاملة جسرية
$Vdc = \frac{Vm}{\pi}$	$Vdc = \frac{2Vm}{\pi}$	$Vdc = \frac{2Vm}{\pi}$
$V_{rms} = \frac{Vm}{2}$	$V_{rms} = \frac{Vm}{\sqrt{2}}$	$V_{rms} = \frac{Vm}{\sqrt{2}}$
$Vdc = \frac{\sqrt{2}V_{rms}}{\pi}$	$Vdc = \frac{2\sqrt{2}V_{rms}}{\pi}$	$Vdc = \frac{2\sqrt{2}V_{rms}}{\pi}$
$V_{Diod} = Vm$	$V_{Diod} = 2Vm$	$V_{Diod} = Vm$
$I_{Diod} = IL$	$I_{Diod} = 0.5IL$	$I_{Diod} = 0.5IL$



المرحلة الثالثة: ترشيح الجهد المقوم باستخدام مكثفات كيميائية ذات قيم محسوبة.

إن الغاية الرئيسية من الترشيح هو جعل قمم التيار المقوم ناعمة بحيث تكون القيمة العظمى هي نفسها القيمة الوسطية للتيار وذلك الحصول على إشارة قريبة من الإشارة الأمثلية للتيار المستمر.



يتم حساب قيمة المكثف اللازم للترشيح انطلاقاً من العلاقة التالية:

$$C = \frac{5 \times I_L}{V_S \times f}$$

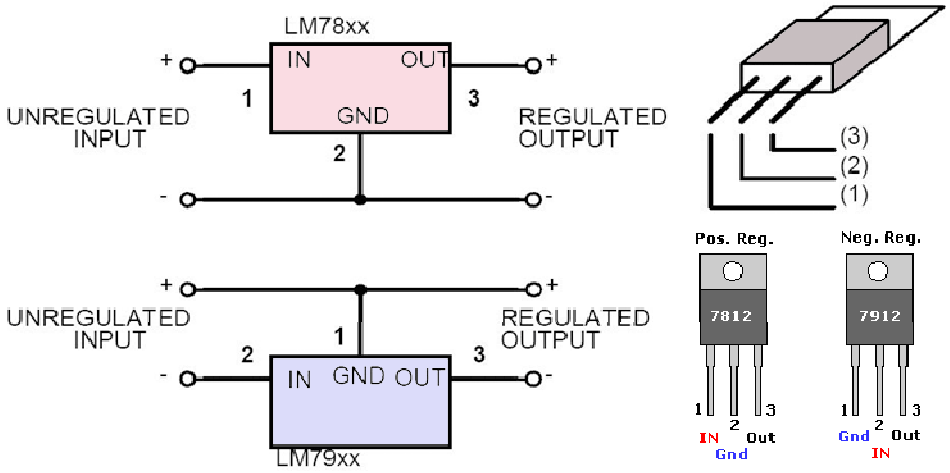
حيث أن: IL هو تيار الحمل، Vs جهد التغذية المقوم (peak value of unsmoothed DC)، f هو تردد الشبكة المتناوبة.

المرحلة الرابعة: تنظيم الجهد المقوم والمرشح باستخدام منظم جهد فعال.

تستخدم منظمات الجهد في الدارات الإلكترونية من أجل تنظيم جهود خرج دارات التقويم من أجل الحصول على جهود خرج ثابتة ودقيقة.

إن أشهر أنواع هذه المنظمات هي منظمات العائلة (78XX, 79XX)، حيث تستخدم العائلة (78XX) من أجل تنظيم الجهود الموجبة، وتستخدم العائلة (79XX) من أجل تنظيم الجهود السالبة.

Fixed Voltage Regulators (7800, 7900 series)



تتوفر منظمات الجهد بحيث تغطي جميع الجهود القياسية (5V, 6V, 8V, 9V, 10V, 12V, 15V, 18V, 24V) المستخدمة في تطبيقات دارات التيار المستمر. كما أن الجهود المطبقة على دخل منظم الجهد يجب أن تكون أكبر من الجهد المراد تنظيمه بحد قليل محدد في الجدول التالي:

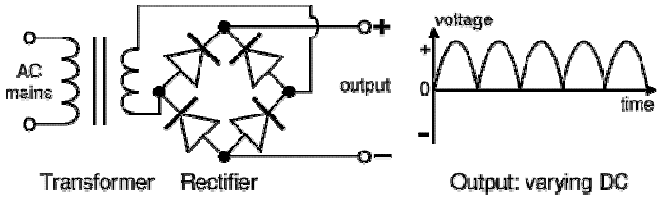
IC Part Number	Output Voltage	Minimum V_I Required
7805	+5V	7.3V
7806	+6V	8.35V
7808	+8V	10.5V
7810	+10V	12.5V
7812	+12V	14.6V
7815	+15V	17.7V
7818	+18V	21V
7824	+24V	27.1V
7905	-5V	-7.3V
7906	-6V	-8.35V
7908	-8V	-10.5V
7910	-10V	-12.5V
7912	-12V	-14.6V
7915	-15V	-17.7V
7918	-18V	-21V
7924	-24V	-27.1V

كما نرى في العمود الثالث فإن هناك جهد أصغري يجب أن يكون على دخل المنظم حتى نحصل في خرجه على الجهد المطلوب.

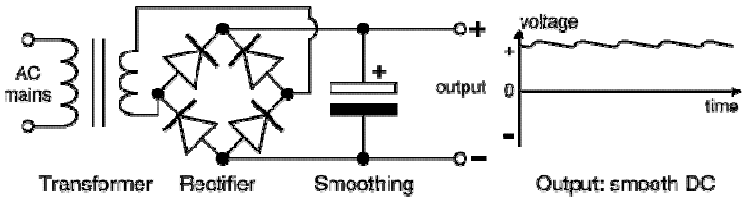
إن منظمات الجهد للعائلة 78XX or 79XX قادرة على تنظيم جهود خرج لأحمال لا تتجاوز تيارها 1A، أما من أجل منظمات ذات تيارات أعلى، فهناك منظمات أخرى مخصصة لهذه الأغراض، ولكن عموماً معظم الدارات الإلكترونية لا تتجاوز تياراتها 1A.

مراحل دارة تقويم موجة كاملة :

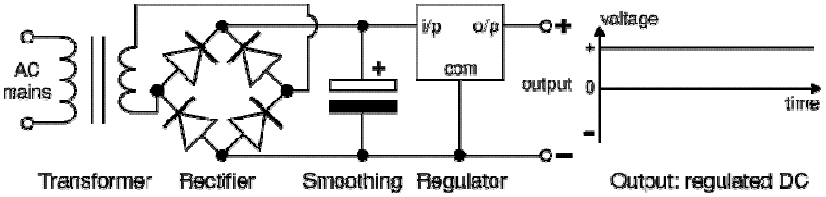
Transformer + Rectifer



Transformer + Rectifer + Smoothing



Transformer + Rectifer + Smoothing + Regulator



المطلوب تصميم وحدة تغذية مستمرة منظمة بجهد خرج 5V وتيار 800mA.

من أجل الحصول على جهد خرج 5V منظم فإننا نحتاج إلى جهد أصغري 7.3V على دخل المنظم، وبالتالي يمكن أن نختار محولة 220V/9V، لأن الجهد على خرج المقوم الجسري سيكون:

$$V_{Rectifier\_Bridge} = VPP_{Trans} - V_{Diod\_Drop}$$

$$V_{Rectifier\_Bridge} = 9 - 1.4 = 7.6V$$

بما أن تيار الحمولة 800mA، فإن استطاعة المحولة والمقوم الجسري ومنظم الجهد يجب أن يكون أكبر من تيار الحمل بمعامل احتياط 20% لكل لا تعمل العناصر عن قيمها الحدية الأعظمية.

$$I_{Device} = I_{Load} \times 20\%$$

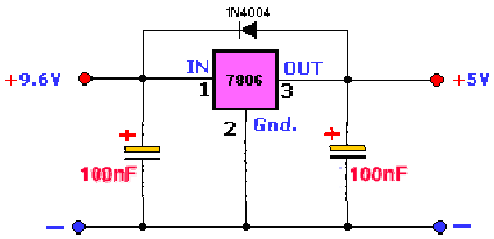
$$I_{Device} = 0.8A \times 20\% = 1A$$

وبالتالي نختار تيار المحولة والمقوم الجسري ليكون 1000mA.

من أجل الترشيح، فإننا دائماً نستخدم مكثف ترشيح قيمته بين **1000uF ~ 2200uF** ويجب أن لا تتجاوز قيمة المكثف عن هذه القيمة في دارات التغذية ذات الأغراض العامة لكي لا يتم استجرار تيار شحن كبير أثناء وصل التغذية (الحالة العابرة).

بالإضافة إلى مكثف الترشيح المذكور، يضاف على التوازي معه مكثف تنعيم قيمته عشر قيمة مكثف الترشيح **100uF ~ 220uF** من أجل تنعيم الإشارة المرشحة.

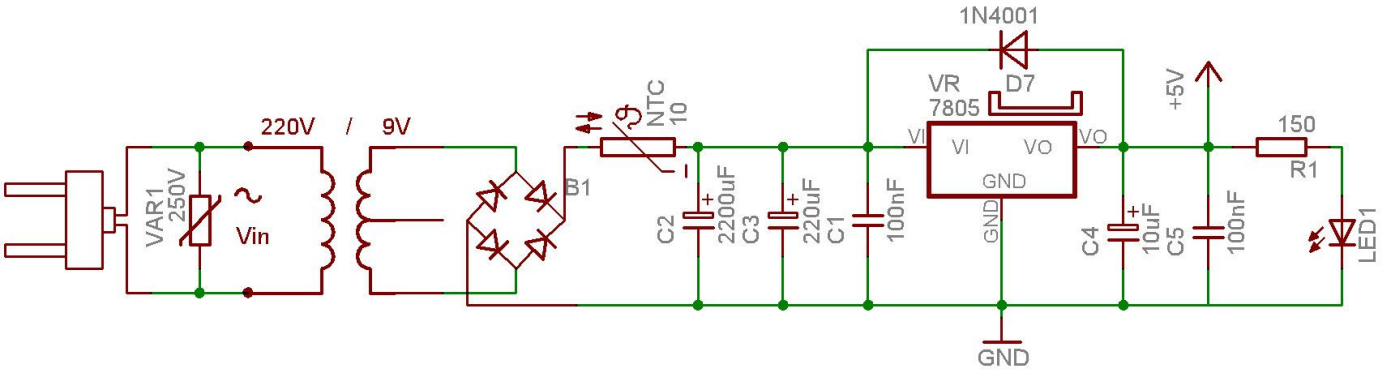
إن منظم الجهد حساس بشكل كبير من الجهود الستاتيكية، لذلك يتم وضع مكثف **100nF** يقوم بامتصاص الجهود الستاتيكية (شوكية نبضية) التي يمكن أن تحدث بشكل عابر نتيجة عدم استقرار الشبكة الكهربائية أو نتيجة لتراكب ضجيج إلكترو مغناطيسي على خطوط التغذية.



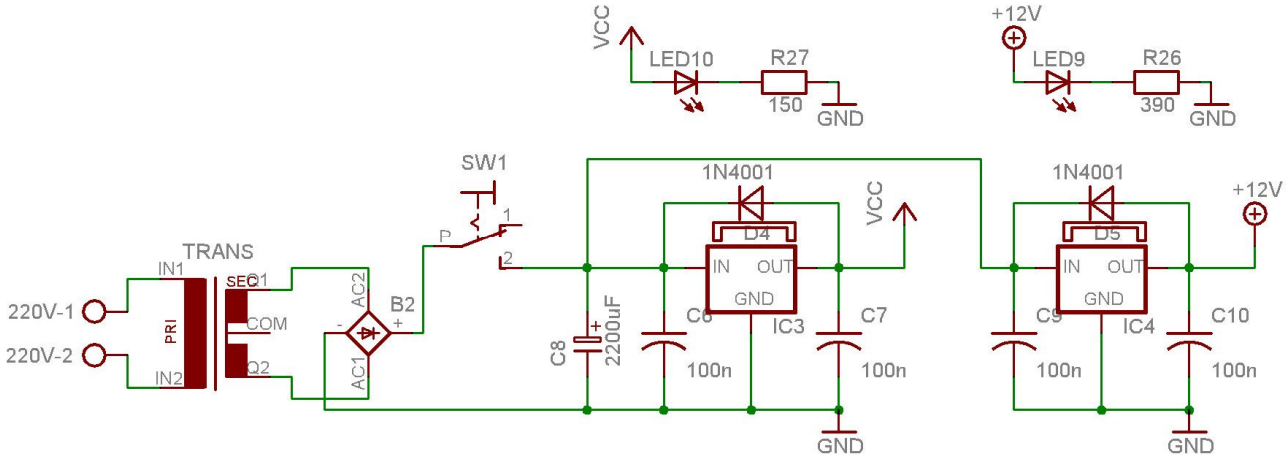
إن منظم الجهد عبارة عن عنصر فعال سيقوم بتبديد الجهد الزائد على دخله على شكل حرارة، وبالتالي سوف يتشكل لدينا ضجيج حراري متراكب على خرج المنظم، وبالتالي يتوجب وضع مكثف **100nF** على خرج المنظم كما في دخله.

في بعض الأحيان يمكن أن يحصل تغذية ارتجاعية من النظام (خرج المنظم) إلى التغذية، وذلك بسبب وجود سعات ومحرضات كبيرة في النظام سوف تقوم بالتفريغ بشكل عكسي عند فصل التغذية عن الدارة وسف تؤدي إلى حرق منظم الجهد، لذلك يجب وصل ثنائي عادي (Diode) على التوازي والتعاكس مع منظم الجهد، والذي بدوره يشكل مساراً لمرور أي تيار ارتجاعي عند فصل التغذية الكهربائية.

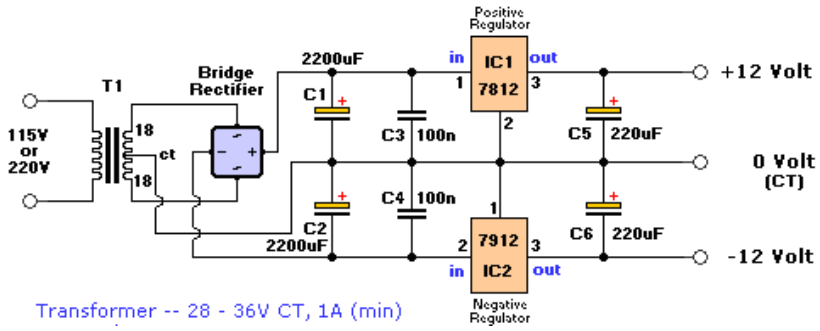
الشكل التالي يبين التصميم النهائي للدارة...



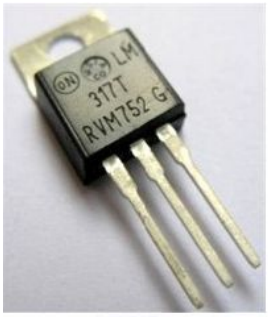
في بعض التطبيقات العملية ربما يحتاج النظام أكثر من جهد تغذية في النظام (5V, 12V مثلاً)، وبالتالي لا حاجة لتصميم دارتي تغذية، يكفي إعادة الحسابات الرياضية للجهود والتيارات. الشكل التالي دارة تغذية مزدوجة 5V, 12V.



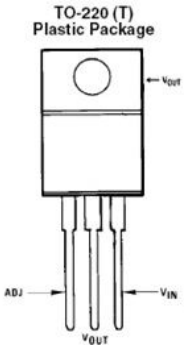
في بعض التطبيقات التي تحوي على دارات مضخمات عملياتية، فإنه يلزم وجود تغذية موجبة وسالبة في النظام، وبالتالي كل ما نحتاجه هو استخدام منظم جهد موجب ومنظم جهد سالب ليتم توصيلهما كما في الشكل التالي:

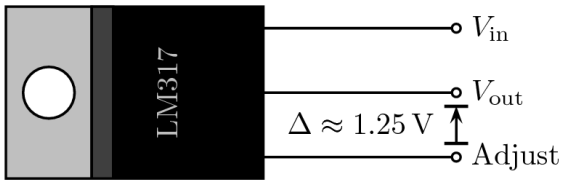


بالإضافة إلى منظمات الجهود الثابتة يوجد منظمات جهود متغيرة مثل المنظم LM317، بحيث يمكن تغيير جهد الخرج عن طريق مقاومة متغيرة موصولة مع المنظم في مجال 1.25V ~ 33V.



LM 317

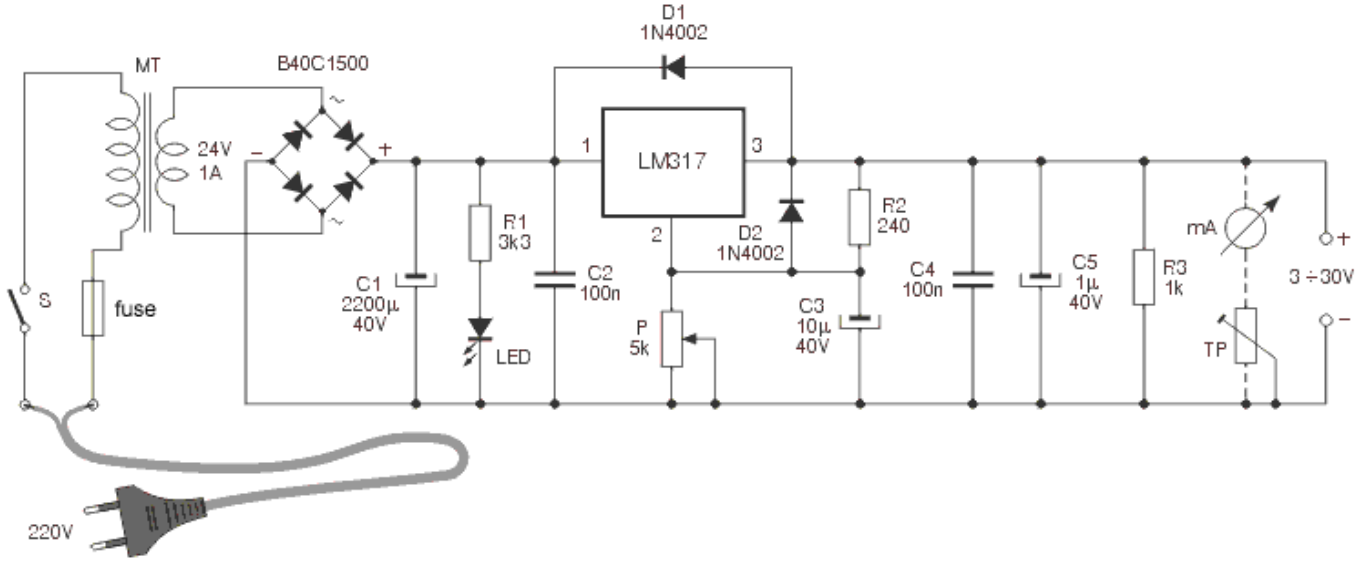




يعطى جهد خرج المنظم بالعلاقة التالية:

$$V_{out} = 1.25 \times \left[\left(1 + \frac{RP}{R2}\right) + \left(\frac{I_{adj}}{R5}\right)\right]$$

الشكل التالي يوضح الدارة العملية لهذا المنظم.



Written by: ***Walid Balid****, Embedded Systems Engineer,*
Aleppo, on Sunday, 22th March, 2009

برمجة المتحكمات المصغرة

التجارب العملية

الجلسة السابعة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

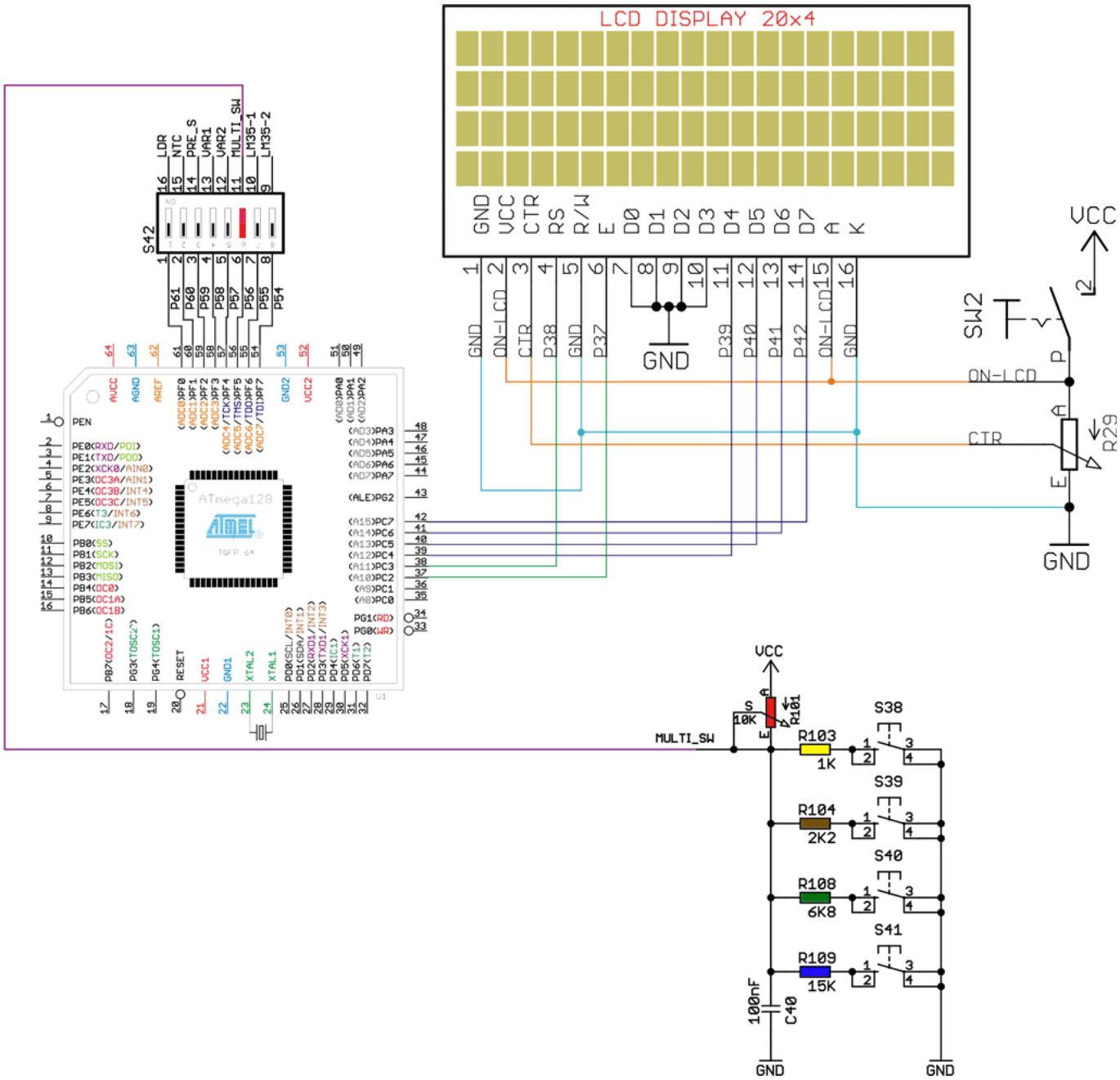
WALID BALID, Tuesday, December 15, 2009

التجربة الثامنة عشرة: ربط عدة مفاتيح إلى قطب وحيد
Exp.18: Interfacing n-Switch to a Single PIN

الغاية من التجربة:

توصيل وبرمجة مجموعة مفاتيح إلى قطب مبدل تشابهي رقمي وحيد.

مخطط التوصيل:



متطلبات التوصيل: يجب إغلاق النقطة 6 من المفتاح S42. وإغلاق المفتاح SW2 لتغذية شاشة الإظهار.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لقراءة حالة خمس مفاتيح موصولة إلى قطب مبدل تشابهي رقمي وحيد.

إن الفكرة الأساسية هي أن كل مفتاح عند ضغطه سوف يطبق جهد تشابهي مختلف وذلك لأن هناك مقاومة مقسم كمون مختلفة لكل مفتاح.

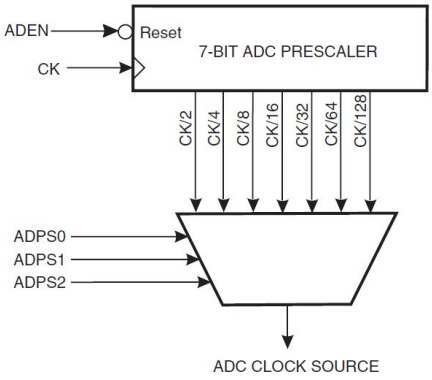
التعليمات الجديدة:

إن معظم التعليمات الجديدة لها علاقة بتشغيل وتعريف المبدل التشابهي الرقمي.

التعليمة البرمجية	شرح التعليمة

`Config Adc= Single/Free,`
`Prescaler = Auto,`
`Reference = off|Avcc|internal`
	إعداد المبدل التشابهي الرقمي.
	`Single` يبدأ التحويل بعد التعليمة Start ADC (ADSC) وكلما مر على التعليمة `Getadc`.
	`Free` يبدأ التحويل بعد التعليمة Start ADC ويستمر بشكل متعاقب في أخذ العينات والتحويل وتحديث القيم في مسجل المبدل (ADCH and ADCL).
	`Prescaler` **تحديد** المقسم الترددي للمبدل.
	`Reference` تحديد الجهد المرجعي للمبدل.

`Start Adc`	بدء عملية التحويل (تغذية المبدل).
`Stop Adc`	إيقاف عملية التحويل (فصل تغذية المبدل).
`var = Getadc(channel)`	قراءة القيمة التي تم تبديلها على قناة المبدل المحددة بـ `channel`



المقسم الترددي للمبدل (Prescaler): سوف يقوم بتقسيم تردد الهزاز الكريستالي للحصول على تردد عمل المبدل، ومن أجل الحصول على أعلى دقة للمبدل يجب أن يكون تردد عمل المبدل يتراوح 50 ~200 KHz، وأمّا من أجل سرعة تبديل أكبر، فإنه يمكن زيادة تردد التبديل إلى قيم أكبر من 200KHz ولكن ذلك سيكون على حساب الدقة، فكلما زاد تردد التبديل نقصت دقة المبدل.

الجهد المرجعي (Reference): يمكن اختيار الجهد المرجعي الداخلي للمبدل "Internal" وهو 2.56V بشرط أن لا يكون مجال إشارة القياس أكبر من الجهد المرجعي، وإلا فإنه يجب استخدام الجهد "AVCC" كجهد مرجعي. في بعض الأحيان يكون مجال إشارة القياس أصغر بكثير من الجهد المرجعي الداخلي، أو يتطلب عملية قياس دقيقة وبالتالي الحاجة إلى جهد مرجعي دقيق، فعندها يمكن استخدام القطب Vref كمدخل جهد مرجعي للمبدل "of".

ميزات المبدلات التشابهية الرقمية في عائلة المتحكمات AVR:

تتميز المبدلات التشابهية الرقمية لمتحكمات العائلة AVR_{mega} بالميزات التالية:

- طول المبدلات 10-bit.
- مجال القياس 0 - VCC
- خطأ القياس ±2 LSB.

- خطأ عدم الخطية 0.5 LSB.
- زمن أخذ العينة 260 Micro Sec – 13.

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	14.5	25
Normal conversions, single ended	1.5	13
Normal conversions, differential	1.5/2.5	13/14

- سرعة أخذ العينات 15KSPS عند أعلى دقة.
- تحوي على جهد مرجعي داخل اختياري 2.56V.
- تملك نمطي عمل (Single, Free).
- تملك مقاطعة اكتمال عملية التحويل.
- تملك نمط تخفيض ضجيج المبدل.
- تردد عمل المبدل يتراوح 50 ~1000 KHz
- تستخدم كمداخل مبدل إشارة ذات دخل وحيد عن طريق موزع.
- يمكن استخدامها كمداخل مبدل إشارة ذات دخل تفاضلي.
- يمكن استخدامها كمداخل مبدل إشارة ذات دخل تفاضلي مع عامل ربح x10 – x200.

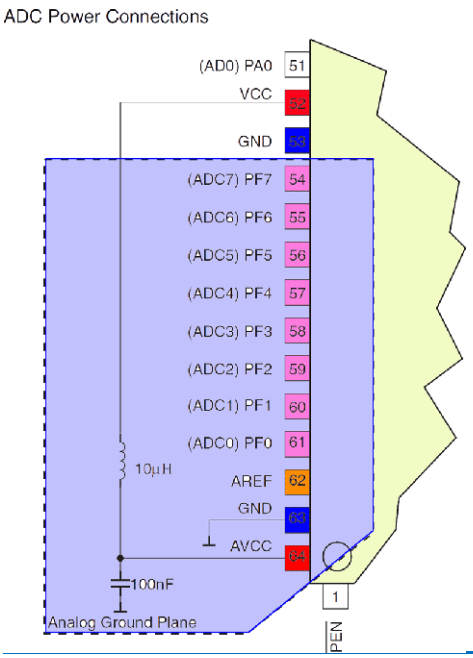
اعتبارات هامة لتخفيض ضجيج المبدلات التشابهية الرقمية واستقرار عملها في عائلة المتحكمات AVR:

يجب أن لا يتم وصل التغذية التشابهية للمبدل مع التغذية الرقمية للدارة لمنع انتقال الضجيج التشابهي على خطوط التغذية الرقمية، وإنما يتم الفصل بينهما عن طريق مرشح تمرير منخفض من نوع LC (L=100uH, C=100nF) أو RC (R=100Ω, C=100nF).

يجب أن لا يكون مجال التغير في الجهد بين التغذية الرقمية والتغذية التشابهية أكبر من **0.3V** ±.

يجب أن تكون المسارات التشابهية الموصولة مع مداخل المبدلات على مخطط الدارة المطبوعة أقصر ما يمكن لتفادي تراكب الضجيج على هذه المسارات وتفادي انتقال الضجيج على مسارات الإشارات الرقمية.

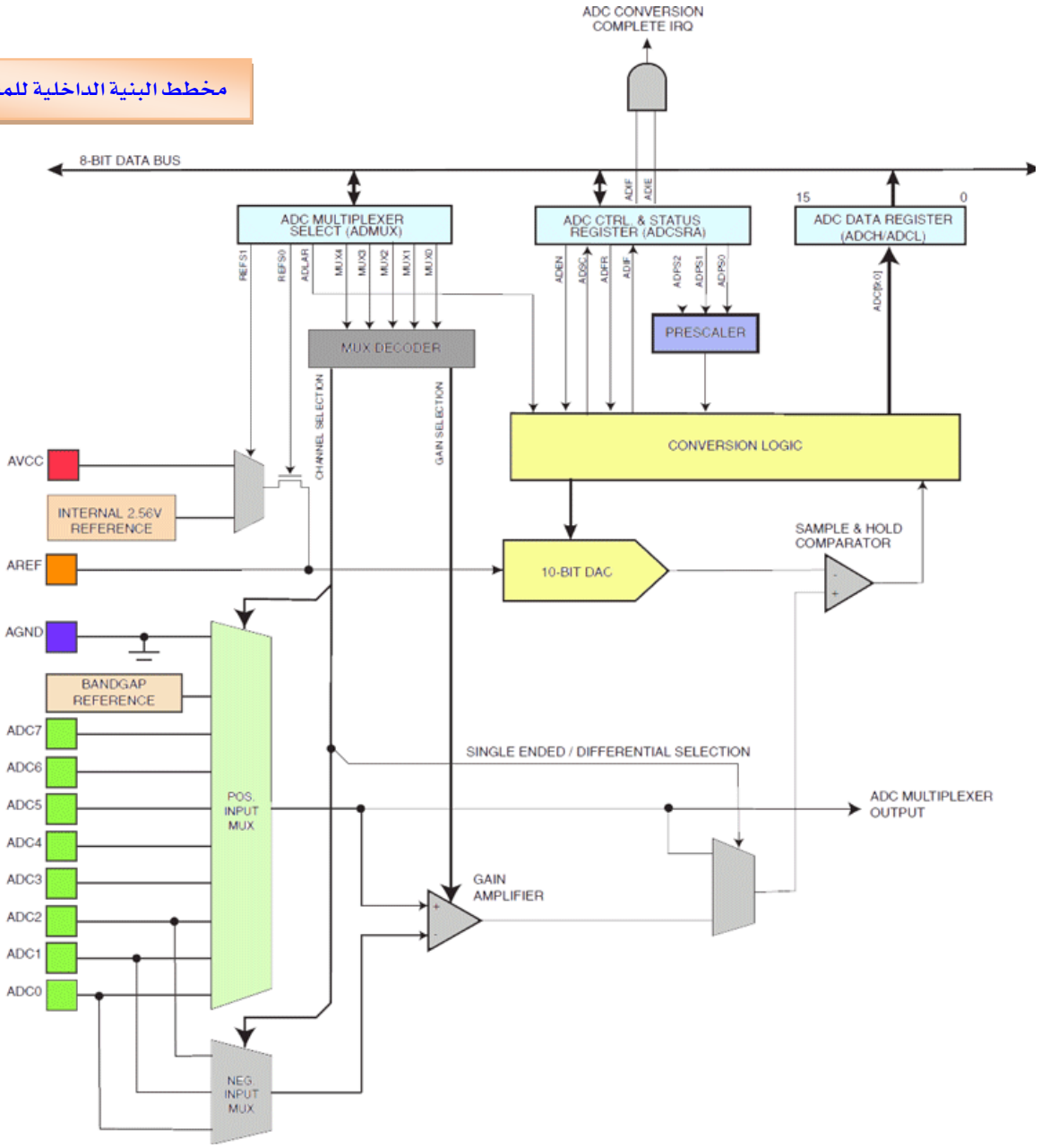
يجب أن تكون الإشارات التشابهية الداخلة إلى المبدل محاطة بمساحة مصمتة من الأرضي التشابهي على هو مبين على الشكل جانباً.



يجب إدخال المتحكم في نمط البطالة (Idle mode) أثناء عملية التبديل وتفعيل مقاطعة انتهاء عملية التبديل للخروج من نمط البطالة من أجل دقة قياس أكبر وعدم تأثر المبدل بضجيج عمل المعالج، وذلك كما يلي:

- يجب اختيار نمط العمل "Single".
- يجب تفعيل مقاطعة اكتمال عملية التحويل "Enable ADC".
- قم بتمكين المبدل "Start ADC" وتأكد من أنه لا يقوم بأي عملية تحويل.
- أدخل المتحكم في نمط البطالة "Idle".
- سوف يبدأ المبدل بالتحويل حالما يتوقف المعالج عند الدخول إلى نمط البطالة، ومع انتهاء عملية التحويل سوف تحدث مقاطعة اكتمال عملية التحويل للمبدل والتي بدورها تخرج المتحكم من حالة الخمول إلى حالة العمل الطبيعي وسوف يقوم المتحكم بتنفيذ برنامج المقاطعة (قراءة القيمة المبدلة).
- قم بعدها بفصل المبدل "Stop ADC".

مخطط البنية الداخلية للمبدل ADC



حساب الجهد على دخل المبدل:

بما أن المبدلات التشابهية الرقمية لمتحكمات العائلة AVR هي بطول 10-bit أي 2^{10}=1024، فإن القيمة التي سيعطيها المبدل ستكون 1023 – 0 من أجل مجال جهد دخل المبدل VCC – 0.

تعطى العلاقة التي تحسب القيمة في مسجل المبدل (قيمة التبديل) بالشكل التالي:

$$ADC_{val} = \frac{V_{in} \times 1024}{V_{ref}}$$

حيث أنّ:

V_{in}: هو الجهد على مدخل قطب المبدل.

V_{ref}: هو الجهد المرجعي للمبدل.

من العلاقة السابقة يمكن إيجاد الجهد على دخل المبدل بالشكل:

$$V_{in} = \frac{ADC_{val} \times V_{ref}}{1024}$$

في حال استخدام المبدل ADC كمبدل إشارة تفاضلية فإن العلاقة تصبح على الشكل التالي:

$$ADC_{val} = \frac{(V_{Pos} - V_{Neg}) \times GAIN \times 512}{V_{ref}}$$

حيث أنّ:

V_{Pos}: هو الجهد الموجب على مدخل القطب الموجب للمبدل.

V_{Neg}: هو الجهد السالب على مدخل القطب السالب للمبدل.

GAIN: هو عامل الربح (الضرب) المختار للمبدل (1X, 10X, 200X).

في هذه الحالة سيتم تمثيل القيمة ADC_{val} بالشكل التالي:

$$0 \rightarrow 511 \gg Positive$$
$$512 \rightarrow 1023 \gg Negative$$

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 8000000
'-----------------------
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 =
Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E
= Portc.2 , Rs = Portc.3
Config Lcd = 20 * 4

Config Adc = Single , Prescaler = Auto ,
Reference = Avcc
Start Adc
'-----------------------
Dim W1 As Word , Voltage As Single
Dim Sw As String * 1
Const V_ref = 5
'-----------------------

Do
   Gosub Get_var
   Gosub Check_sw
   Gosub Show_temp
   Wait 1
Loop
'-----------------------

Get_var:
   W1 = Getadc(5)
   Voltage = W1 * V_ref
   Voltage = Voltage / 1024
Return
'-----------------------

Check_sw:
   Select Case W1
      Case 150 To 250 : Sw = "1"
      Case 300 To 400 : Sw = "2"
      Case 550 To 700 : Sw = "3"
      Case 750 To 850 : Sw = "4"

      Case Else : Sw = "-"
   End Select
Return
'-----------------------

Show_temp:
   Cls
   Locate 1 , 1 : Lcd "VAR1= " ; W1
   Locate 2 , 1 : Lcd "VOLT= " ; Voltage
   Locate 3 , 1 : Lcd "Switch NO. " ; Sw
Return
```

التوجيهات.

تعريف البوابة الموصل معها شاشة الإظهار الكريستالية.

تعريف المبدل التشابهي الرقمي

تعريف المتحولات

حلقة البرنامج الرئيسي يتم فيها:
استدعاء برنامج قراءة المبدل
استدعاء برنامج الطباعة على شاشة الإظهار

برنامج قراءة المبدل التشابهي الرقمي وحساب قيمة الجهد على قطب المبدل

برنامج تحديد المفتاح المضغوط بناءً على قيمة القراءة المحصلة من المبدل

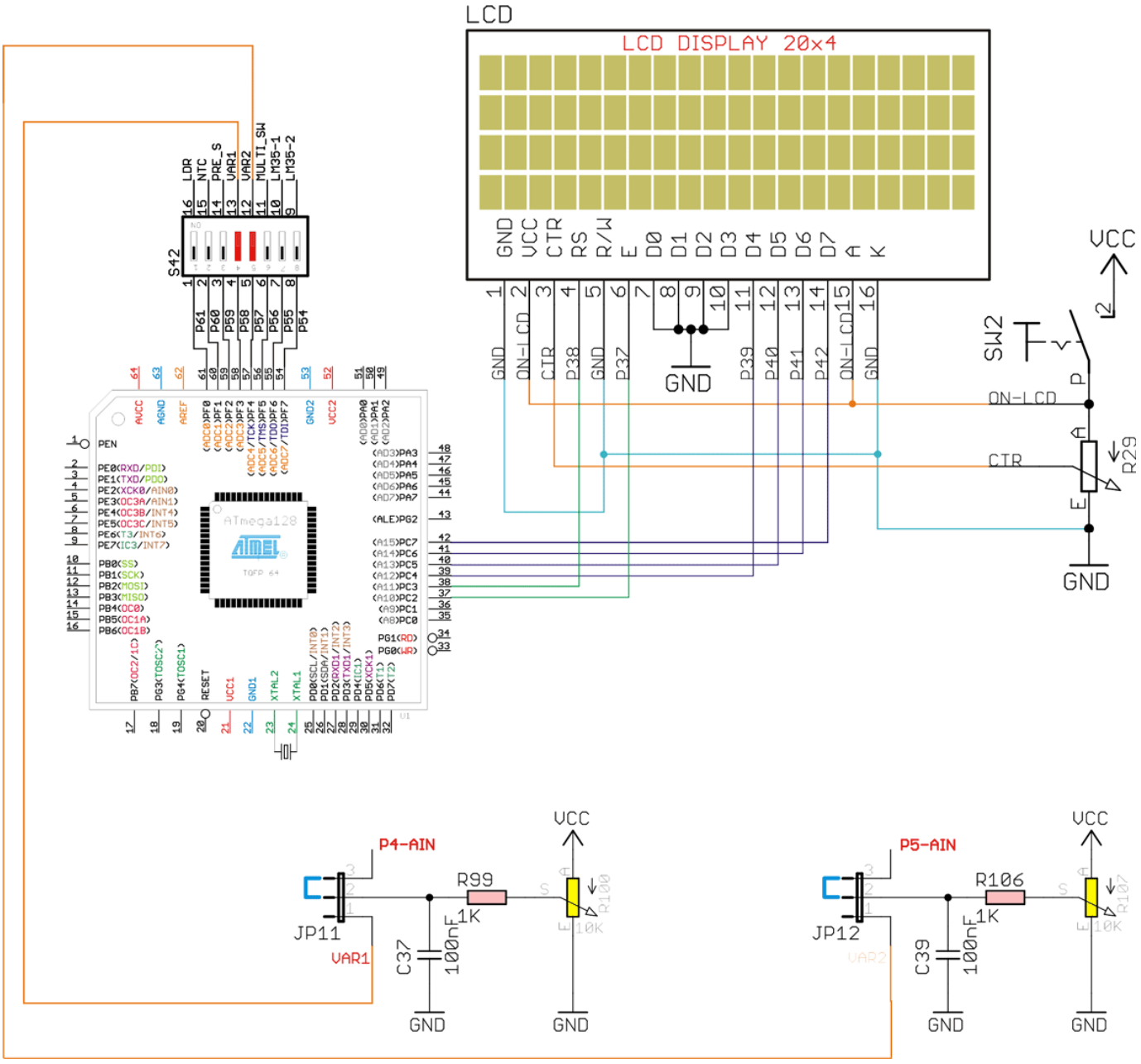
برنامج طباعة اسم المفتاح المضغوط على شاشة الإظهار

التجربة التاسعة عشرة: قياس جهد تشابهي خطي
Exp.19: Reading Analog Liner Voltage

الغاية من التجربة:

توصيل وبرمجة مقاومات تقسيم كمون متغيرة إلى قطب مبدل تشابهي رقمي.

مخطط التوصيل:



متطلبات التوصيل:

يجب إغلاق النقطة 4,5 من المفتاح S42. وإغلاق المفتاح SW2 لتغذية شاشة الإظهار.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لقراءة الجهد المطبق على مداخل المبدل والناتج عن تغيير موضع المقاومات المتغيرة R107, R100 ، والناتج في خرجهما هو جهد تشابهي خطي.

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 8000000
'-----------------------
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 =
Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E
= Portc.2 , Rs = Portc.3
Config Lcd = 20 * 4

Config Adc = Single , Prescaler = Auto ,
Reference = Avcc
Start Adc
'-----------------------
Dim W1 As Integer , W2 As Integer ,
Temperature As Integer
Dim Voltage1 As Single , Voltage2 As Single
Const V_ref = 5
'-----------------------

Do
   Gosub Get_var
   Gosub Show_var
   Wait 1
Loop
'-----------------------

Get_var:
   W1 = Getadc(3)
   W2 = Getadc(4)

   Voltage1 = W1 * V_ref
   Voltage1 = Voltage1 / 1024

   Voltage2 = W2 * V_ref
   Voltage2 = Voltage2 / 1024
Return
'-----------------------

Show_var:
   Cls
   Locate 1 , 1 : Lcd "VAR1= " ; W1
   Locate 2 , 1 : Lcd "VAR2= " ; W2
   Locate 3 , 1 : Lcd "VOLT1= " ; Voltage1
   Locate 3 , 1 : Lcd "VOLT2= " ; Voltage2
Return
```

التوجيهات.

تعريف البوابة الموصل معها شاشة الإظهار الكريستالية.

تعريف المبدل التشابهي الرقمي

تعريف المتحولات

حلقة البرنامج الرئيسي يتم فيها:
استدعاء برنامج قراءة المبدل
استدعاء برنامج الطباعة على شاشة الإظهار

برنامج قراءة المبدل التشابهي الرقمي وحساب قيمة الجهد على قطب المبدل لكل المقاومتين

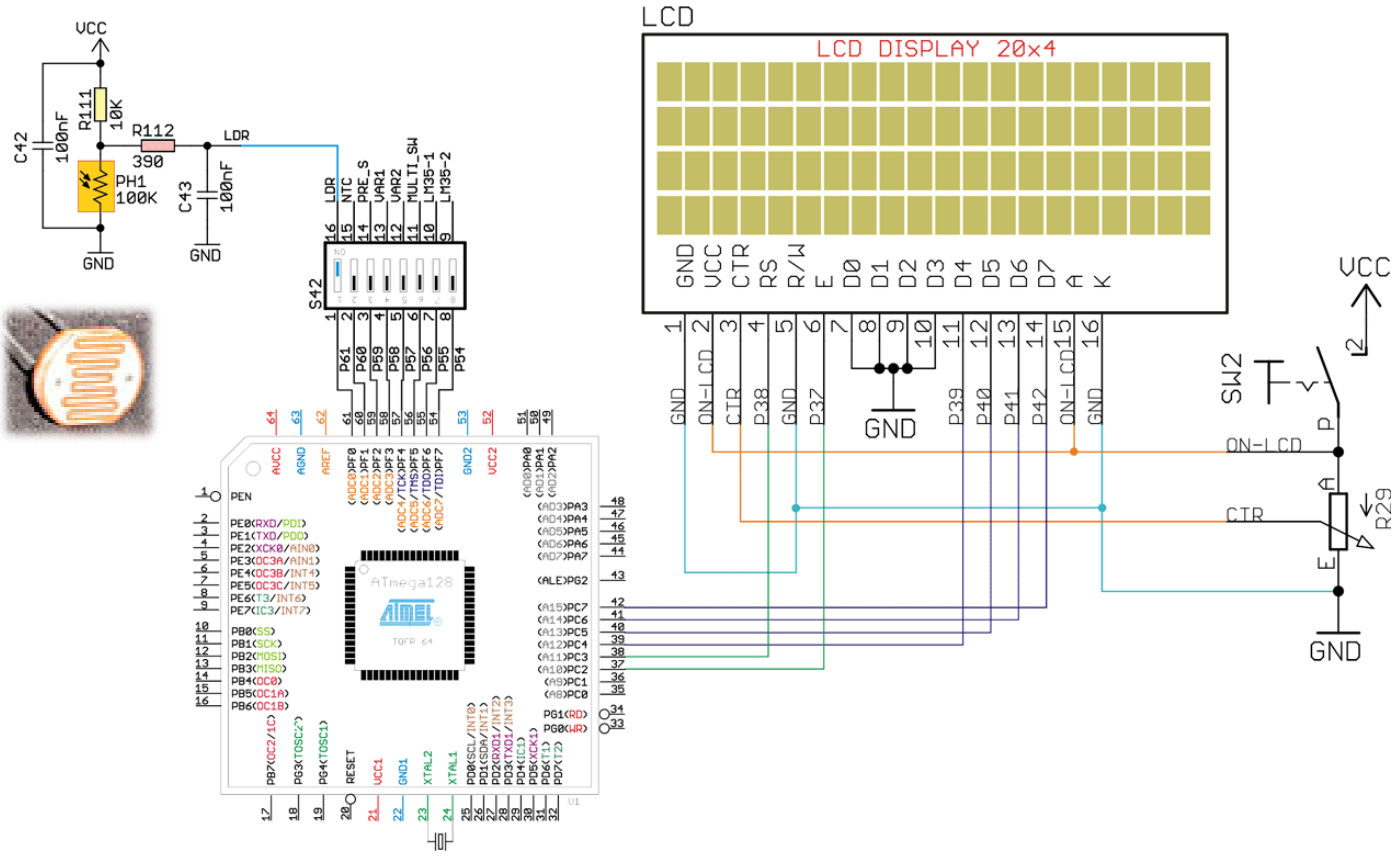
برنامج طباعة الجهود المحسوبة على شاشة الإظهار

التجربة العشرون: ربط مقاومة ضوئية وقياس شدة الإضاءة Exp.20: Interfacing LDR

الغاية من التجربة:

توصيل وبرمجة مقاومة ضوئية إلى قطب مبدل تشابهي رقمي بهدف قياس شدة الإضاءة للوسط المحيط.

مخطط التوصيل:



متطلبات التوصيل:

يجب إغلاق النقطة 1 من المفتاح S42. وإغلاق المفتاح SW2 لتغذية شاشة الإظهار.

شرح عمل الدارة:

سوف نقوم باستخدام مقاومة ضوئية لقياس شدة الضوء المحيط، وهي تقوم على تحويل الضوء إلى مقاومة.

تصنع هذه المقاومات من سلفيدالكاديوم (CDS) حيث تنخفض قيمتها الأومية عند ازدياد شدة الإضاءة حيث تصل قيمتها في الضوء الشديد إلى (100Ω)، وتزداد قيمتها عند انخفاض شدة الإضاءة حيث أنّ قيمتها الأعظمية في الظلام حوالي (2MΩ)، وتستطيع قياس شدة الإضاءة للموجات في المجال 350nm ~ 800nm.

إن العلاقة بين شدة الإضاءة (illumination) وقيمة المقاومة الضوئية (LDR) تعطى بالشكل التالي:

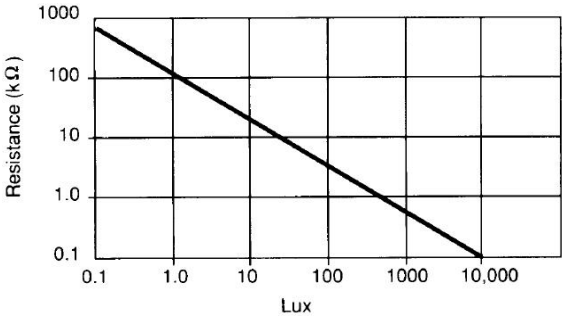
$$R_{LDR} = A \times L^{-0.85}$$

حيث أن:

A: هو ثابت يساوي $340x10^3$. L: هو شدة الإضاءة ويعطى بـ LUX.

الجدول التالي يوضح المعطيات والثوابت الأساسية للمقاومة الضوئية.

LDR Technical Specifications	
Dark Resistance	1MΩ
Resistance @ 10 Lux	10 – 20 KΩ
Resistance @ 100 Lux	2 – 4 KΩ
Peak Spectral response	540nm
Maximum Voltage	150V peak AC or DC
Power Dissipation	100mW
Operating Temperature	-30°C to + 70°C



برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 8000000
'-----------------------
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 =
Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E
= Portc.2 , Rs = Portc.3
Config Lcd = 20 * 4

Config Adc = Single , Prescaler = Auto ,
Reference = Avcc
Start Adc
'-----------------------
Dim Ldr_r As Single, W1 As Word
Dim Vin As Single
Dim Var1 As Single , Var2 As Single
Const V_ref = 5 : Const Vcc = 5
Const R_serial = 10000
'-----------------------

Do
   Gosub Get_adc
   Gosub Calc_r
   Gosub Show_val
   Wait 1
Loop
'-----------------------

Get_adc:
   W1 = Getadc(0)
Return
'-----------------------

Calc_r:
   Vin = W1 * V_ref
   Vin = Vin / 1024

   Var1 = Vcc - Vin
   Var2 = Vin * R_serial
   Ldr_r = Var2 / Var1
Return
'-----------------------

Show_val:
```

التوجيهات.

تعريف البوابة الموصل معها شاشة الإظهار الكريستالية.

تعريف المبدل التشابهي الرقمي

تعريف المتحولات

حلقة البرنامج الرئيسي يتم فيها:
استدعاء برنامج قراءة المبدل
استدعاء برنامج حساب قيمة المقاومة الضوئية
استدعاء برنامج الطباعة على شاشة الإظهار

برنامج قراءة المبدل التشابهي الرقمي

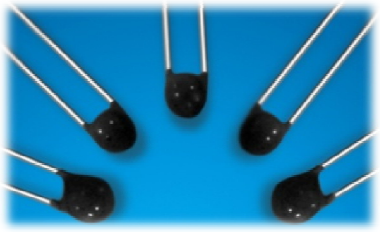
برنامج حساب الجهد على مدخل قطب المبدل وحساب قيمة المقاومة الضوئية
'LDR R=(Vin.10K)/(Vcc-Vin)

برنامج طباعة القيم المحسوبة على شاشة الإظهار

```
   Cls
   Locate 1 , 1 : Lcd "ADC= " ; W1
   Locate 2 , 1 : Lcd "Vin= " ; Vin
   Locate 3 , 1 : Lcd "LDR= " ; Ldr_r
Return
```

التجربة الحادية والعشرون: قياس الحرارة باستخدام NTC

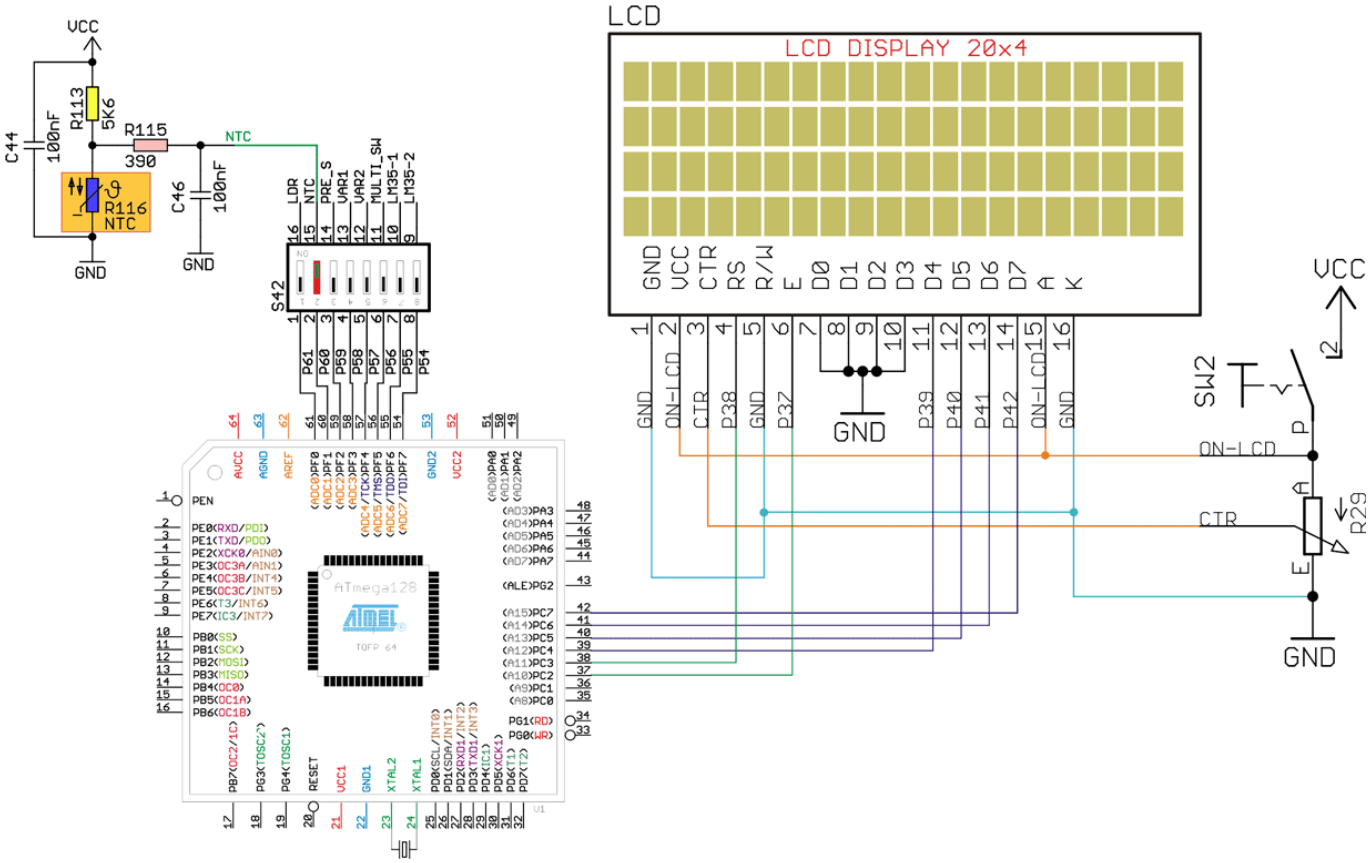
Exp.21: Interfacing NTC Resistor



الغاية من التجربة:

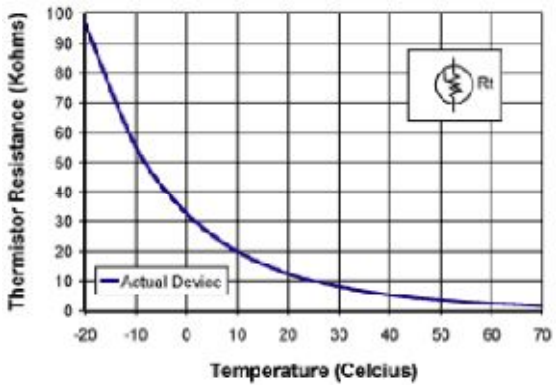
استخدام المقاومة NTC في تطبيقات قياس درجات الحرارة العالية.

مخطط التوصيل:



متطلبات التوصيل:

يجب إغلاق النقطة 2 من المفتاح S42. وإغلاق المفتاح SW2 لتغذية شاشة الإظهار.



شرح عمل الدارة:

سوف نقوم بتوصيل مقاومة NTC (Negative Temperature Coefdient) مع متحكم مصغر بهدف استخدام العلاقة اللاخطية بين تغير درجة الحرارة وقيمة المقاومة في قياس درجات

الحرارة.

بشكل عام فإن المقاومة ذات المعامل الحراري السالب عبارة عن مقاومة لا خطية تكون قيمتها عند درجة الحرارة 25°C معطاة بالقيمة الاسمية للمقاومة (10K)، ومع ازدياد درجة الحرارة تنقص قيمتها، وبنقصان درجة الحرارة تزداد قيمتها، وهذا التغير في قيمة المقاومة يكون بشكل لا خطي.

تعطى قيمة المقاومة عند تغير درجة الحرارة بالعلاقة التالية:

$$R_T = R_{25C} \times e^{\beta\left[\left(\frac{1}{T+273}\right)-\left(\frac{1}{T+298}\right)\right]}$$

R_{25C}: هي قيمة المقاومة عند الدرجة 25°C

β: ثابت يمكن الحصول عليه من الوثيقة الفنية للمقاومة (3450).

من العلاقة السابقة يمكننا استنتاج العلاقة التي تحسب قيمة درجة الحرارة من أجل مقاومة معلومة عند ربطها مع قطب المبدل التشابهي للمتحكم.

$$T_{\circ C} = \frac{\beta}{\ln\left(\frac{V_{ADC}}{V_{ref}-V_{ADC}}\right)+\frac{\beta}{T_{amb}}} - T_{zero}$$

حيث أنّ:

V_{ADC}: هو الجهد المقاس والمحسوب على طرفي المقاومة.

V_{REF}: هو الجهد المرجعي للمبدل التشابهي الرقمي.

T_{zero}: درجة الحرارة في الصفر المطلق وتساوي 273°K.

T_{amb}: درجة الحرارة المعيارية للمقاومة 273°K + 25°C = 298°K.

يمكن تبسيط العلاقة السابقة إلى الشكل التالي:

$$T_{\circ C} = \frac{\beta}{\ln\left(\frac{ADC_{val}}{1024-ADC_{val}}\right)+\frac{\beta}{T_{amb}}} - T_{zero}$$

حيث أنّ:

ADC_{val}: هي القيمة المقروءة للمبدل التشابهي الرقمي.

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 8000000
'-----------------------
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 =
Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E
= Portc.2 , Rs = Portc.3
Config Lcd = 20 * 4

Config Adc = Single , Prescaler = Auto ,
Reference = Avcc
Start Adc
'-----------------------
Dim W1 As Word , Voltage As Single
Dim Temp As Single
Dim F1 As Single, Ntc_volt As Single
Const V_ref = 5
'-----------------------
Do
   Gosub Get_temp
   Gosub Show_temp
   Wait 1
Loop
'-----------------------
Get_temp:
   W1 = Getadc(1)

   Ntc_volt = W1 * V_ref
   Ntc_volt = Ntc_volt / 1024

   F1 = 1024 - W1
   F1 = W1 / F1
   F1 = Log(f1)
   F1 = F1 + 14.2617
   F1 = 4250 / F1
   Temp = F1 - 273
Return
'-----------------------
Show_temp:
   Cls
   Locate 1 , 1 : Lcd "VAR1= " ; W1
   Locate 2 , 1 : Lcd "VOLT= " ; Ntc_volt
   Locate 3 , 1 : Lcd "TEMP= " ; Temp
Return
```

التوجيهات.

تعريف البوابة الموصل معها شاشة الإظهار الكريستالية.

تعريف المبدل التشابهي الرقمي

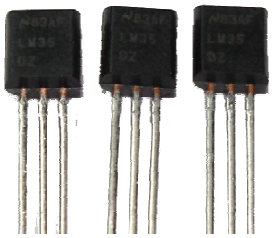
تعريف المتحولات

حلقة البرنامج الرئيسي يتم فيها:
استدعاء برنامج قراءة المبدل
استدعاء برنامج الطباعة على شاشة الإظهار

برنامج قراءة المبدل التشابهي الرقمي وحساب قيمة المقاومة ودرجة الحرارة الموافقة.
'Beta/Tamp = 4250/298 = 14.2167

برنامج طباعة القيم المحسوبة على شاشة الإظهار

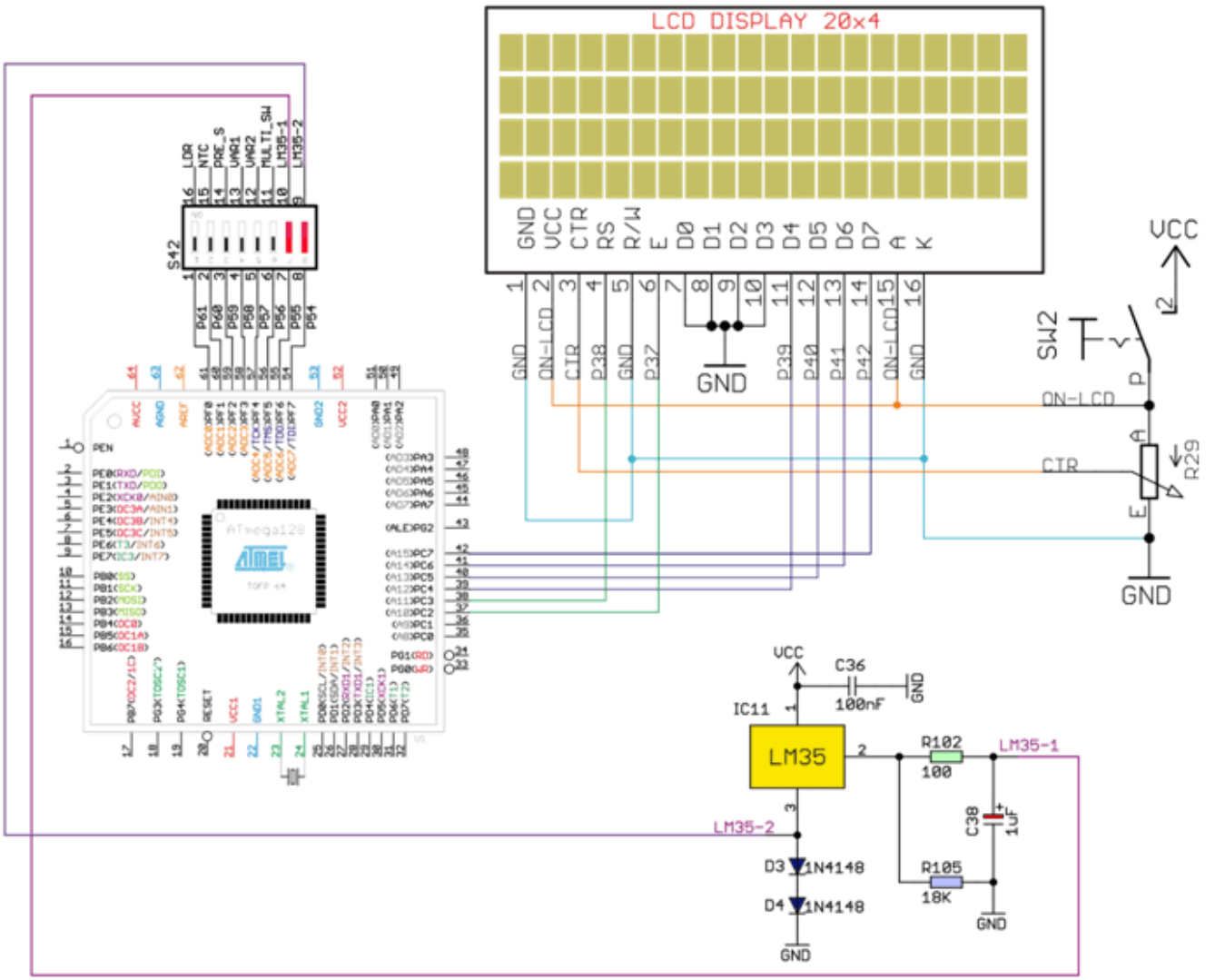
Exp.22: Interfacing LM35 التجربة الثانية والعشرون: قياس الحرارة باستخدام LM35



الغاية من التجربة:

استخدام الحساس التشابهي LM35DZ في تطبيقات قياس درجات الحرارة في المجال من -45°C ~ +145°C.

مخطط التوصيل:



متطلبات التوصيل:

يجب إغلاق النقطة 7,8 من المفتاح S42. وإغلاق المفتاح SW2 لتغذية شاشة الإظهار.

شرح عمل الدارة:

سوف نقوم باستخدام الحساس التشابهي LM35 لقياس درجات الحرارة وذلك بعد إزاحة النقطة الصفرية للحساس بمقدار (1.2 V) بواسطة ثنائيين موصولين على التسلسل بهدف قياس درجات الحرارة السالبة.

إن العلاقة بين درجة الحرارة وجهد خرج الحساس تعطى بالشكل التالي:

$$1℃ \rightarrow 10mV$$

وبالتالي يمكن إيجاد معادلة الحساس والمبدل بالتعويض في معادلة المبدل التشابهي.

من أجل درجة مؤوية واحدة يكن الجهد على دخل المبدل 10mV وبالتالي فإن كل درجة حرارة يقابلها:

$$ADC_{val} = \frac{V_{in} \times 1024}{V_{ref}} = \frac{0.01 \times 1024}{2.56} = 4$$

وهذا يعني أنه يكفي أن نقسم قيمة قراءة المبدل على 4 لنحصل على درجة الحرارة الحقيقية.

برنامج تشغيل الدارة:

التوجيهات.	`$regfile = "m128def.dat"` `$crystal = 8000000`

تعريف البوابة الموصل معها شاشة الإظهار الكريستالية.	`'-----------------------`
	`Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.2 , Rs = Portc.3`
	`Config Lcd = 20 * 4`

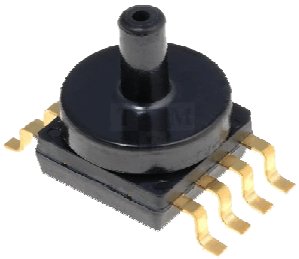
تعريف المبدل التشابهي الرقمي	`Config Adc = Single , Prescaler = Auto , Reference = Internal` `Start Adc`
تعريف المتحولات	`'-----------------------` `Dim W1 As Integer , W2 As Integer , Temperature As Integer`

حلقة البرنامج الرئيسي يتم فيها:
استدعاء برنامج قراءة المبدل
استدعاء برنامج الطباعة على شاشة الإظهار
	`'-----------------------`
	`Do`
	`  Gosub Get_temp`
	`  Gosub Show_temp`
	`  Wait 1`
	`Loop`

برنامج قراءة المبدل التشابهي الرقمي وحساب قيمة درجة الحرارة
`' 4 steps = 1 degree`
	`'-----------------------`
	`Get_temp:`
	`  W1 = Getadc(6)`
	`  W2 = Getadc(7)`
	`  Temperature = W1 - W2`
	`  Temperature = Temperature / 4`
	`Return`

برنامج طباعة القيمة المحسوبة على شاشة الإظهار	`'-----------------------`
	`Show_temp:`
	`  Cls`
	`  Lcd "Temp= " ; Temperature ; " C"`
	`Return`

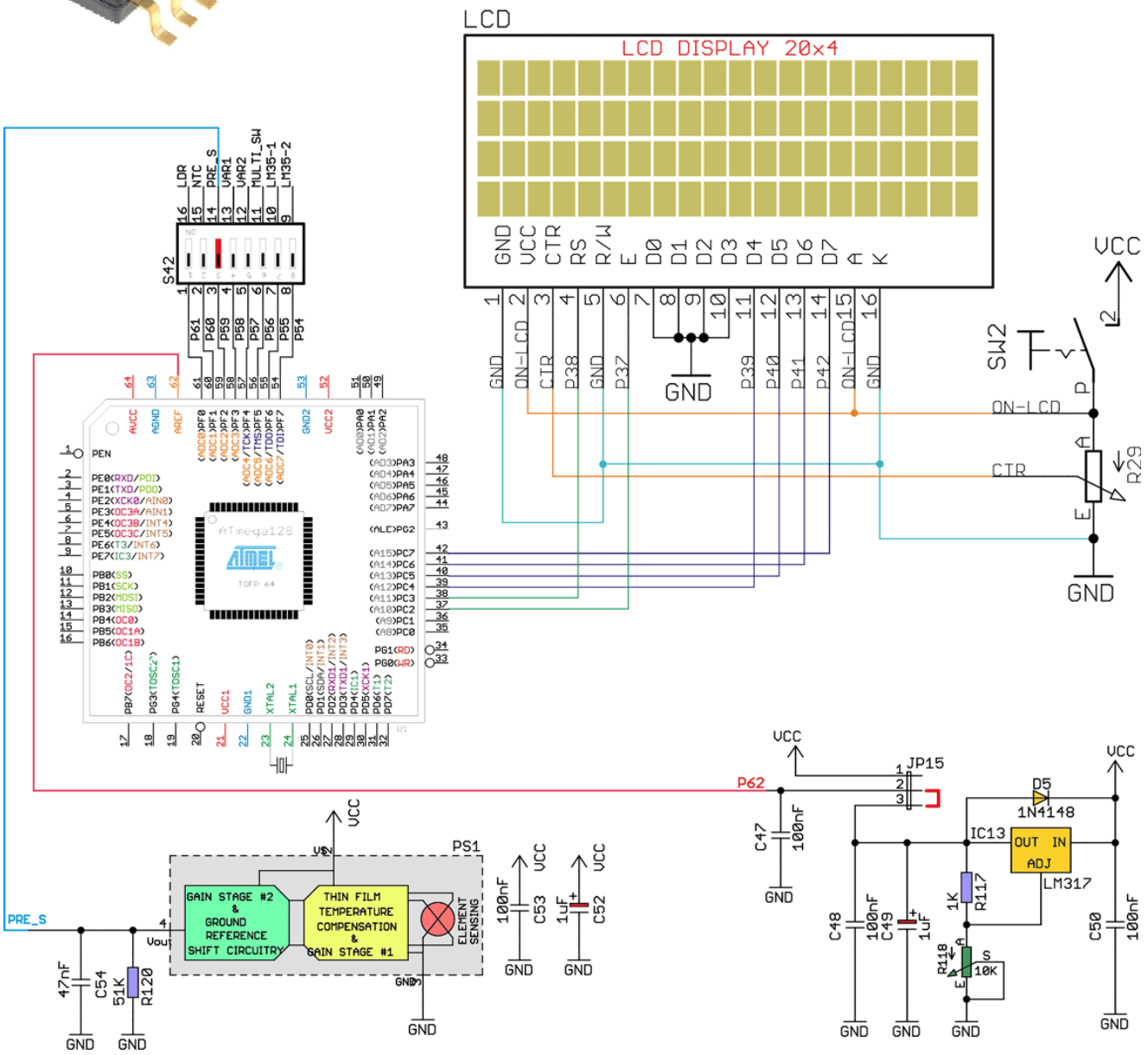
Exp.23: Interfacing Pressure Sensor التجربة الثالثة والعشرون: قياس الضغط الجوي



الغاية من التجربة:

توصيل وبرمجة مقياس ضغط تشابهي لأغراض قياس الضغط الجوي.

مخطط التوصيل:



متطلبات التوصيل:

يجب إغلاق النقطة 3 من المفتاح S42. وإغلاق المفتاح SW2 لتغذية شاشة الإظهار.

شرح عمل الدارة:

سوف نقوم باستخدام حساس ضغط تشابهي (MPXAZ6115A) مصنع من قبل شركة "Freescale" لقياس الضغط الجوي والارتفاع عن سطح البحر.

إن مجال قياس هذا الحساس هو (kilopascals)kPa 115 ~ 15 مع العلم أن الضغط الجوي القياسي (للهواء) يساوي إلى (atmosphere)**1-atm** أو **101.325-kPa**.

Atmosphere = 101,325 * Pascal

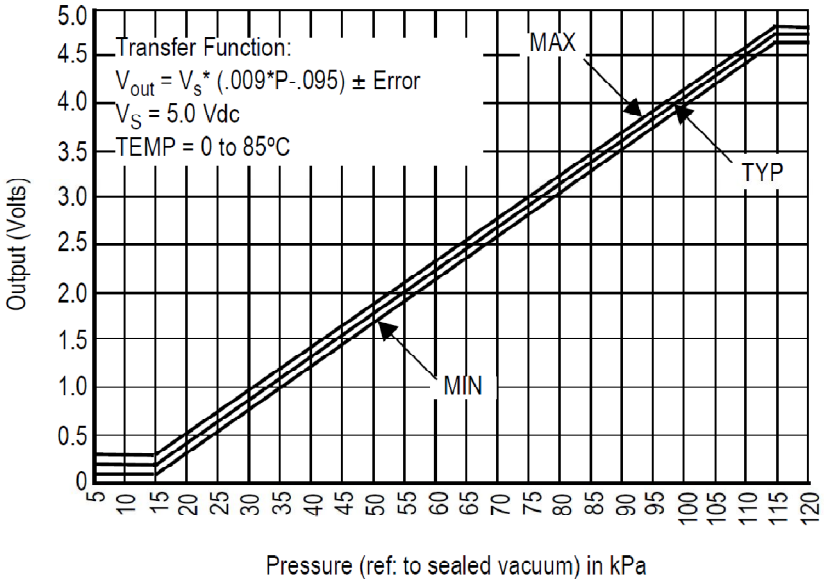
Atmosphere = 1.013 * bar

إن خرج الحساس (التشابهي) يتراوح من **0.2V ~ 4.8V** من أجل تغير الضغط **15 ~ 115 kPa**

إن العلاقة التي تعبر تربط بين جهد خرج الحساس والضغط الجوي المقاس لها الشكل التالي:

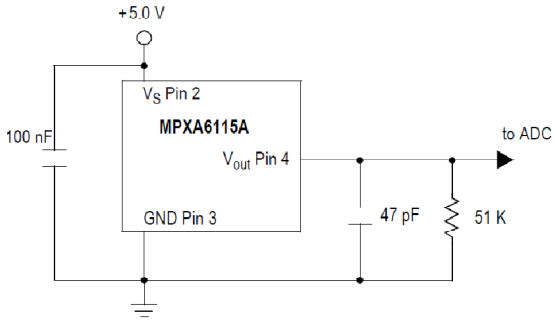
$$V_{out} = V_s \times (0.009 \times P - 0.095) \mp (Pressure\ Error \times Temp\ Error \times 0.009 \times V_s)$$

$$V_s = 5.0 \mp 0.25\ V_{DC}$$



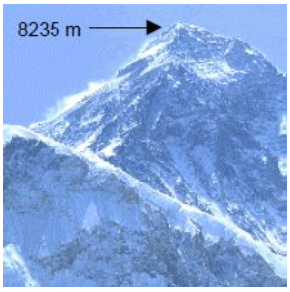
من أجل تسهيل المعادلة فإنه يمكن تجاهل القسم الأيمن الذي يعالج مسألة خطأ الدقة في القياس إذ أنّ خطأ خرج الحساس لا يتجاوز 1.5kPa على كامل المجال.

$$V_{out} = V_s \times (0.009 \times P - 0.095)$$



وبالتالي يمكن حساب الضغط بالشكل التالي:

$$P = \frac{V_{out}}{0.009 \times V_s} + \frac{0.095}{0.009}$$



إن قيمة الضغط الجوي ينقص بمقدار 7mbar كلما ارتفعنا مسافة 100m عن سطح البحر، حيث يبلغ عند أعلى قمة على الأرض (8848m) حوالي 310mbar مع العلم أن هذا التناسب غير خطي، وبالتالي انطلاقاً من الضغط الجوي المقاس فإنه يمكن إيجاد الارتفاع عن سطح البحر (Altimeters) بمعادلة التحويل (Pressure-to-altitude) التالية:

$$h = \frac{288.15K}{0.0065K/m} \cdot \left\{1 - \left(\frac{P}{101325Pa}\right)^{0.0065K/m \cdot \frac{R}{g}}\right\}$$

Zero altitude pressure = 101325Pa = 1'013.25mbar; (100Pa = 1mbar)

Zero altitude temperature = 288.15K

Temperature gradient = 6.5K/1000m

R is the specifc gas constant : $R = \frac{R^*}{M0} = 287.052 \frac{J}{K.kg}$

إن معادلة التحويل أعلاه صالحة حتى ارتفاع 12km، كما أنها لا تأخذ بعين الاعتبار تغيرات الجو (خصوصاً خلال فصل الشتاء)، وبالتالي فإن تراكم الأخطاء في معادلة التحويل نتيجة للعوامل المتغيرة يمكن أن يؤدي إلى انحراف قياس حوالي +/-100m عند التغيرات الكبيرة.

برنامج تشغيل الدارة:

التوجيهات.	`$regfile = "m128def.dat"` `$crystal = 8000000`

تعريف البوابة الموصل معها شاشة الإظهار الكريستالية.	`'-----------------------`
	`Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.2 , Rs = Portc.3`
	`Config Lcd = 20 * 4`

تعريف المبدل التشابهي الرقمي	`Config Adc = Single , Prescaler = Auto , Reference = Avcc` `Start Adc`

تعريف المتحولات	`'-----------------------`
	`Dim W1 As Word , Voltage As Single , Pressure As Single`
	`Const V_ref = 5`

حلقة البرنامج الرئيسي يتم فيها :
استدعاء برنامج قراءة المبدل
استدعاء برنامج الطباعة على شاشة الإظهار
	`'-----------------------`
	`Do`
	`  Gosub Get_temp`
	`  Gosub Show_temp`
	`  Wait 1`
	`Loop`

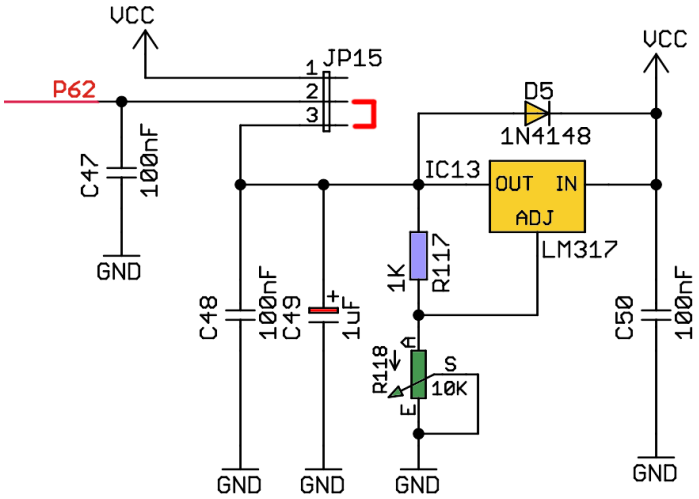
برنامج قراءة المبدل التشابهي الرقمي
وحساب الجهد على مدخل قطب المبدل وحساب الضغط الجوي
	`'-----------------------`
	`Get_temp:`
	`  W1 = Getadc(2)`
	`  Voltage = W1 * V_ref`
	`  Voltage = Voltage / 1024`
	`  Pressure = Voltage / 0.045`
	`  Pressure = Pressure + 10.55`

برنامج طباعة القيم المحسوبة على شاشة الإظهار	`Return`
	`'-----------------------`
	`Show_temp:`
	`  Cls`

```
   Locate 1 , 1 : Lcd "VAR1= " ; W1
   Locate 2 , 1 : Lcd "VOLT= " ; Voltage
   Locate 3 , 1 : Lcd "PRES= " ; Pressure
Return
```

دارات الجهد المرجعي:

على المخطط النظري السابق للدارة يوجد قسم دارة جهد مرجعي متغير تعتمد على منظم الجهد المتغير LM317.



عن طريق تغيير قيمة المقاومة المتغيرة R118 فإنه يمكن ضبط الجهد على النقطة P62 من 1.3V~5V

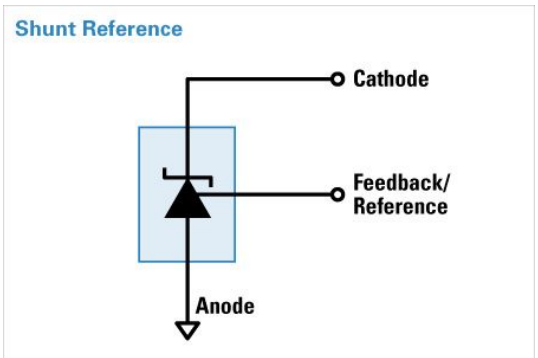
إن المنظم LM317 لا يعتبر من منظمات الجهد المرجعي الدقيقة مثل المنظم LM385.

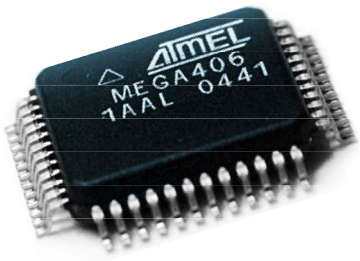
يتوفر المنظم LM385 بلواحق مختلفة مثل:

LM385-1.3: وهو منظم 1.3V.

LM385-2.5: وهو منظم 2.5V.

بالإضافة إلى المنظم LM385 بجهد خرج قابل للضبط من 1.1V~5V





برمجة المتحكمات المصغرة

التجارب العملية

الجلسة الثامنة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

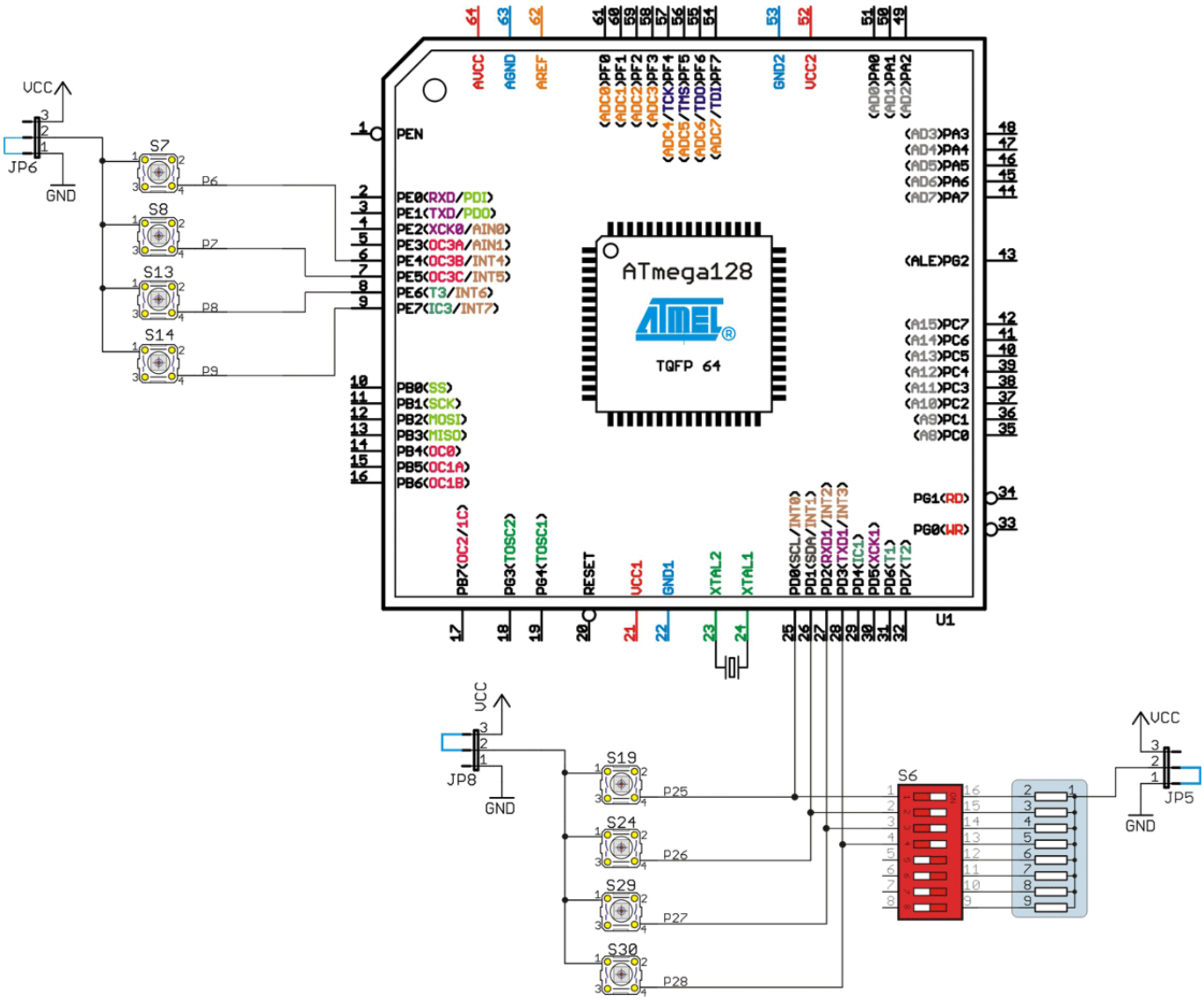
WALID BALID, Tuesday, December 15, 2009

التجربة الرابعة والعشرون: المقاطعات الخارجية
Exp.24:External Interrupts

الغاية من التجربة:

توصيل وبرمجة مجموعة مفاتيح إلى أقطاب المقاطعات الخارجية INT7 ~ INT0.

مخطط التوصيل:



متطلبات التوصيل:

يجب إغلاق النقطة JP5, JP6, JP8 إلى النقطة الأرضية.

شرح عمل الدارة:

سوف نقوم بكتابة برنامج لقراءة مفاتيح موصولة إلى أقطاب المقاطعات الخارجية.

إن الفكرة الأساسية هي أن المعالج لن ينشغل بتفحص المفاتيح من أجل معرفة حالة المفتاح، وإنما عندما تتغير الحالة المنطقية للمفتاح على قطب المقاطعة سوف يتم مقاطعة المعالج ويقفز إلى برنامج خدمة المقاطعة الخارجية المتعينة من أجل تنفيذها.

إن الطريقة التقليدية لقراءة حالة مفتاح موصول إلى قطب متحكم مصغر يمكن أن تتم بأشكال متعددة:

```
Do
      If Key1 = 0 Then
         ...
         ...
      End If

      Debounce Key1 , 0 , Sw1 , Sub

      Bitwait Key1, reset
Loop
```

التحقق من حالة المفتاح باستخدام التعليمات الشرطية

التحقق من حالة المفتاح باستخدام Debounce

التحقق من حالة المفتاح باستخدام Bitwait

إن جميع الطرق أعلاه سوف تشغل المعالج في عملية الفحص الدوري للتحقق من حالة المفتاح، أما في حال استخدمنا لمقاطعات الخارجية، فإن المعالج يقوم بمعالج الحالة فقط عندما تتحقق المقاطعة.

التعليمات الجديدة:

شرح التعليمة	التعليمة البرمجية
إعداد وتعريف المقاطعة الخارجية (x=0,1,~,7). يمكن تحديد أربعة حالات (State) للاستجابة للمقاطعة كما يلي: Falling: يتم توليد المقاطعة عند الجبهة الهابطة للنبضة المطبقة على القطب Rising: يتم توليد المقاطعة عند الجبهة الصاعدة للنبضة المطبقة على القطب Low Level: يتم توليد مقاطعة طالما أن المستوى "0" مطبقة على القطب Change: يتم توليد المقاطعة عند تغير الجبهة للنبضة المطبقة على القطب	`Config Intx = State`
عند تحقق المقاطعة يتم القفز إلى برنامج خدمة المقاطعة الموجود عن اللافتة label وينفذ برنامج خدمة المقاطعة ويعود إلى البرنامج الرئيسي.	`On Intx label`
تفعيل الاستجابة للمقاطعة الخارجية	`Enable Intx`
إلغاء تفعيل الاستجابة للمقاطعة الخارجية	`Disable Intx`
تفعيل شعاع المقاطعات العام	`Enable Interrupts`

ملاحظة: في حال كان المعالج يقوم بتنفيذ برنامج خدمة مقاطعة ما، وفي نفس الوقت حصلت مقاطعة أخرى، فإن المعالج سوف يكمل المقاطعة الجارية ويقوم بتخزين المقاطعة الطارئة حتى إذا انتهى من المقاطعة الجارية عاد إلى البرنامج الرئيسي واستدعى المقاطعة الطارئة وقام بتنفيذها.

ملاحظة: إذا حصلت عدة مقاطعات أثناء عمل المعالج في برنامج خدمة مقاطعة ما فإنه يقوم بمراكمتها حسب أولويتها ويقوم بتنفيذها وفق تسلسل الأولوية بعد انتهائه من برنامج خدمة المقاطعة الجارية.

ملاحظة: في حال أريد رفض أي مقاطعة أخرى (Int0 مثلاً) أثناء تنفيذ برنامج خدمة المقاطعة عينها، فإنه يجب إلغاء تفعيل شعاع المقاطعة (Disable Int0) أثناء معالجة برنامج المقاطعة المذكورة، وبعد الانتهاء يتم إعادة تفعيل شعاع المقاطعة (Enable Int0).

ملاحظة: ينصح بأن يكون برنامج خدمة المقاطعة قصيراً جداً (يمكن تفعيل علم تحقق المقاطعة وتفحص العلم في البرنامج الرئيسي وتنفيذ جملة تعليمات تبعاً لحالة علم المقاطعة) وجميع المعالجات تتم في البرنامج الوظيفي.

ملاحظة: عند استخدام أي مقاطعة (داخلية أو خارجية) فإنه يجب تفعيل شعاع المقاطعات العام والذي يمكن تمثيله كقاطع رئيسي لجميع المقاطعات (**Enable Interrupts**).

الجدول التالي يبين ترتيب أولوية المقاطعات.

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty
21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator
25	$0030(3)	TIMER1 COMPC	Timer/Counter1 Compare Match C
26	$0032(3)	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034(3)	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036(3)	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038(3)	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A(3)	TIMER3 OVF	Timer/Counter3 Overflow
31	$003C(3)	USART1, RX	USART1, Rx Complete
32	$003E(3)	USART1, UDRE	USART1 Data Register Empty
33	$0040(3)	USART1, TX	USART1, Tx Complete
34	$0042(3)	TWI	Two-wire Serial Interface
35	$0044(3)	SPM READY	Store Program Memory Ready

برنامج تشغيل الدارة:

```
$regfile = "m128def.dat"
$crystal = 1000000
$baud = 9600
'-----------------------
Config Int0 = Rising
Config Int1 = Rising
Config Int2 = Change
Config Int3 = Change
Config Int4 = Falling
Config Int5 = Falling
Config Int6 = Change
Config Int7 = Low Level

On Int0 Int0_isr
On Int1 Int1_isr
On Int2 Int2_isr
On Int3 Int3_isr
On Int4 Int4_isr
On Int5 Int5_isr
On Int6 Int6_isr
On Int7 Int7_isr Nosave

Porte = &B11110000 : Portd = &B00000000

Enable Int0 : Enable Int1
Enable Int2 : Enable Int3
Enable Int4 : Enable Int5
Enable Int6 : Enable Int7
Enable Interrupts
'-----------------------
Do
   nop
Loop Until Inkey() = 27
End
'-----------------------
Int0_isr:
   Disable Interrupts
   Print "Int0 Occurred at Falling Edge!"
   Enable Interrupts
Return
'-----------------------
Int1_isr:
   Print "Int1 Occurred at Falling Edge!"
Return
'-----------------------
Int2_isr:
   Print "Int2 Occurred at Low Level!"
Return
'-----------------------
Int3_isr:
   Print "Int3 Occurred at Low Level!"
Return
'-----------------------
Int4_isr:
   Print "Int4 Occurred at Rising Edge!"
Return
'-----------------------
Int5_isr:
   Print "Int5 Occurred at Rising Edge!"
Return
'-----------------------
Int6_isr:
   Disable Int6
   Print "Int6 Occurred at Change of Edge!"
Return
'-----------------------
Int7_isr:
   Disable Int7
   Print "Int7 Occurred at Change of Edge!"
Return
```

التوجيهات.

إعداد وتعريف جميع المقاطعات الخارجية (x=0,1,~,7) للمتحكم ATmega128

عند تحقق المقاطعة يتم القفز إلى برنامج خدمة المقاطعة الموافقة الموجود عن اللافتة label

تفعيل مقاومات الرفع الداخلية للمقاطعات التي ستحدث عند الجبهات الهابطة وإلغائها من أجل المقاطعات عند الجبهات الصاعدة.

تفعيل المقاطعات الخارجية الثمانية.

حلقة البرنامج الرئيسي لا يتم فيها أي عملية

برنامج خدمة المقاطعة الخارجية INT0 الذي يتم تنفيذه عند ورود جبهة صاعدة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT1 الذي يتم تنفيذه عند ورود جبهة صاعدة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT2 الذي يتم تنفيذه عند تغير الجبهة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT3 الذي يتم تنفيذه عند تغير الجبهة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT4 الذي يتم تنفيذه عند ورود جبهة هابطة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT5 الذي يتم تنفيذه عند ورود جبهة هابطة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT6 الذي يتم تنفيذه عند ورود تغير الجبهة على قطب هذه المقاطعة

برنامج خدمة المقاطعة الخارجية INT7 الذي يتم تنفيذه طالما أن المستوى low على قطب هذه المقاطعة

التجربة الخامسة والعشرون: التعامل مع المؤقتات Exp.25: Dealing with Timers

الغاية من التجربة:

التعرف إلى أنماط عمل المؤقت Timer0 والمؤقت Timer1.

المؤقتات/العدادات:

تملك متحكمات العائلة AVR مؤقتات زمنية/عدادات ذات أداء عالي وميزات عديدة، كما أن موضوع المؤقتات/عدادات في متحكمات العائلة AVR هو موضوع واسع جداً لما لهذه المؤقتات من أنماط عمل متعددة بالإضافة لوجود أكثر من مؤقت مختلف الطول على نفس الشريحة.

ما هو المؤقت/عداد؟

بأبسط تعبير يمكننا القول بأن المؤقت/عداد عبارة عن مسجل يختلف طوله بحسب نوع المؤقت، غالباً تكون المؤقتات بطول (دقة) 8bit أو 16bit.

من أجل مؤقت/عداد 8bit فإن المؤقت يستطيع العد من 0 – 255، وأما من أجل مؤقت/عداد بطول 16bit فإن المؤقت يستطيع العد من 0 – 65536.

يملك المؤقت/عداد مدخل قدح (Clock) حيث يقوم بالعد (تصاعدياً أو تنازلياً) عند جبهات الساعة الداخلية للهزاز الكريستالي، وهذه النبضات يمكن تقسيمها عبر مقسم ترددي دخلي في بنية المؤقت (Prescaler)، وبالتالي لا حاجة لإشغال المعالج بزيادة أو إنقاص قيمة المؤقت والمؤقت سوف يعمل بشكل مستقل تماماً.

بما أن المؤقت/عداد يعمل بشكل مستقل عن المعالج، فإن المؤقت/عداد يملك مقاطعات من أجل إعلام المعالج بإتمام الوظيفة التي أعد المؤقت/عداد من أجلها.

على سبيل المثال مقاطعة الطفحان (Overflow)، حالما تصبح القيمة في مسجل المؤقت/عداد مساوية إلى القيمة العظمى (255>8bit | 65535>16bit)، فإن المؤقت/عداد يقوم بتوليد مقاطعة تسمى مقاطعة الطفحان (يجب أن تكون معدة بشكل مسبق) يقوم فيها بإعلام المعالج أنه انتهى من العد إلى القيمة العظمى.

المقسم الترددي (Prescaler):

هو عبارة عن آلية ذات مسجل بطول 10bit تقوم بتقسيم تردد الهزاز الكريستالي للمعالج من أجل تطبيقه كتردد عمل للمؤقت/عداد، بمعنى آخر إن هذا التردد المطبق على قطب الـ Clock للمؤقت/العداد هو للتزامن؛ إن قيم التقسيم الترددي المتاحة موضحة بالجدول التالي:

قيمة التقسيم	تردد عمل المؤقت بعد التقسيم
Prescaler = 1	$\mathbf{TIMER_{Clock} = F_{CPU}}$
Prescaler = 8	$\mathbf{TIMER_{Clock} = F_{CPU} / 8}$
Prescaler = 64	$\mathbf{TIMER_{Clock} = F_{CPU} / 64}$
Prescaler = 256	$\mathbf{TIMER_{Clock} = F_{CPU} / 256}$
Prescaler = 1024	$\mathbf{TIMER_{Clock} = F_{CPU} / 1024}$

إن الإعداد السابق للمؤقت يعتمد بالكلية على تردد الهزاز الكريستالي للمعالج، بالإضافة إلى نمط الإعداد لمصدر التواقت للمؤقت، بالإضافة إلى ذلك يمكن أيضاً إعداد المؤقت ليتم قدحه عن طريق تطبيق مصدر نبضات (Clock) على قطب المؤقت/العداد الخارجي (T0, T1) وبالتالي سيعمل كعداد.

حساب الزمن الأعظمي للمؤقت:

إن العلاقة التي تحسب القيمة الأعظمية لزمن المؤقت حتى الوصول إلى القمة (الطفحان) بناءً على تردد الهزاز الكريستالي وقيمة المقسم الترددي المحدد تعطى بالشكل التالي:

$$T = 2^N \frac{Prescaler}{fosc}$$

حيث أنّ:

T: الزمن الأعظمي لحدوث الطفحان للمؤقت وتعطى بالثانية.

N: دقة (طول) المؤقت؛ $N_{TIMER0}=8$، $N_{TIMER1}=16$.

Prescaler: قيمة المقسم الترددي المحددة [1, 8, 64, 256 or 1024].

f_{OSC}: تردد الهزاز الكريستالي.

حساب دقة المؤقت:

إن الدقة هنا تعني أصغر زمن يستطيع المؤقت قياسه (عده) وهو عبارة عن دور نبضة توقيت واحدة مطبقة على مدخل الـ Clock للمؤقت ويعطى بالعلاقة التالية:

$$Timer_{RESOLUTION} = \frac{1}{Input\ Ferequesncy}$$

مثال: بفرض أن تردد الهزاز الكريستالي للمعالج f_{OSC}=2MHZ وتم إعداد المقسم الترددي Prescaler=64 وبالتالي فإن تردد عمل المؤقت (Clock) يساوي:

$$Timer_{CLOCK} = \frac{fosc}{Prescaler} = \frac{2000000}{64} = 31250HZ$$

وبالتالي فإن دقة المؤقت تساوي:

$$Timer_{RESOLUTION} = \frac{1}{Input\ Ferequesncy} = \frac{1}{31250} = 0.000032\ Sec = 32\mu S$$

حيث أن الفترة 23uS هي أصغر زمن يستطيع المؤقت قياسه وفق الشروط المحددة أعلاه.

حساب القيم الأعظمية والدقة للأزمنة:

الجدول التالي يلخص القيم الأعظمية والدقة لأزمنة كلاً من Timer0, Timer1 من أجل عدة ترددات قياسية.

Timing of Timer0 at f_{OSC} = 1MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.001	0.008	0.064	0.256	1.024
Maximum Timer Period (mSec)	0.256	2.048	16.384	65.536	262.144

Timing of Timer0 at f_{OSC} = 2MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.0005	0.004	0.032	0.128	0.512
Maximum Timer Period (mSec)	0.128	1.024	8.192	32.768	131.072

Timing of Timer0 at f_{OSC} = 4MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.00025	0.002	0.016	0.064	0.256
Maximum Timer Period (mSec)	0.064	0.512	4.096	16.384	65.536

Timing of Timer0 at f_{OSC} = 8MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.000125	0.001	0.008	0.032	0.128
Maximum Timer Period (mSec)	0.032	0.256	2.048	8.192	32.768

Timing of Timer1 at f_{OSC} = 1MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.001	0.008	0.064	0.256	1.024
Maximum Timer Period (Sec)	0.0655	0.5243	4.1943	16.777	67.1088

Timing of Timer1 at f_{OSC} = 2MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.0005	0.004	0.032	0.128	0.512
Maximum Timer Period (Sec)	0.0328	0.262	2.097	8.3886	33.554

Timing of Timer1 at f_{OSC} = 4MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.00025	0.002	0.016	0.064	0.256
Maximum Timer Period (Sec)	0.016	0.131	1.049	4.194	16.777

Timing of Timer1 at f_{OSC} = 8MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (mSec)	0.000125	0.001	0.008	0.032	0.128
Maximum Timer Period (mSec)	0.0082	0.0655	0.5243	2.097	8.3886

Timing of Timer1 at f_{OSC} = 12MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (uSec)	0.0833	0.666	5.333	21.333	85.333
Maximum Timer Period (mSec)	0.0055	0.0437	0.3495	1.3981	5.5924

Timing of Timer1 at f_{OSC} = 16MHZ

Prescaler	1	8	64	256	1024
Timer Resolution in (uSec)	0.0625	0.5	4	16	64
Maximum Timer Period (mSec)	0.0041	0.0328	0.262	1.0486	4.1943

التعليمات الجديدة:

التعليمة البرمجية	شرح التعليمة
Config Timer0 = **Timer** , Prescale = 1|8|64|256|1024	إعداد وتعريف المؤقت Timer0 ليعمل كمؤقت بطول 8bit. Prescaler: تعيين قيمة المقسم الترددي.
Config Timer0 = **Counter** , Prescale = 1|8|64|256|1024 , Edge = Rising / Falling , **Clear Timer** = 1|0	إعداد وتعريف المؤقت Timer0 ليعمل كمؤقت بطول 8bit. Edge: تعيين الجبهة (صاعدة أو هابطة) التي سيحصل عندها العد. Clear: تصفير محتوى المؤقت عند حدوث المقاطعة.
Start Timer0 **Stop Timer0**	تفعيل|إلغاء نبضة البدء للمؤقت Timer0 ليبدأ|يتوقف العد.
Enable Ovf0 / **Enable Timer0** **Disable** Ovf0 / **Disable Timer0**	تفعيل|إلغاء مقاطعة الطفحان للمؤقت Timer0.
On Ovf0 T0_isr	في حال تحقق مقاطعة الطفحان يقفز إلى البرنامج T0_isr
Timer0 = Value Value = **Timer0**	إسناد قيمة أولية|قراءة محتوى المؤقت Timer0.
Config Timer1 = Counter|Timer, Edge = Rising|Falling, Prescale = 1|8|64|256|1024, Noise Cancel = 0|1 , Capture Edge = Rising|Falling , Clear Timer = **1|0**, Compare A = Clear|Set|Toggle|Disconnect, Compare B = Clear|Set|Toggle|Disconnect إعداد وتعريف المؤقت Timer1 ليعمل كمؤقت|عداد بطول 16bit. Edge: تعيين الجبهة (صاعدة|هابطة) التي سيحصل عندها العد. Prescaler: تعيين قيمة المقسم الترددي. Noise Cancel: تفعيل|إلغاء نمط رفض الضجيج للمؤقت Timer1. Capture Edge: تعيين الجبهة (صاعدة أو هابطة) التي سيحصل عندها حادثة المسك للمؤقت Timer1. Clear Timer: تصفير|عدم تصفير محتوى المؤقت Timer1 عند حدوث مقاطعة نظير المقارنة. Compare A: تعيين سلوك خرج نظير المقارنة (OC1A) عند تحقق المقارنة بين محتوى مسجل النظير (OCR1A) وبين قيمة مسجل المؤقت Timer1 (TCNT1)، فإما أن يقوم بتطبيق "1" أو "0" أو "Inv" على القطب OC1A أو يقوم بفصل القطب. Compare B: تعيين سلوك خرج نظير المقارنة (OC1B) عند تحقق المقارنة بين محتوى مسجل النظير (OCR1B) وبين قيمة مسجل المؤقت Timer1 (TCNT1)، فإما أن يقوم بتطبيق "1" أو "0" أو "Inv" على القطب OC1B أو يقوم بفصل القطب.	
Stop Timer1 **Start Timer1**	تفعيل|إلغاء نبضة البدء للمؤقت Timer1 ليبدأ|يتوقف العد.
Enable Ovf1 / **Enable Timer0** **Disable** Ovf1 / **Disable Timer0**	تفعيل|إلغاء مقاطعة الطفحان للمؤقت Timer1.
On Ovf1 T1_isr	في حال تحقق مقاطعة الطفحان يقفز إلى البرنامج T1_isr
Enable Capture1 **Disable** Capture1	تفعيل|إلغاء مقاطعة حادثة المسك للمؤقت Timer1.
On Capture1 Cpa1_isr	في حال تحقق مقاطعة حادثة المسك يقفز إلى البرنامج Cap1_isr
Enable Compare1a **Enable Compare1b**	تفعيل|إلغاء مقاطعة نظير المقارنة للمؤقت Timer1.
On Compare1a Oc1a **On Compare1b** Oc1b	في حال تحقق مقاطعة نظير المقارنة يقفز إلى البرنامج OC1a/b
Compare1a = Value **Compare1b** = Value	إسناد قيمة أولية لمسجل نظير المقارنة للمؤقت Timer1.
Value = **Compare1a** Value = **Compare1b**	قراءة محتوى مسجل نظير المقارنة المؤقت Timer1.

أنماط عمل المؤقتات في عائلة المتحكمات AVR

أولاً: نمط العمل العام (Normal Mode):

في هذه الحالة سيعمل المؤقت عند نفس تردد عمل المعالج، وبافتراض أن تردد المعالج هو 1MHZ فإن دقة المؤقت هي 1uS، وهذا يعني أنه من أجل كل نبضة على مدخل الـ Clock للمؤقت سينقضي زمن قدره 1uSec.

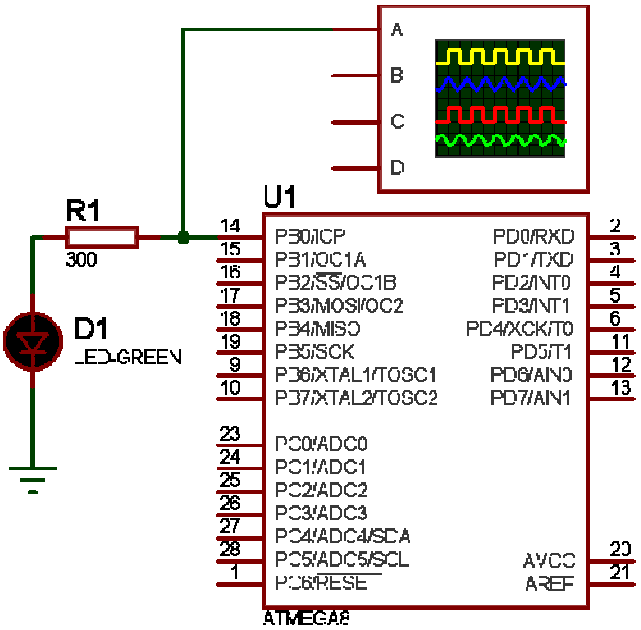
بافتراض أنه يراد انقضاء زمن تأخير قدره 50mSec، فإنه يمكن حساب عدد النبضات على مدخل التوقيت للمؤقت من أجل الزمن المذكور بالعلاقة التالية:

$$TimeCount_{Target} = \frac{\frac{1}{T_{Elapsed}}}{\frac{1}{Timer_{CLOCK}}} = \frac{\frac{1}{50\times10^{-3}}}{\frac{1}{1\times10^{6}}} = \frac{0.05}{10^{-6}} = 50000\ Pulses$$

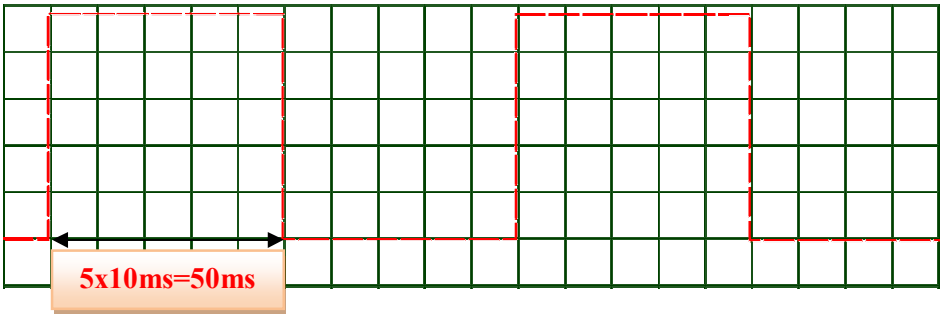
كما هو ملاحظ أن هذا العدد هو أكبر بكثير من القيمة الأعظمية التي يمكن أن يعدها المؤقت Timer0 والتي تساوي 255، لذلك يمكن أن نستخدم المؤقت Timer1.

تطبيق: سوف نستخدم زمن التأخير المسحوب أعلاه من أجل توليد قطار نبضات بفواصل زمنية 50mS.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'-----------------------
Config Timer1 = Timer , Prescale = 1
Start Timer1

Config Portb.0 = Output
Led Alias Portb.0
'-----------------------
Do
   If Timer1 >= 50000 Then
      Stop Timer1
         Toggle Led
         Timer1 = 0
      Start Timer1
   End If
Loop
End
```



سنحصل على راسم الإشارة على الشكل التالي:



ثانياً: نمط طفحان المؤقت (Overflow):

يمكن تمثيل المؤقت على أنه خزان "مياه" يتسع لكم معين من "المياه" متعلق بأبعاد هذا الخزان وسعته الحجمية (f_{OSC}, Prescaler, N)، وسوف يتم تفريغه بشكل آني كلما تم تعبئته بشكل كامل. وبالتالي فإنه عند تعبئة هذا الخزان بالمياه ووصول المياه إلى قمة الخزان، فإن الخزان سيصل إلى حالة تسمى الطفحان (القيمة الأعظمية للمؤقت)، عندها سيتم تفعيل علم الطفحان "OVF1 or OVF0"[1][2] الموجود في مسجل علم مقاطعة المؤقت "TIFR"، وأثناء البرنامج الرئيسي سيقوم المراقب (CPU) بمراقبة حالة هذا العلم، وعند تحقق الشرط (تصبح حالة العلم "1") أي: **Tifr.x[3] = 1**، يقوم بتنفيذ البرنامج المتعلق.

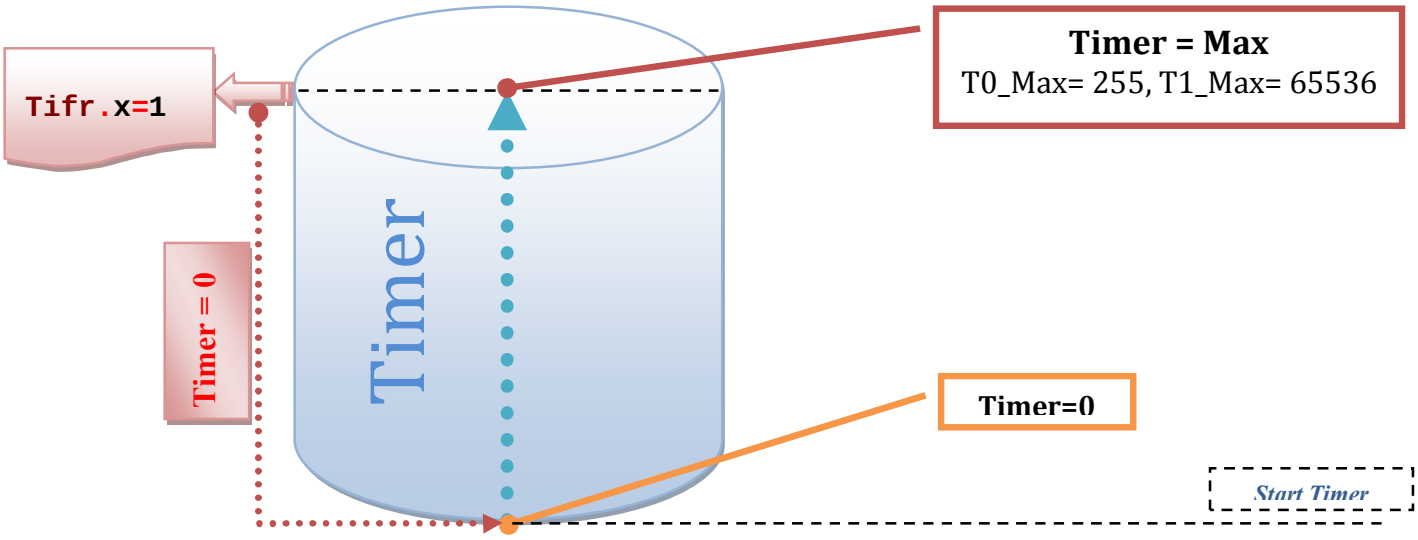
ملاحظة: في نمط الصفحان بدون تفعيل مقاطعة الطفحان، فإنه يجب تصفير علم الطفحان بكتابة القيمة "1" عليه [**Tifr.x = 1**]، أمّا عند تفعيل مقاطعة الطفحان فإن هذا العلم يتم تصفيره بشكل آني.

$$MaxVal = 2^{RES} - 1$$

إن الزمن الذي سوف يستهلكه الخزان لكي يصل إلى حالة الطفحان هو نفسه الزمن الأعظمي للمؤقت (*Maximum Timer Period*)، وبالتالي فإن هذا الزمن متعلق بشكل مباشر بما يلي:

$$T = 2^N \frac{Prescaler}{fosc}$$

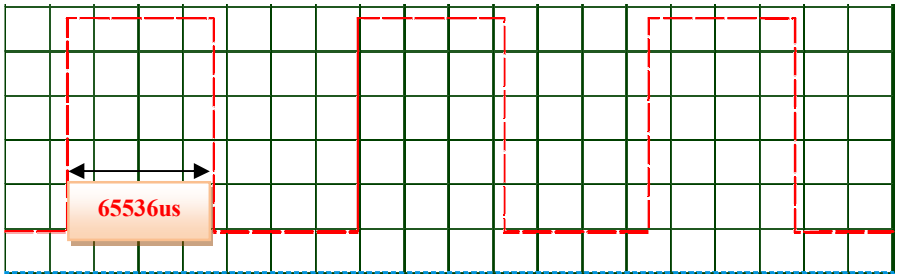
- دقة المؤقت الذي تم اختياره (N=8, 16).
- تردد الهزاز الكريستالي للمعالج (f_{OSC}).
- قيمة المقسم الترددي (Prescaler).



[1] OVF1: Timer/Counter1 Overflow Flag located in TIFR/bit 2, to check or reset this bit: TIFR.2=1
[2] OVF0: Timer/Counter0 Overflow Flag located in TIFR/bit 0, to check or reset this bit: TIFR.0=1
[3] **X = 0** for Timer0, **X = 2** for Timer1

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'-----------------------
Config Timer1 = Timer,
Prescale = 1
Start Timer1

Config Portb.0 = Output
Led Alias Portb.0
'-----------------------
Do
   If Tifr.2 = 1 Then
      Tifr.2 = 1
      Stop Timer1
         Toggle Led
      Start Timer1
   End If
Loop
End
```

تطبيق: البرنامج جانباً يقوم بفحص علم الطفحان الذي سيفعّل عندما تصل قيمة مسجل المؤقت إلى القيمة الأعظمية، وبالتالي سوف يتم توليد قطار نبضات بفواصل زمنية 65536µSec وذلك لأن القيمة الأعظمية للمؤقت Timer1 هي 65536 والمقسم الترددي يساوي الواحد وبالتالي تردد عمل المؤقت يساوي تردد عمل المعالج (1MHZ).



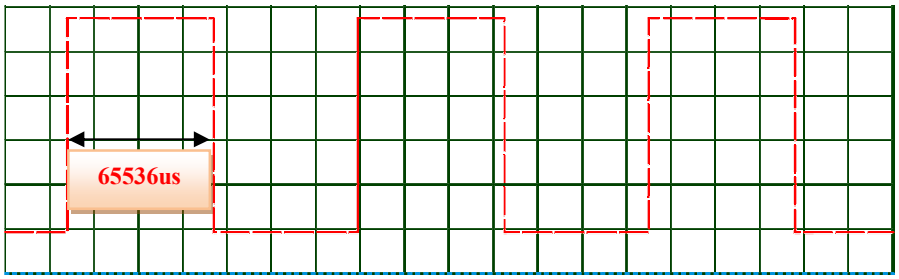
ثالثاً: نمط مقاطعة طفحان المؤقت (Overflow INT):

إن هذا النمط مشابه تماماً لنمط طفحان المؤقت السابق إلا أنه يعتمد على مقاطعة الطفحان (Hardware) لمعرفة إذا ما حصل الطفحان، وبالتالي لن ينشغل المعالج بتفحص خانة الطفحان في مسجل TIFR.

في هذه الحالة سيتم تصفير خانة علم الطفحان بشكل آلي كلما تحققت المقاطعة.

```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Prescale = 1
Enable Ovf1
On Ovf1 T1_isr
Enable Interrupts

Config Portb.0 = Output
Led Alias Portb.0

Dim Flag As Bit
Start Timer1
'-----------------------
Do
   If Flag = 1 Then
     Reset Flag
     Toggle Led
     Start Timer1
   End If
Loop
End
'-----------------------
T1_isr:
   Stop Timer1 : Set Flag
Return
```

تطبيق: البرنامج جانباً يعتمد على مقاطعة الطفحان التي سوف تحدث عندما تصل قيمة مسجل المؤقت إلى القيمة الأعظمية، وبالتالي سوف يتم توليد قطار نبضات بفواصل زمنية 65536µSec وذلك لأن القيمة الأعظمية للمؤقت Timer1 هي 65536 والمقسم الترددي يساوي الواحد وبالتالي تردد عمل المؤقت يساوي تردد عمل المعالج (1MHZ).



رابعاً: نمط مسجل نظير المقارنة (CTC_{OCR} Mode):

يملك المؤقت Timer0 مسجل نظير مقارنة وحيد OCR0، بينما يملك المؤقت Timer1 مسجلي نظير مقارنة OCR1A, OCR1B وكذلك يملك المؤقت Timer2 مسجل نظير مقارنة وحيد OCR2.

في هذا النمط تستخدم مسجلات نظير المقارنة من أجل تخزين القيمة التي سيتم مقارنة محتويات المؤقت (TCNTn) في كل لحظة معها، وعند تحقق نتيجة المقارنة (TCNT=OCRnx) سوف يفعّل علم نظير المقارنة في مسجل الحالة للمؤقت (OCFnx).

من أجل تفعيل هذا النمط يجب كتابة "1" إلى البت الرابع (WGM12=1) في مسجل TCCR1B.

يتم حساب قيمة الشحن (المكافئة للزمن المطلوب) لمسجل نظير المقارنة بالعلاقة التالية:

$$OCRnx_{VALUE} = \frac{fosc \times T_{required}}{Prescaler}$$

حيث أن:

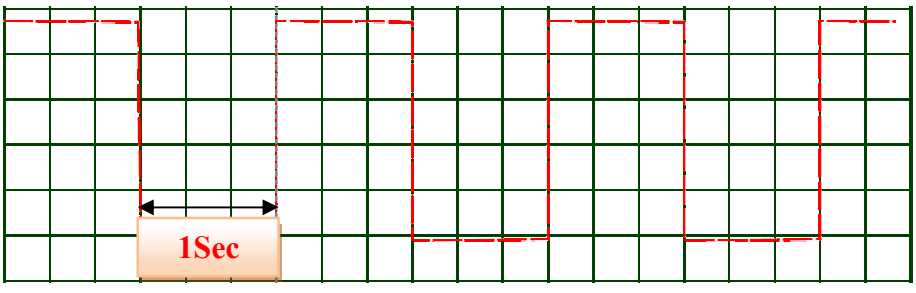
$OCRnx$: هي القيمة التي سيتم شحنها لمسجل النظير (OCR1AL, OCR1AH).

$T_{required}$: الزمن المطلوب والذي يكافئ قيمة الشحن المحسوبة.

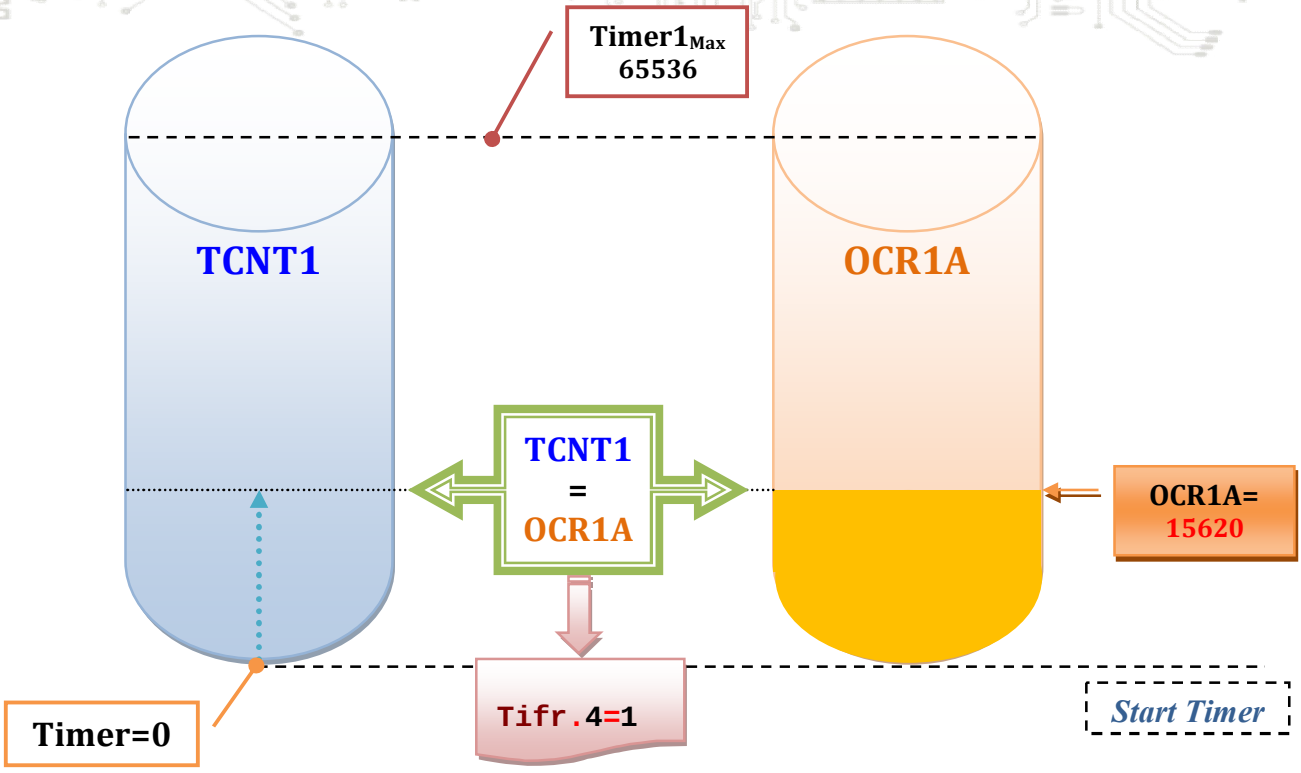
تطبيق: المطلوب حساب قيمة الشحن للمؤقت Timer1 من أجل توليد زمن قدره 1Sec علماً أن تردد عمل المعالج هو 1MHZ والمقسم الترددي 64.

$$OCR1A_{VALUE} = \frac{1MHZ \times 1Sec}{64} = 15620$$

البرنامج جانباً يعتمد على مسجل نظير المقارنة الذي سوف يتم شحنه بالقيمة 15620 من أجل توليد قطار نبضات بفواصل زمنية 1Sec




```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Prescale = 64
Compare1a = 15625
Set Tccr1b.3

Config Portb.0 = Output
Led Alias Portb.0

Start Timer1
'-----------------------
Do
   If Tifr.4 = 1 Then
      Stop Timer1
      Timer1 = 0
      Toggle Led
      Tifr.4 = 1
      Start Timer1
   End If
Loop
End
'-----------------------
```



خامساً: نمط مقاطعة نظير المقارنة (CTC Mode using Interrupt):

إن هذا النمط مشابه تماماً لنمط نمط مسجل نظير المقارنة السابق إلا أنه يعتمد على مقاطعة مسجل النظير عند حصول المساواة، وبالتالي لن ينشغل المعالج بتفحص خانة نظير المساواة (**Tifr.4**) بشكل دائم.

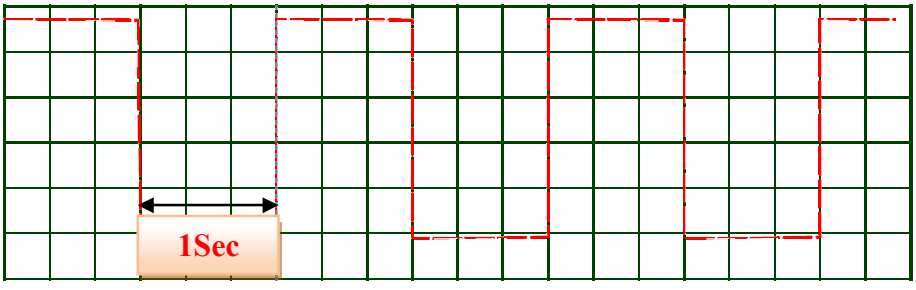
كما أنه عند تفعيل علم مقاطعة نظير المقارنة (OCIEnx) سيتم عندها تصفير خانة نظير المقارنة (**Tifr.4**) بشكل آلي كلما تحققت المقارنة (OCF1A).

تطبيق: المطلوب حساب قيمة الشحن للمؤقت Timer1 من أجل توليد زمن قدره 1Sec علماً أن تردد عمل المعالج هو 1MHZ والمقسم الترددي 64.

$$OCR1A_{VALUE} = \frac{1MHZ \times 1Sec}{64} = 15620$$

البرنامج جانباً يعتمد على مقاطعة مسجل نظير المقارنة الذي سوف يتم شحنه بالقيمة 15620 وسيحدث مقاطعة نظير المقارنة كل 1Sec من أجل توليد قطار نبضات بفواصل زمنية 1Sec.

```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Prescale=64,Clear Timer=1
On Compare1a Timer1_isr
Compare1a = 15625
Enable Compare1a

Config Portb.0 = Output
Led Alias Portb.0

Dim Flag As Bit
Start Timer1
Enable Interrupts
'-----------------------
Do
   If Flag = 1 Then
      Reset Flag
      Toggle Led
      Start Timer1
   End If
Loop
'-----------------------
Timer1_isr:
   Stop Timer1 : Set Flag
Return
```



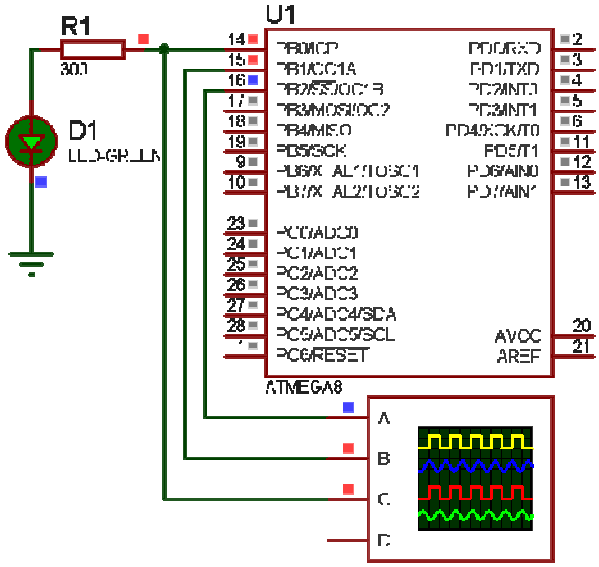
إن البرنامج أعلاه يتضمن التعامل مع المسجل OCR1A فقط؛ يمكن التعامل من المسجل OCR1B أيضاً بنفس الطريقة، البرنامج التالي يعطي مثالاً عن استخدام كلا مسجلي المقارنة من أجل توليد مقاطعات بأزمنة مختلفة.

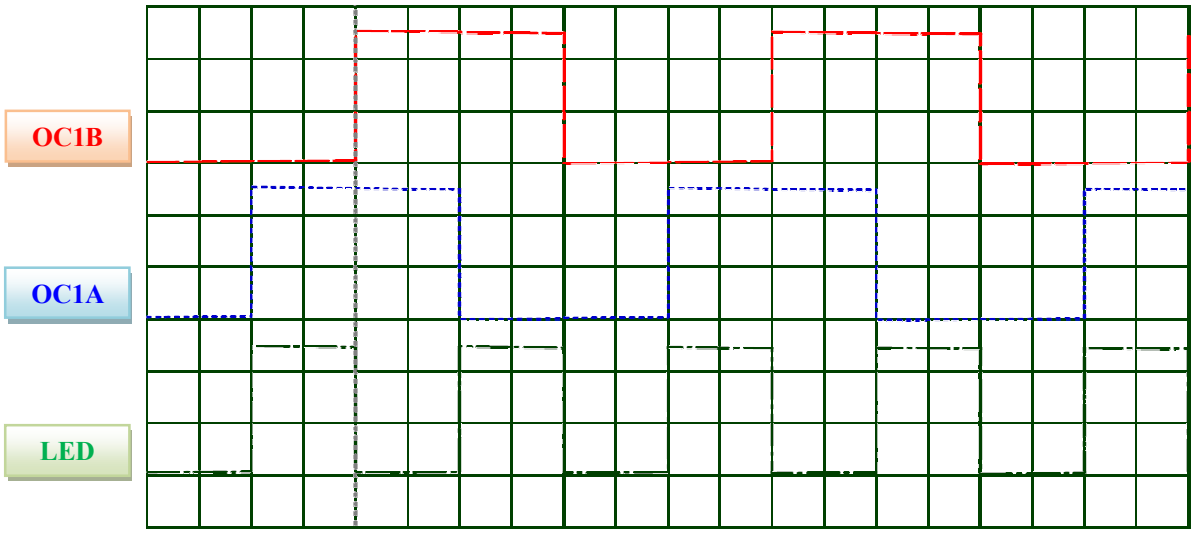
تطبيق: المطلوب حساب قيمة الشحن للمؤقت Timer1 من أجل توليد مقاطعة كل 1Sec على الخرج OC1A ومقاطعة أخرى كل 2Sec على الخرج OC1B وكذلك الخرج على القطب Pinb.0 سوف يتغير عند كل مقاطعة علماً أن تردد عمل المعالج هو 1MHZ والمقسم الترددي 64.

$$OCR1A_{VALUE} = \frac{1MHZ \times 1Sec}{64} = 15620$$

$$OCR1B_{VALUE} = \frac{1MHZ \times 2Sec}{64} = 31250$$

```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Compare A= Toggle, Compare B= Toggle
, Prescale = 64
Compare1a = 15625
Compare1b = 31250
On Compare1a Oc1a
On Compare1b Oc1b
Enable Compare1a
Enable Compare1b

Config Portb.0 = Output
Led Alias Portb.0

Dim Flag1 As Bit , Flag2 As Bit
Start Timer1
Enable Interrupts
'-----------------------
Do
   If Flag1 = 1 Then
      Reset Flag1 : Toggle Led
   End If
   If Flag2 = 1 Then
      Reset Flag2 : Toggle Led
      Timer1 = 0 : Start Timer1
   End If
Loop
End
'-----------------------
Oc1a:
   Set Flag1
Return
'-----------------------
Oc1b:
   Set Flag2 : Stop Timer1
Return
```





سادساً: نمط مقاطعة الطفحان من أجل قيمة شحن أولية (CTC Overflow Mode):

إن هذا النمط يعمل وفق مبدأ طفحان المؤقت وبشكل مشابه لمبدأ مقارن النظير، حيث أنه يتم حساب قيمة الشحن اللازمة لتحقيق زمن معين (مطلوب تحقيقه) ومن ثم يتم طرح هذه القيمة الناتجة من القيمة الأعظمية للمؤقت ($Timerx_{MAX}$).

إن المؤقت سوف يبدأ الزيادة من ناتج الطرح بين القيمة الأعظمية والقيمة المسحوبة للزمن المطلوب، بمعنى آخر إن المؤقت بدلاً من أن يبدأ العد من الصفر إلى قيمة المقارنة كما في نمط "نظير المقارنة"، فإنه سوف يبدأ من قيمة معينة إلى أن يصل إلى القيمة العظمى ويحصل الطفحان وتتولد مقاطعة الطفحان.

يتم حساب قيمة الشحن اللازمة لتوليد زمن مقاطعة طفحان محدد (التي سيبدأ المؤقت منها إلى أن يصل إلى القيمة الأعظمية) بالعلاقة التالية:

$$TCNT_{OC\_Val} = 2^N - \frac{fosc \times T_{required}}{Prescaler}$$

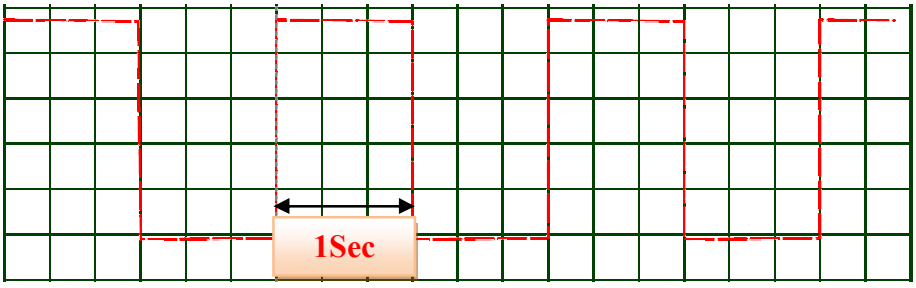
حيث أن: $T_{required}$ قيمة الزمن المطلوب تحقيق المقاطعة من أجله.

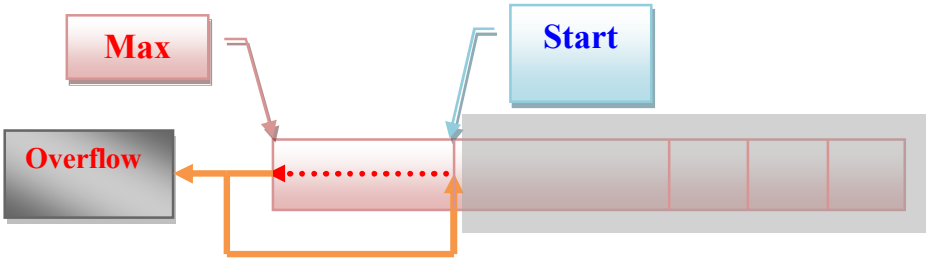
أو يمكن إيجاد الزمن انطلاقاً من القيمة الموجودة في مسجل المؤقت:

$$T_{required} = \frac{Prescaler}{fosc}\left(2^N - TCNT_{OC\_Val}\right)$$

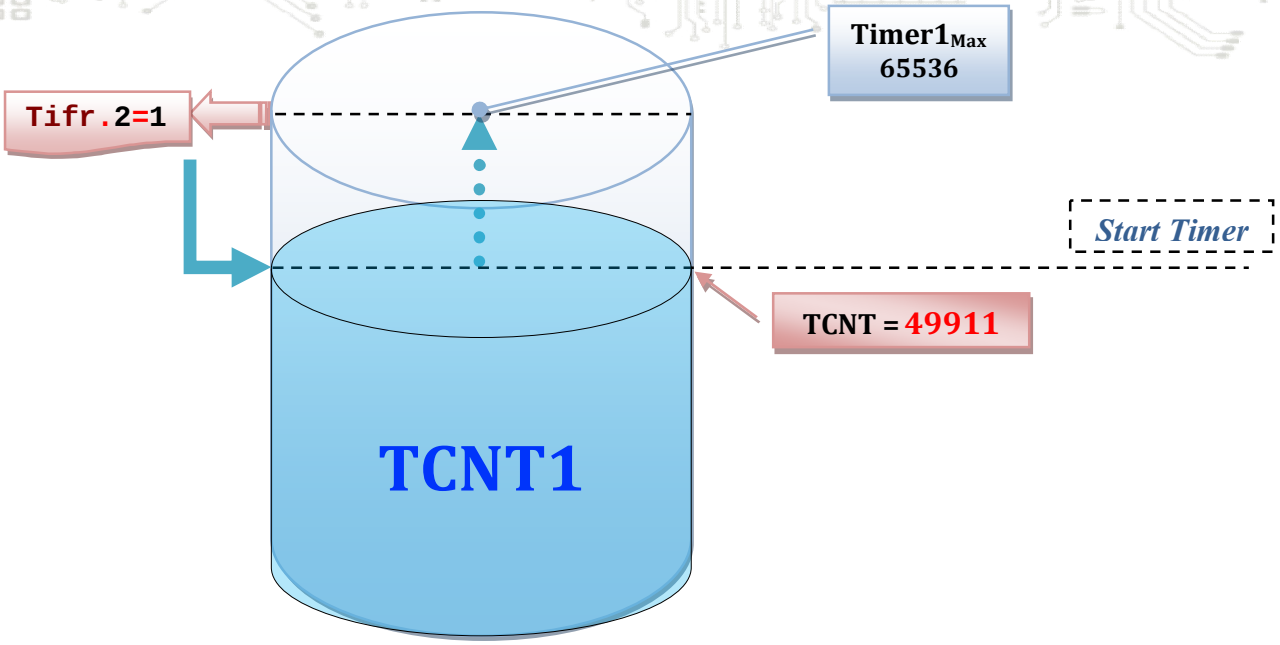
تطبيق: المطلوب حساب قيمة الشحن للمؤقت Timer1 من أجل توليد مقاطعة كل 1Sec علماً أن تردد عمل المعالج هو 1MHZ والمقسم الترددي 64.

$$TCNT_{OC_{Val}} = 65536 - \frac{1MHZ \times 1Sec}{64} = 49911$$






```
$regfile = "m8def.dat"
$crystal = 1000000
'-----------------------
Config Timer1 = Timer ,
Prescale=64
On Timer1 Timer1_isr
Timer1 = 49911
Enable Timer1

Config Portb.0 = Output
Led Alias Portb.0

Dim Flag As Bit
Start Timer1
Enable Interrupts
'-----------------------
Do
   If Flag = 1 Then
      Reset Flag
      Toggle Led
      Start Timer1
   End If
Loop
End
'-----------------------
Timer1_isr:
   Stop Timer1
   Set Flag
   Timer1 = 49911
Return
```

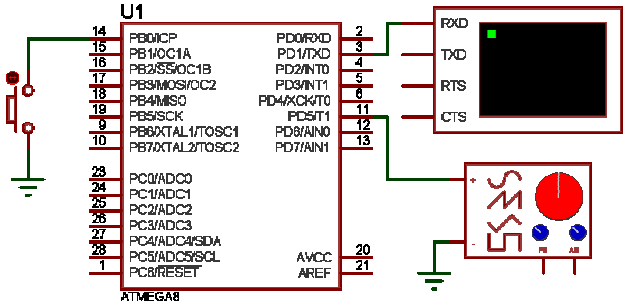


سابعاً: نمط حادثة المسك (Capture Mode):

يملك المؤقت Timer1 ميزة تتعلق بشك فيزيائي بأحد أقطاب المتحكم الخارجية والذي يسمى ICP، حيث إنه عند تطبيق نبضة على هذا القطب (جبهة صاعدة أو هابطة) يتم أخذ صورة (Capture) من محتوى مسجل المؤقت Timer1 (TCCR1) ووضعها في مسجل حادثة المسك Capture1 (ICR1).

تطبيق: يعمل المؤقت1 كعداد من 0 إلى القيمة Max وفي أي لحظة يتم الضغط على المفتاح سوف يتم تخزين القيمة الحالية في مسجل حادثة المسك.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer1 = Timer, Capture Edge = Rising
, Noise Cancel = 1 , Prescale = 1024
Start Timer1

Config Pinb.0 = Input
Portb.0 = 1
'----------------------
Do
   Print "Timer: " ; Timer1
   Print "Icr1l: " ; Icr1l
   Print "Icr1H: " ; Icr1h
   Print "Capture: " ; Capture1
   Print "--------------"
Loop
End
```

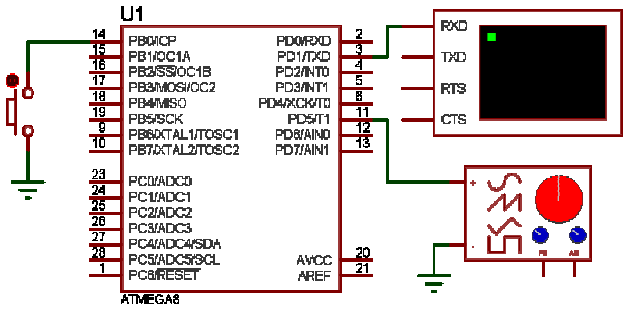


ثامناً: نمط مقاطعة حادثة المسك (Capture Mode Interrupt):

في هذه الحالة وعند تطبيق نبضة على القطب ICP يتم أخذ صورة (نسخة) من محتوى مسجل المؤقت Timer1 (TCCR1) ووضعها في مسجل حادثة المسك Capture1 (ICR1) وتوليد مقاطعة حادثة المسك وسوف يقفز إلى برنامج المقاطعة.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer1 = Timer , Capture Edge =
Rising , Noise Cancel = 1 , Prescale = 1024
Enable Interrupts
Enable Capture1
Start Timer1
On Icp1 Timer1_cpa_isr

Config Pinb.0 = Input
Portb.0 = 1
'----------------------
Do
   nop
Loop
End
'----------------------
Timer1_cpa_isr:
   Stop Timer1
   Disable Capture1

   Print "Timer: " ; Timer1
   Print "Captr: " ; Capture1

   Capture1 = 0 : Timer1 = 0
   Enable Capture1
   Start Timer1
Return
```

تطبيق: يعمل المؤقت1 كعداد من 0 إلى القيمة Max وفي أي لحظة يتم الضغط على المفتاح سوف يتم تخزين القيمة الحالية في مسجل حادثة المسك، ويقفز مؤشر البرنامج إلى برنامج خدمة مقاطعة حادثة المسك، وخلال هذا يجب إلغاء تفعيل مقاطعة حادثة المسك وإيقاف المؤقت حتى الانتهاء من معالجة برنامج خدمة المقاطعة.



العدادات (Counters):

في جميع الأنماط أعلاه تم استخدام المؤقت كوحدة منطقية موجودة في داخل المتحكم ولا اتصال لها بالعالم الخارجي (فيزيائياً)، رغم كون وجود أقطاب للمتحكم تدعى T0, T1 (أقطاب دخل للمؤقتات)، إلا أنها تستخدم كمداخل نبضات لدارة العداد.

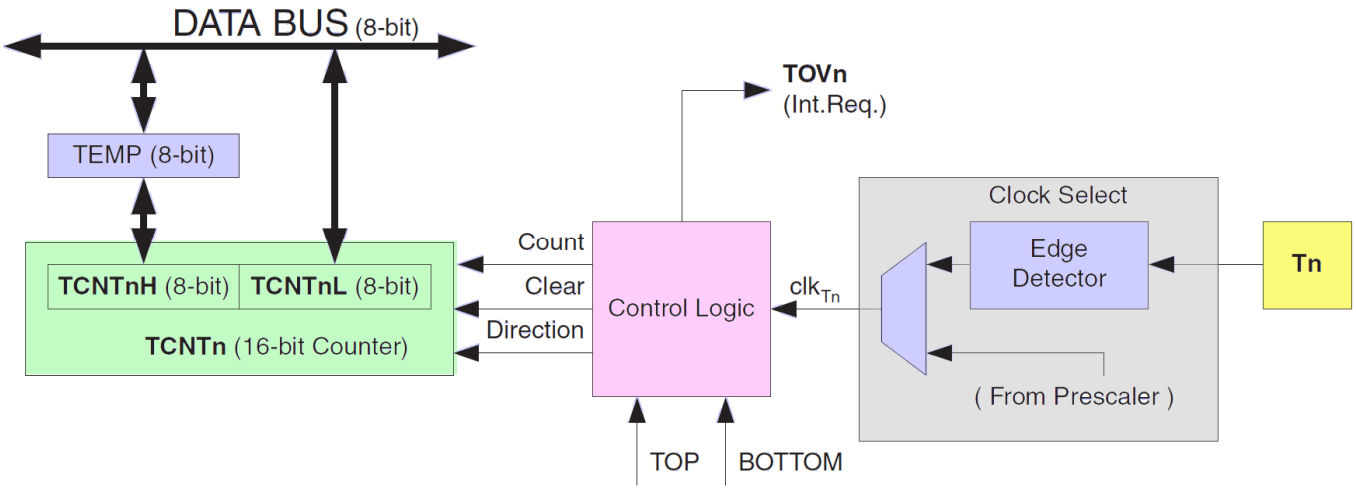
يستخدم نمط العداد من أجل عد نبضات أو أحداث خارجية مطبقة على الأقطاب T0, T1، هذه الأحداث سوف تقوم بقدح دارة المؤقت الداخلي (بدلاً من أن يتم قدحه من دارة الهزاز) ويزداد المؤقت عند كل جبهة واردة (صاعدة أو هابطة أو عند كلاهما)، وعدد النبضات الواردة سوف يكون مخزن في مسجل المؤقت (TCNTx).

إن القطب T0 هو مدخل عداد لدارة المؤقت Timer0 وبالتالي فإن العدد الأعظمي للنبضات على المطبقة على القطب قبل أن تحصل مقاطعة الطفحان هو 255 نبضة أو حدث خارجي.

كما أن القطب T1 هو مدخل عداد لدارة المؤقت Timer1 وبالتالي فإن العدد الأعظمي للنبضات على المطبقة على القطب قبل أن تحصل مقاطعة الطفحان هو 65535 نبضة أو حدث خارجي.

أحد الاستخدامات الهامة للعدادات هو قياس ترددات إشارات خارجية.

ملاحظة هامة: يجب أن لا يكون تردد الإشارة المطبقة على مدخل العداد أكبر من نصف تردد العمل للمعالج وإلا فإن المؤقت لن يستطيع عدد النبضات لأنه عند كل نبضة يقوم المتحكم بتحديد مستوى العينات!

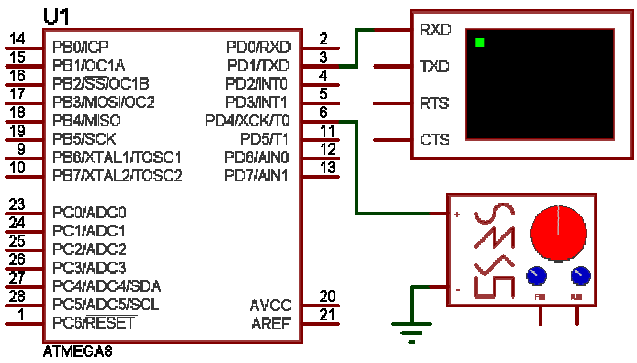


إن أنماط عمل العدادات مشابهة قليلاً إلى أنماط عمال المؤقتات، الأمثلة التالية توضح هذه الأنماط.

العداد Counter1 في النمط العام (Conter0 Normal Mode):

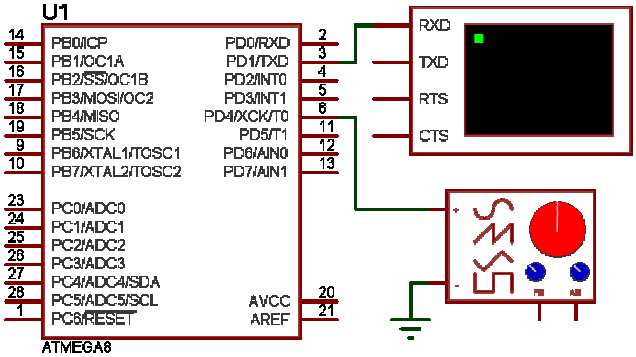
تطبيق(1): البرنامج التالي سيقوم بقراءة النبضات المطبقة على القطب T0 وطباعتها على النافذة التسلسلية، وحالما تصبح قيمة النبضات في مسجل العداد مساوية للقيمة الأعظمية (255) سوف يتوقف البرنامج.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer0 = Counter , Edge = Falling
Start Timer0
Tcnt0 = 0
'----------------------
Do
   nop
   Print Counter0
Loop Until Tcnt0 >= 255
End
```



تطبيق(2): البرنامج التالي سيقوم بقراءة النبضات المطبقة على القطب T0 خلال دور 1Sec وطباعتها على النافذة التسلسلية وبالتالي فإن القيمة الناتجة هي التردد الحقيقي المطبق، مع العلم أن التردد الأعظمي الذي يمكن قياسه في هذا البرنامج هو 255HZ.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer0 = Counter , Edge = Falling
, Prescale = 1
Stop Timer0
Config Pind.4 = Input
'----------------------
Do
   Counter0 = 0
      Start Counter0
         Waitms 1000
      Stop Counter0
  Print "Counter0: " ; Counter0 ; "HZ"
Loop
```



العداد Counter1 في نمط مقاطعة الطفحان (Overflow Counter0 Interrupt):

تطبيق(3): البرنامج التالي سيقوم بقراءة النبضات المطبقة على القطب T0 خلال دور 1Sec، ولكن لن يكون هناك محدودية لتردد الإشارة المقاسة حيث أنه عندما يصل عدد النبضات الواردة على مدخل العداد إلى 255> سيتولد مقاطعة طفحان العداد وسيتم عد المقاطعات الكلية وحساب التردد الموافق.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer0 = Counter , Edge = Falling ,
On Timer0 Counter0_isr
Enable Timer0
Enable Interrupts
'----------------------
Dim Count As Word , Fereq As Long

Do
   Start Timer0
   Waitms 1000
   Stop Timer0
   Fereq = Count * 255
   Fereq = Fereq + Timer0
   Print "Fereq= " ; Fereq
   Timer0 = 0 : Count = 0
Loop
End
'----------------------
Counter0_isr:
   Incr Count
Return
```

العداد Counter1 في نمط مقاطعة الطفحان من أجل قيمة شحن أولية (Overflow Counter0 Interrupt):

تطبيق(4): بنفس المبدأ الذي ناقشناه في المؤقتات فإنه عند شحن العداد بقيمة أولية، فإنه سوف يبدأ العد بدءً من القيمة المشحونة، فإذا كانت القيمة المشحونة على سبيل المثال "100" فإن العداد بعد "155" نبضة سوف يولد مقاطعة الطفحان.

البرنامج جانباً سوف يعطي العداد Counter0 مقاطعة طفحان كلما وردت 10 نبضات.

```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer0 = Counter , Edge = Falling ,
Prescaler = 1
Timer0 = 255 - 10
Enable Timer0
Enable Interrupts
On Timer0 Counter0_isr
'----------------------
Dim Count As Byte
Do
   nop
Loop
End
'----------------------
Counter0_isr:
   Load Timer0 , 10
   Incr Count
   Print "Count=" ; Count
Return
```

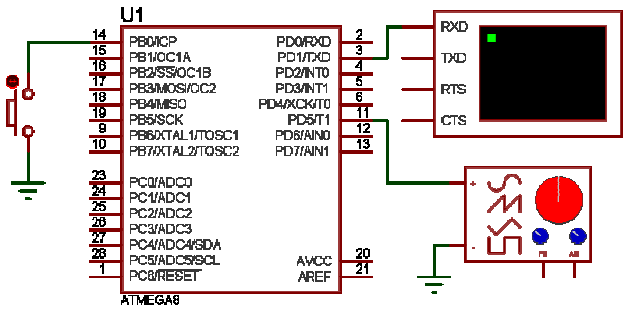
العداد Counter1 في نمط حادثة المسك (Capture Mode):

يمتلك العداد Counter1 ميزات عديدة تشابه ميزات العداد Counter0 وأخرى تفوقها بكثير ، من هذه الميزات:

- إمكانية رفض الضجيج.
- نمط حادثة المسك مع إمكانية تحديد الجبهة التي يحدث عندها المسك.
- نمط نظير مقارنة قيمة العداد.

تطبيق(5): يعمل المؤقت1 كعداد من 0 إلى القيمة Max وفي أي لحظة يتم الضغط على المفتاح سوف يتم تخزين القيمة الحالية في مسجل حادثة المسك.

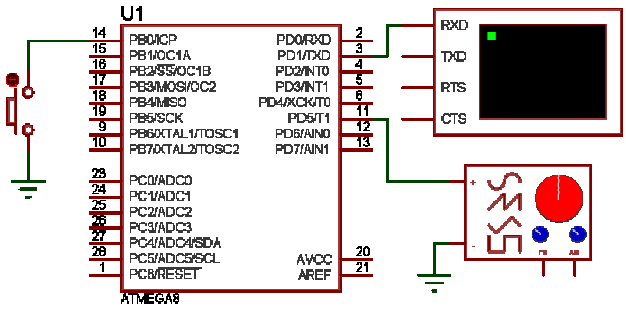
```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer1 = Counter , Prescale = 1024 ,
Edge = Rising , Capture Edge = Falling ,
Noise Cancel = 1
Start Timer1
Config Pinb.0 = Input
Portb.0 = 1
'----------------------
Do
   Print "Timer: " ; Counter1
   Print "Icr1l: " ; Icr1l
   Print "Icr1H: " ; Icr1h
   Print "Capture: " ; Capture1
Loop
End
```



العداد Counter1 في نمط مقاطعة حادثة المسك (Capture Mode Interrupt):

تطبيق(6): يقوم Counter1 بعد النبضات الواردة على القطب T1 وفي أي لحظة يتم فيها الضغط على المفتاح سوف يتم تخزين القيمة الحالية في مسجل حادثة المسك ويقفز مؤشر البرنامج إلى برنامج خدمة مقاطعة حادثة المسك، وخلال هذا يجب إلغاء تفعيل مقاطعة حادثة المسك وإيقاف المؤقت حتى الانتهاء من معالجة برنامج خدمة المقاطعة.

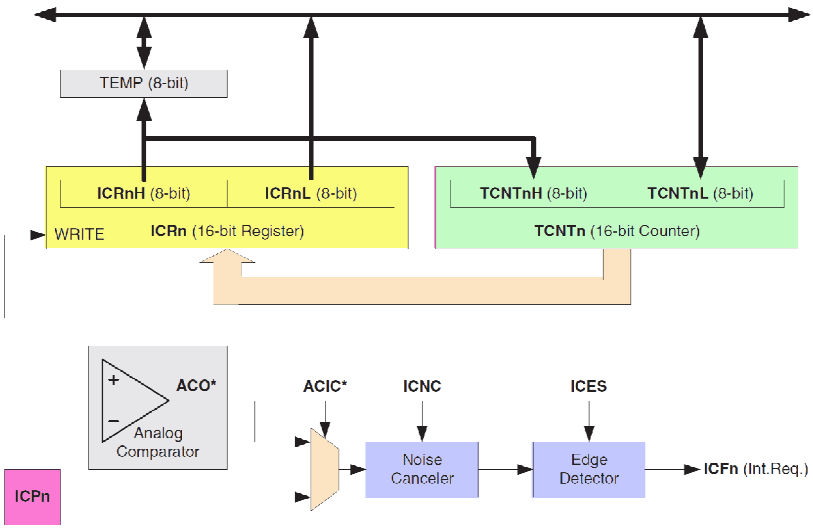
```
$regfile = "m8def.dat"
$crystal = 1000000
$baud = 4800
'----------------------
Config Timer1 = Counter , Edge = Falling ,
Capture Edge = Falling , Noise Cancel = 1 ,
Prescale = 1
Enable Timer1
Enable Capture1
On Capture1 Ctr1_cpa_isr
Enable Interrupts

Config Pinb.0 = Input
Portb.0 = 1
'----------------------
Do
   nop
Loop
End
'----------------------
Ctr1_cpa_isr:
   Stop Counter1
   Disable Capture1

   Print "Counter1: " ; Counter1
   Print "Capture1: " ; Capture1

   Enable Capture1
   Start Counter1
Return
```



العداد Counter1 في نمط نظير المقارنة (Counter1 OC Mode):

تطبيق(7): يقوم Counter1 في هذا النمط بعد نبضات الهزاز الكريستالي بعد تقسيمها عبر Prescaler وعندما تصبح قيمة العداد مساوية إلى القيمة في مسجل نظير المقارنة (Compare1a) يتم تغير حالة القطب OC1A (كل 13uS~)، والتردد الناتج هو 38Khz.

```
$regfile = "m8def.dat"
$crystal = 8000000
'----------------------
Config Timer1 = Counter , Edge = Rising ,
Prescale = 8 , Compare A = Toggle
Compare1a = 102

Config Portb.1 = Output
'----------------------
Do
   nop
Loop
```



مشروع مقياس تردد باستخدام Timer1 & Timer0:

سوف نقوم بإعداد المؤقت Timer0 في نمط مقاطعة الطفحان بحيث يتم توليد مقاطعة طفحان كل 20uS

$$T_{Overflow} = 2^N \frac{Prescaler}{fosc} = 256 \frac{1}{12.8Mhz} = 20\mu Sec$$

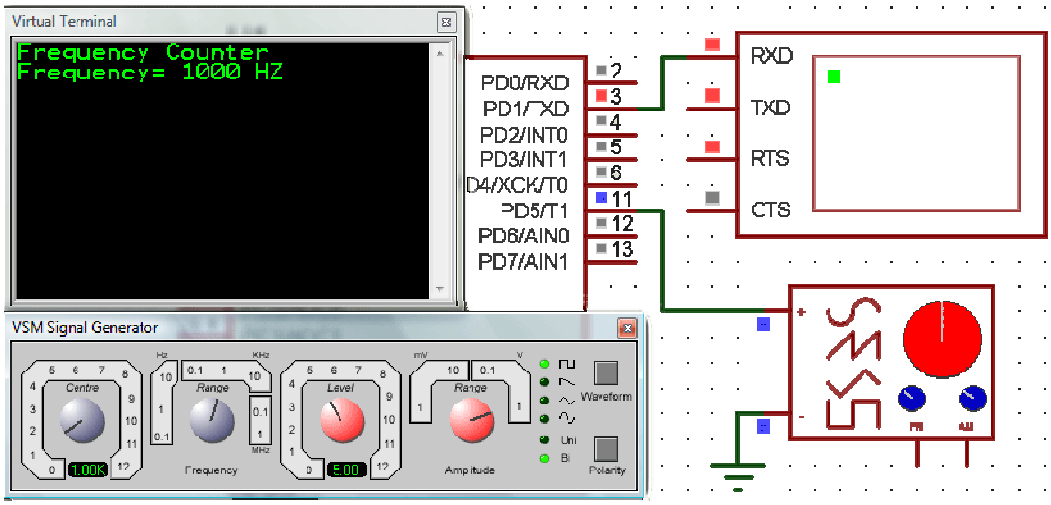
وبالتالي من أجل الحصول على زمن قياس 1Sec نحتاج إلى:

$$INT_{Number} = \frac{Total\ Time}{INT\ Time} = \frac{1Sec}{20\mu Sec} = 500000$$

أي أننا سنحتاج إلى 50000 مقاطعة Timer0 لذلك سنقوم بإعداد متحول يتم إنقاصه من 49999 = Cnt إلى Cnt=0.

سنقوم أيضاً بإعداد المؤقت Timer1 في نمط مقاطعة الطفحان ليعمل كعداد أحداث خارجية على القطب T1 (مدخل الإشارة المراد قياسها) وسنقوم بعد مقاطعات الطفحان لهذا المؤقت خلال زمن القياس (1Sec).

الناتج هو جداء مقاطعات الطفحان للمؤقت Timer1 بدقة المؤقت (65536) مضافاً إليها القيمة الأخيرة المتبقية في مسجل المؤقت ولم يصل إلى طفحان جديد بعد.




```
$regfile = "m8def.dat"
$crystal = 12800000                          ' External 12.8MHZ
$baud = 4800
'----------------------
Config Timer1 = Counter , Edge = Rising , Noise Cancel = 1
Config Timer0 = Timer , Prescale = 1                        '20uS
On Timer0 Timer0_isr
On Timer1 Timer1_isr
Stop Timer0
Stop Counter1

Config Pind.5 = Input
'----------------------
Dim Frequency As Long , Flag As Bit , Overflow As Byte , Cnt As Word
'----------------------
Enable Timer0
Enable Timer1
Enable Interrupts
```

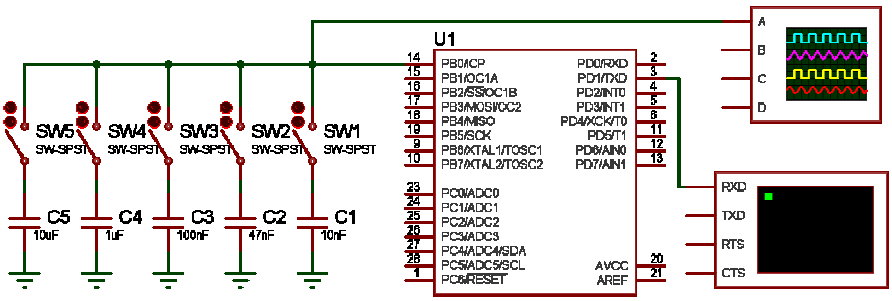
```
Print "Frequency Counter"
Main:
Counter1 = 0 : Overflow = 0 : Cnt = 49999
'----------------------
Start Timer0
Start Counter1
'----------------------
Do
   If Flag = 1 Then
      Reset Flag

      Frequency = Overflow * 65536
      Frequency = Frequency + Counter1

      Print "Frequency= " ; Frequency ; " HZ"
      Goto Main
   End If
Loop
'----------------------

Timer1_isr:
   Incr Overflow
Return
'----------------------
Timer0_isr:
   If Cnt <> 0 Then
      Decr Cnt
   Else
      Stop Counter1
      Stop Timer0
      Set Flag
   End If
Return
'----------------------
```

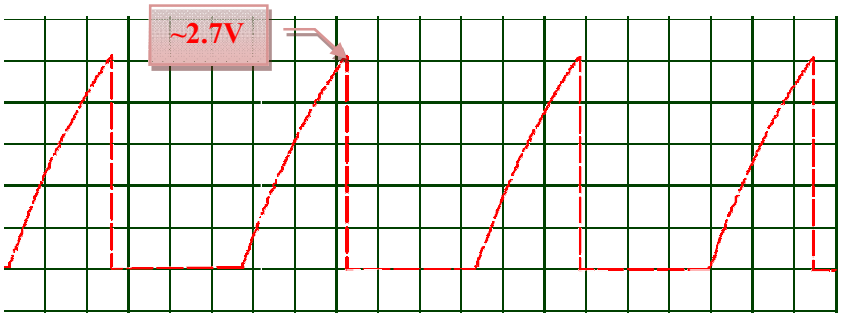
مشروع مقياس سعات (1nF~100uF):

إن المبدأ المستخدم في البرنامج التالي يعتمد على تعريف القطب الموصول معه المكثف كقطب خرج ووضع القيمة "0" على القطب من أجل تفريغ المكثف، وبعد ذلك يتم تحويل القطب إلى قطب دخل ورفع مقاومة الرفع الداخلية للقطب (50KΩ) من أجل شحن المكثف عن طريقها.

يتم قياس الزمن الذي سيستغرقه المكثف ليصل الجهد على طرفيه جهد العمل الأصغري للقطب (2.7V)، ويحسب ثابت الشحن بشكل تقريبي.

```
$regfile = "m8def.dat"
$crystal = 8000000
$baud = 4800
'----------------------
Dim T As Long , C As Single
'----------------------
Do
   T = 0
   Config Pinb.0 = Output
   Portb.0 = 0  'Low Z, 0V

   Waitms 1000

   Config Pinb.0 = Input
   Portb.0 = 1

   Do
      T = T + 1
   Loop Until Pinb.0 = 1

   C = T * 0.0866
   C = Round(c)
   Print C ; " nF"
Loop
End
```





إن الطريقة السابقة لا تعطي الدقة الكافية في القياس لأن الزمن المقاس يعتمد على حلقة شرطية، كما أن القياس سيتوقف عند جهد شحن 2.7V.

من أجل التغلب على المشاكل السابقة، يتم استخدام المؤقت كعداد لنبضات الهزاز الكريستالي ويتم استخدام المبدل التشابهي الرقمي لقياس جهد الشحن.

برمجة المتحكمات المصغرة

التجارب العملية

الجلسة التاسعة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

WALID BALID, Tuesday, December 15, 2009

التجربة السادسة والعشرون: الاتصالات التسلسلية Exp.26:Serial Communications RS232

الغاية من التجربة:

دراسة بروتوكول الاتصال التسلسلي RS232 وتطبيقاته في أنظمة التحكم الرقمي.

برتوكولات الاتصال:

تتفرع بروتوكولات الاتصال بشكل عام إلى فرعين رئيسيين:

- اتصالات تفرعية.
- اتصالات تسلسلية.

يختصر استخدم الاتصالات التفرعية من أجل نقل البيانات بسرعات عالية جداً ولمسافات قصيرة جداً، والسبب في محدودية المسافة هو تشكل السعات الطفيلية والضجيج العالي على مسارات خطوط النقل التفرعية عند ازدياد طول الناقل، كما أن حجم الناقل سيكون كبير وبالتالي فإن كلفة الناقل ستكون كبيرة أيضاً.

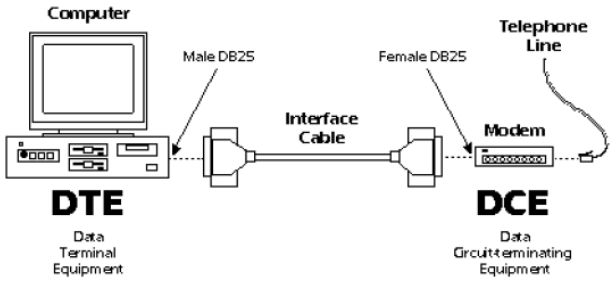
تستخدم الاتصالات التسلسلية على نطاق أوسع بكثير من الاتصالات التفرعية وتمتاز بمناعة عالية ضد الضجيج ونقل لمسافات بعيدة، كما أن حجم الناقل سيكون صغير وكلفته ضئيلة نسبياً مقارنة مع الناقل التفرعية.

Serial Communications		Parallel Communications
Asynchronous	**Synchronous**	
• Morse code telegraphy • RS-232 (COM Port) • RS-423 • RS-485 • Universal Serial Bus (USB) • FireWire • Ethernet • Fiber Channel[1] • InfiniBand[2] • MIDI[3] • DMX512[4] • Serial ATA[5] • SpaceWire[6] • PCI Express • SONET and SDH[7] • T-1, E-1[8]	o I2C o SPI o PS2	§ **LPT** § **ISA** § **EISA** § **VESA** § **ATA** § **SCSI** § **PCI** § **PCMCIA** § **IEEE-1284** § **IEEE-488**

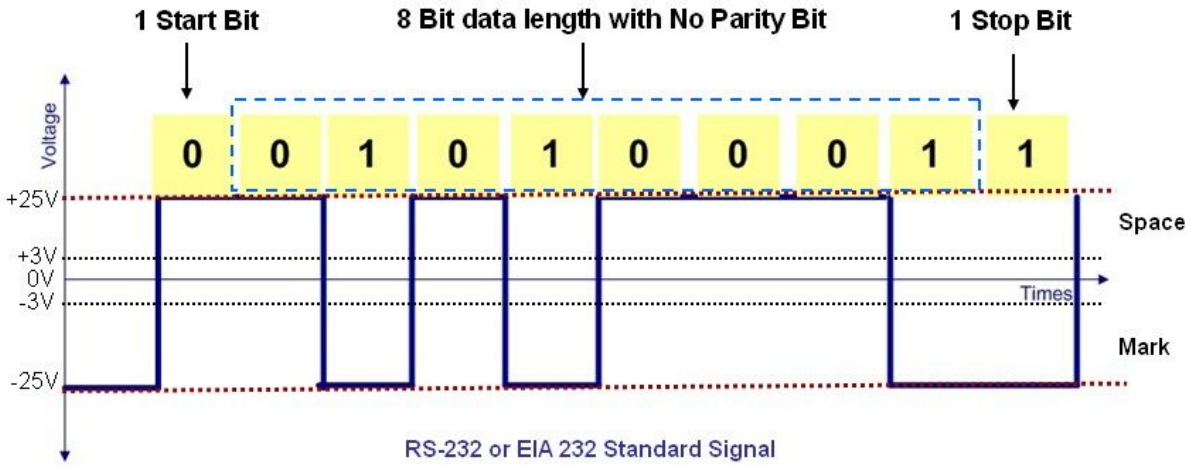
[1] High-speed, for connecting computers to mass storage devices
[2] Very high speed, broadly comparable in scope to PCI
[3] Control of electronic musical instruments
[4] Control of theatrical lighting
[5] New replacement for parallel IDE
[6] Spacecraft communication network
[7] High speed telecommunication over optical fibers
[8] High speed telecommunication over copper pairs

بروتوكول الاتصال التسلسلي RS232:

هو عبارة عن برتوكول اتصال تسلسلي غير متواقت يستخدم من أجل الربط بين طرفيتين، تسمى الأولى DTE[9] وتسمى الثانية DCE[10].



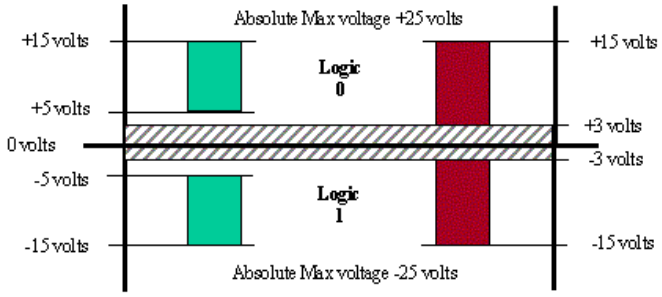
يتم إرسال كل بايت كحزمة مؤلفة من مجموعة بتات على الشكل التالي:



كما هو ملاحظ فإن المستويات المنطقية لهذا المعيار مختلفة تماماً عن المنطق TTL حيث أن:

- "0": المستوى المنطقي المنخفض ويسمى بـ "Space" ويتراوح بين +3V ~ +25V.
- "1": المستوى المنطقي العالي ويسمى بـ "Mark" ويتراوح بين -3V ~ -25V.
- "x": مستوى منطقي غير معرف ويتراوح بين -3V ~ +3V.

ملاحظة: إن جهد الدارة المفتوحة يجب أن لا يتجاوز ±25V بالنسبة للنقطة الأرضية "GND"، كما أن تيار الدارة القصيرة يجب أن لا يتجاوز 500mA.



[9] Data Terminal Equipment (computer)
[10] Data Circuit-terminating Equipment (modem)

الميزات والمساوئ لبروتوكول الاتصال RS232:

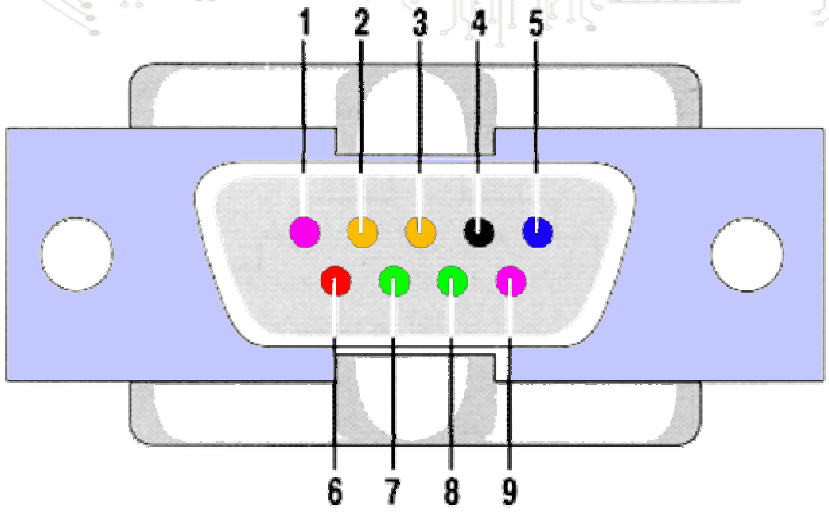
المساوئ (Disadvantages)	المحاسن (Advantages)
× مناسب فقط من أجل الربط بين System-to-System أكثر من كونه قابلاً للربط بين Chip-2-Chip أو من أجل تطبيقات Chip-2-Sensor.	ü بروتوكول اتصال شائع الاستخدام في كثير من التطبيقات ومعتمد من قبل العديد من الشركات.
× معدل نقل بيانات منخفض جداً من أجل مسافة اتصال كبيرة.	ü مسافة الاتصال طويلة نسبياً حوالي 50 قدم عند معدل إرسال منخفض، ويمكن زيادة المسافة باستخدام معدلات نقل منخفضة وتصحيح أخطاء.
× يحتاج إلى وحدة تبديل المستوى المنطقي RS232<>TTL.	ü مناعة ضد الضجيج بسبب الجهد المرتفع نسبياً (±25) للمستويات المنطقية ("1" , "0").
× مخصص للربط بين Single Master/Single Slave.	ü سهل البناء والبرمجة ومتوفر برمجياً وكعيان صلب.
× غير قابل للتوسع.	

عناوين بوابات الاتصال التسلسلي RS232 في الحاسب:

يوجد في الحاسب منافذ اتصال تسلسلي وفق المعيار RS232 وتسمى "Serial Port "COM، الجدول التالي يوضح عناوين هذه المنافذ.

Port	Address
COM1	0x3F8
COM2	0x2F8
COM3	0x3E8
COM4	0x2E8

يتوضع منفذ الاتصالات التسلسلي COMx على الوجه الخلفي للحاسب وهو من النوع DB-9Pin كما في الشكل:



يحتوي المنفذ على تسع نقاط (1, 2, 3, …., 9) وظائفها مبينة في الجدول التالي:

Pin	Name	Direction	Function	Description
1	CD	In	Control	Carrier Detect
2	RXD	In	Data	Receive Data
3	TXD	Out	Data	Transmit Data
4	DTR	Out	Control	Data Terminal Ready
5	GND	---	Ground	System Ground
6	DSR	In	Control	Data Set Ready
7	RTS	Out	Control	Request to Send
8	CTS	In	Control	Clear to Send
9	RI	In	Control	Ring Indicator

CD – Carrier Detect (Control sent from DCE to DTE):

قطب كشف حامل إشارة الرنين ويستخدم فقط في حال استخدام البوابة من أجل ربط بين حاسب وجهاز مودم.

RxD – Receive Data (Data sent from DCE to DTE):

قطب مدخل استقبال البيانات المرسلة من الطرفية الثانوية (DCE) إلى الطرفية الرئيسية (DTE).

فعال (Mark state, "0 or Positive") عند استقبال البيانات، ويعود إلى نمط البطالة (Idle State, "1 or Negative")

عند انتهاء استلام البيانات.

TxD – Transmit Data (Data sent from DTE to DCE):

قطب خرج البيانات المرسلة من الطرفية الرئيسية (DTE) إلى الطرفية الثانوية (DCE).

فعال (Mark state, "0 or Positive") خلال إرسال البيانات، ويعود إلى نمط البطالة (Idle State, "1 or Negative")

عند انتهاء إرسال البيانات.

DTR – Data Terminal Ready (Control sent from DTE to DCE):

قطب تحكم يشير إلى أن الطرفية (DTE) جاهزة للاتصال مع الطرفية الأخرى، فإذا كانت الطرفية الثانية (DCE) في نمط البطالة يقوم بإخراجها إلى النمط الفعال.

DSR – Data Set Ready (Control sent from DCE to DTE):

قطب تحكم يشير إلى أن الطرفية (DCE) في حالة اتصال مع الطرفية الرئيسية (DTE).

فعال ("0") عند وجود الاتصال، ويعود إلى نمط البطالة ("1") فور انتهاء الاتصال.

RTS – Request To Send (Control sent from DTE to DCE):

قطب تحكم يقوم بإعلام الطرفية (DCE) أن البيانات جاهزة للإرسال من الطرفية الرئيسية (DTE)، وبالتالي يمكن استخدام هذه الإشارة من أجل تفعيل دارة الاستقبال قبل إرسال أي إشارة.

فعال ("0") عندما تكون الطرفية الرئيسية جاهزة لإرسال البيانات، ويعود إلى نمط البطالة ("1") فور انتهاء إرسال البيانات.

CTS – Clear To Send (Control sent from DCE to DTE):

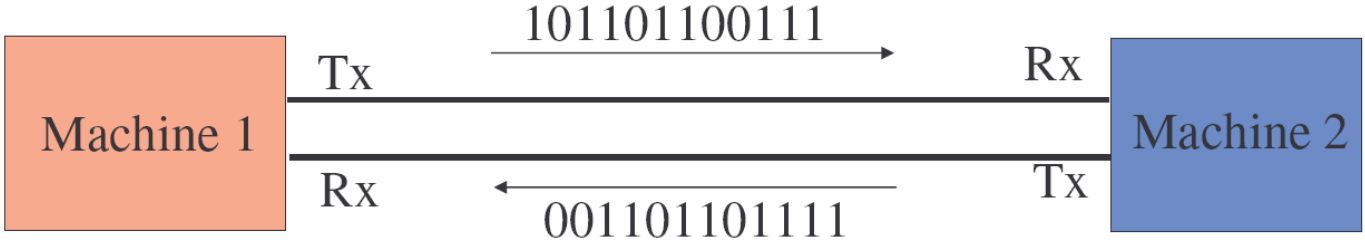
قطب تحكم يقوم بإعلام الطرفية الرئيسية (DTE) أنه استلم إشارة الإعلام بإرسال البيانات السابقة ويمكنها الآن أن تبدأ بإرسال البيانات إلى الطرفية الثانوية (DCE)، وبالتالي يمكن استخدام هذه الإشارة من أجل تفعيل دارة الاستقبال قبل إرسال أي إشارة.

فعال ("0") عندما تكون الطرفية الثانوية جاهزة لاستلام البيانات، ويعود إلى نمط البطالة ("1") فور انتهاء استلام البيانات.

RI – Ring Indicator (Control sent from DCE to DTE):

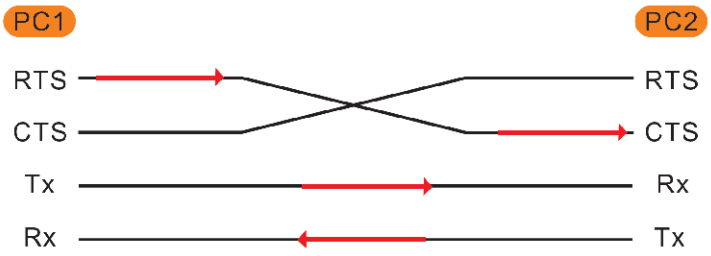
قطب تحكم يقوم بإعلام الطرفية الرئيسية (DTE) بوجود رنين من أجل فتح الخط، ويستخدم فقط في حال استخدام البوابة من أجل ربط بين حاسب وجهاز مودم.

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



أما في حال وجود مصافحة (Hardware handshaking) بين الطرفيتين فإنه يجب توصيل قطبي التحكم المتناظرين (RTS, CTS) بالإضافة لقطبي الإرسال والاستقبال (TxD, RxD)، ويتم التخاطب بين الطرفيتين:

- تقوم الطرفية الأولى بتفعيل أمر التحكم على القطب CTS من أجل إعلام الطرفية الثانية بأنها سوف ترسل بيانات.
- تقوم الطرفية الثانية بالرد على الطرفية الأولى بتفعيل القطب RTS إذا كانت جاهزة لاستقبال البيانات، وإلا يبقى القطب RTS في حالة عدم تفعيل (نمط البطالة).
- في حال كانت الطرفية الثانية مشغولة ولم ترد على طلب الطرفية الأولى فيوجد لدينا حالتين:
  - إما أن تقوم الطرفية الأولى بإعادة الطلب مرة ثانية بعد زمن محدد حتى تحصل على أذن الإرسال.
  - أو أن تقوم الطرفية الثانية بتفعيل القطب RTS فور انتهائها من العملية التي كانت تشغلها، وخلال هذا الوقت تبقى الطرفية الأولى في حالة انتظار رد الطرفية الثانية.



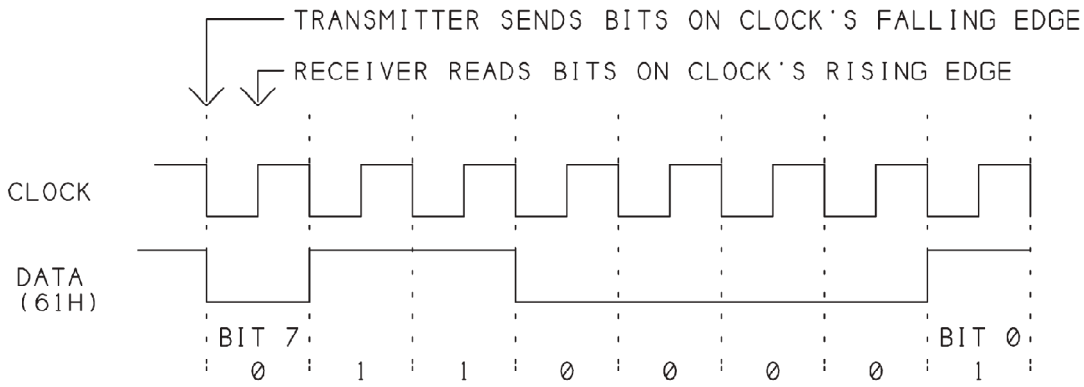
المواصفات الفنية للبروتوكول RS232:

SPECIFICATIONS		RS232	RS423
Mode of Operation		SINGLE-ENDED	SINGLE-ENDED
Total Number of Drivers and Receivers on One Line		1DRIVER/1RECVR	1DRIVER/10RECVR
Maximum Cable Length		50 FT	4000 FT
Maximum Data Rate		20kb/s	100kb/s
Maximum Driver Output Voltage		±25V	±6V
Driver Output Signal Level (Loaded Min.)	*Loaded*	±5V to ±15V	±3.6V
Driver Output Signal Level (Unloaded Max)	*Unloaded*	±25V	±6V
Driver Load Impedance (Ohms)		3k to 7k	>=450
Max. Driver Current in High Z State	*Power On*	N/A	N/A
Max. Driver Current in High Z State	*Power Off*	±6mA @ ±2v	±100uA
Slew Rate (Max.)		30V/µS	Adjustable
Receiver Input Voltage Range		±15V	±12V
Receiver Input Sensitivity		±3V	±200mV
Receiver Input Resistance (Ohms)		3k to 7k	4k min

إن المواصفات القياسية لبروتوكولات الاتصال المذكورة أعلاه توصي باستخدام كابل مزدوج مجدول 24AWG ويحوي على Shield محيط بالعازل الداخلي، وذو سعة نقل 16PF/FT وممانعة مميزة 100Ω.

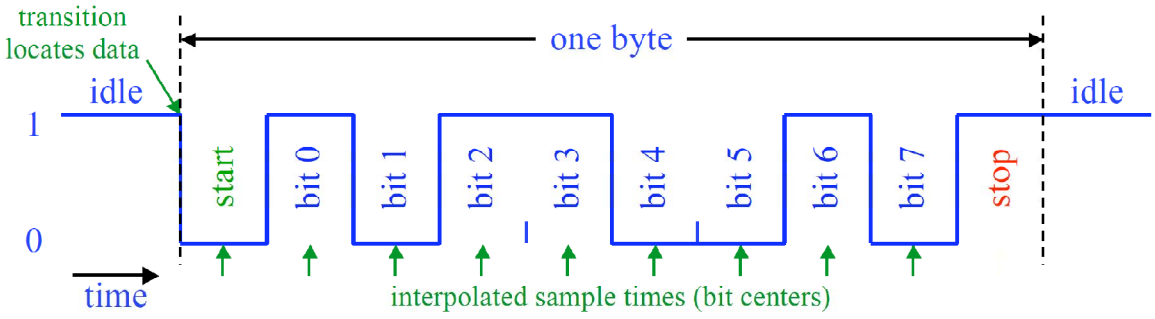
مفهوم الاتصالات التسلسلية المتزامنة (Synchronize) وغير المتزامنة (Asynchronize):

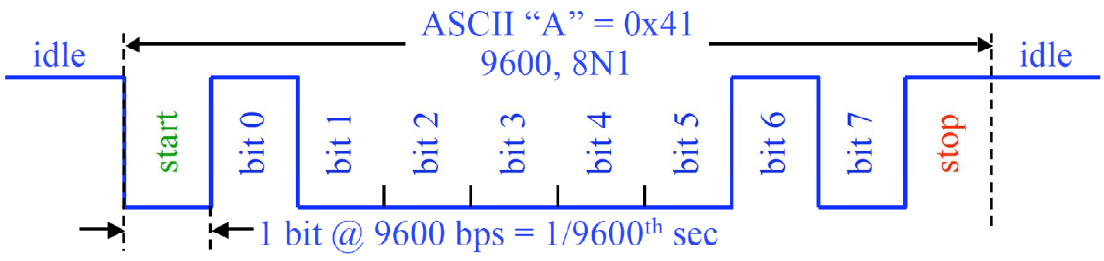
أولاً: الاتصالات المتواقتة (المتزامنة): يكون فيها بروتوكول الإرسال مؤلف من خطين على الأقل أحدهما خط التزامن (clock or strobe)، وبالتالي فإن سرعة إرسال البيانات تتحدد من خلال تردد إشارة التزامن بحث يتم إرسال كل بت من البتات تسلسلياً عند جبهة التزامن (صاعدة أو هابطة).



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

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

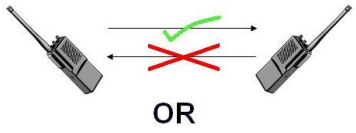




ملاحظة: بخلاف الاتصالات المتواقتة فإن ازدياد المسافة بين الطرفيتين لا يؤدي إلى فشل عملية النقل، كما أن هذه الطريقة أقل كلفة وأبسط بنية وأسهل برمجة.

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

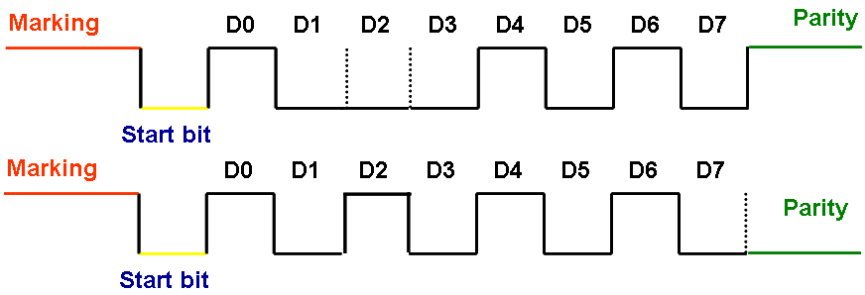
- تحديد نمط الإرسال: أحادي الاتجاه (Half-Duplex) أو ثنائي الاتجاه (Full-Duplex).
- تحديد عدد البتات لكل محرف: 6, 7 or 8 bit.
- تحديد معدل سرعة الإرسال (Baud Rate).
- تحديد استخدام أو عدم استخدام خانة فحص الإيجابية (Parity Bit)، وفي حال الاستخدام يجب تحديد نمط فحص خانة الإيجابية (Even or Odd).
- تحديد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



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

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

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



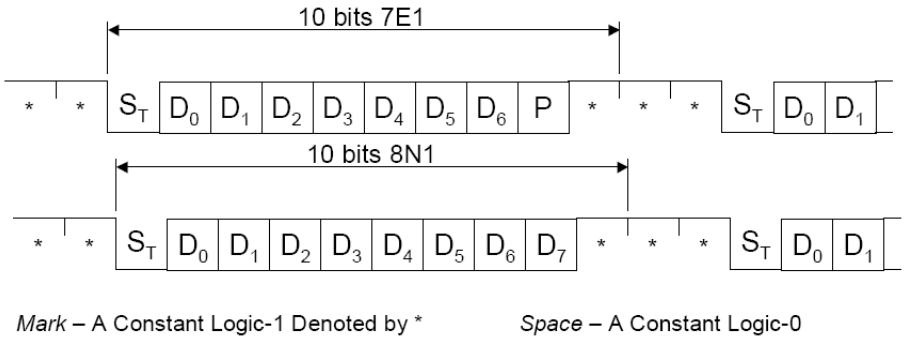
في حال كون خانة الإيجابية "Even" فإن هذه الخانة تملك القيمة "0" إذا كان عدد الواحدات في البايت المرسل زوجي وإلا فستصبح "1". الأمثلة التالية توضح ذلك.

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

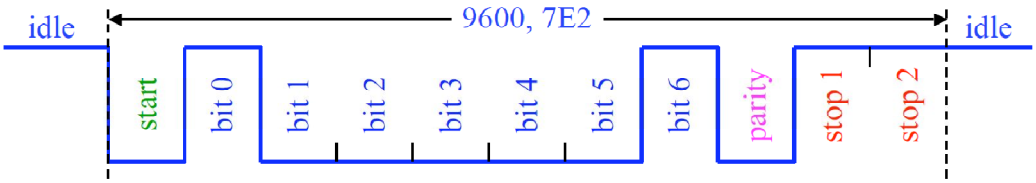
في حال كون خانة الإيجابية "Odd" فإن هذه الخانة تملك القيمة "0" إذا كان عدد الواحدات في البايت المرسل فردي وإلا فستصبح "1". الأمثلة التالية توضح ذلك.

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

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



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



معدل سرعة النقل (Baud Rate): وهو عدد البتات المرسلة خلال ثانية واحد على خط اتصال تسلسلي، وهناك قيم قياسية متعارف عليها لمعدلات النقل وهي:

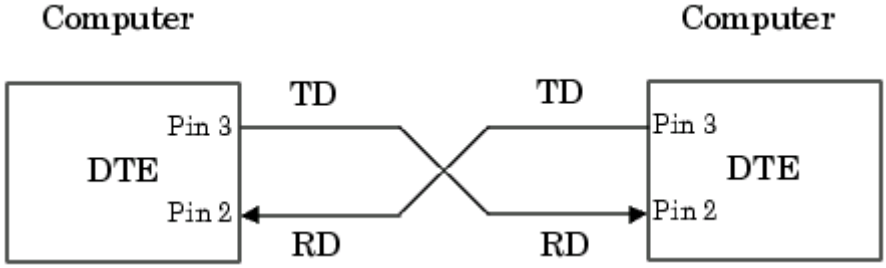
300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, etc...

إن الزمن اللازم لإرسال بت واحد يعطى بالعلاقة التالية:

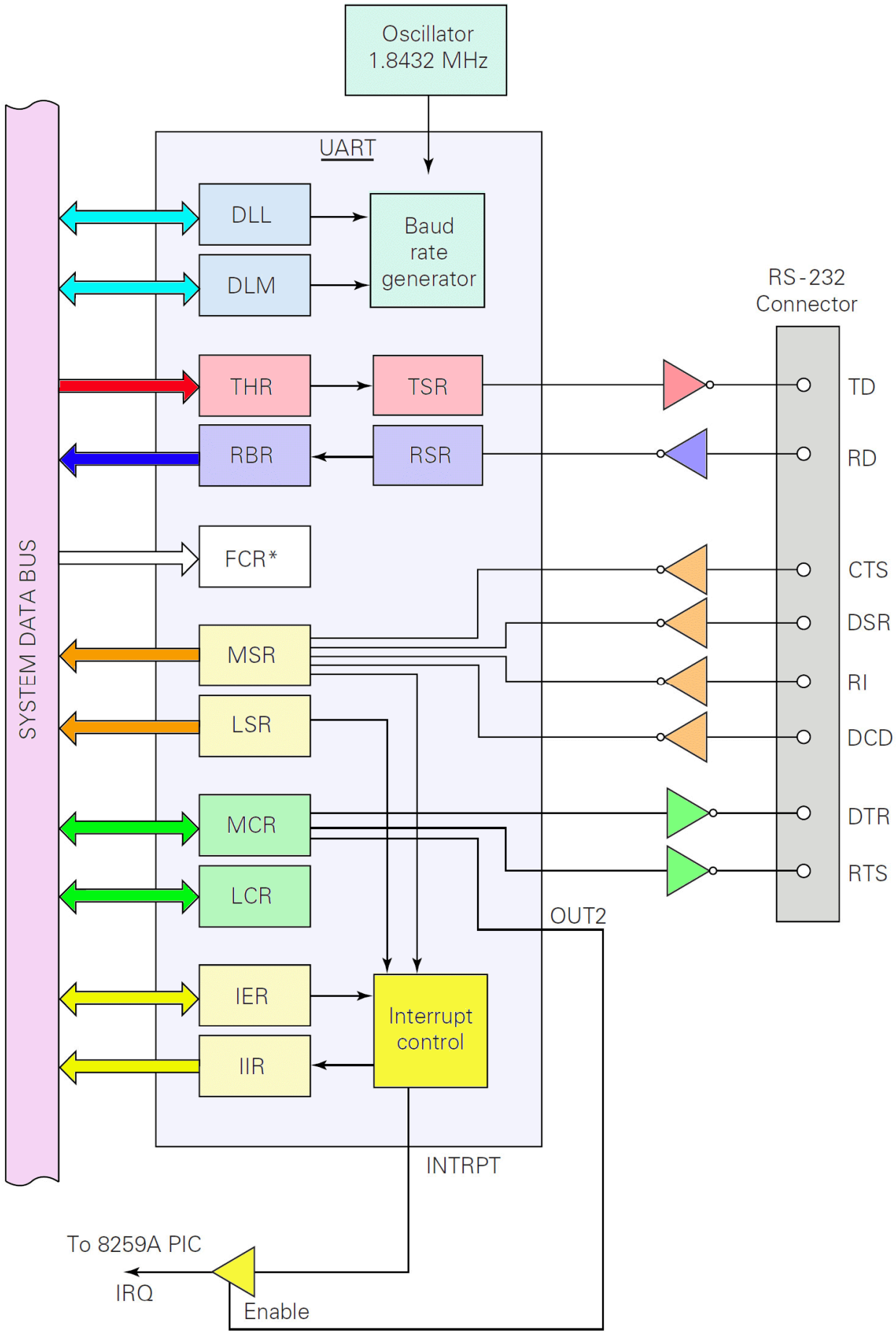
$$Bit_{Time} = \frac{1}{Baud\ Rate}$$

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

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



بنية النافذة التسلسلية RS232 في الحاسب:



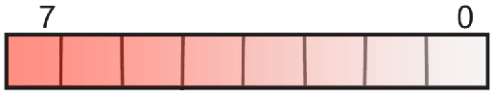
المسجلات الداخلية للنافذة التسلسلية RS232 في الحاسب:

إن بنية منفذ الاتصالات التسلسلية في الحاسب عبارة عن الدارة المتكاملة 8250-UART حيث تمتلك هذه بدورها مجموعة من المسجلات الوظيفية ومسجلات التحكم والحالة ومسجلات مقاطعات النافذة التسلسلية.

مسجل الدخل/الخرج (IOR, Input/Output Register):

COM1: 0x3F8 | COM2: 0x2F8

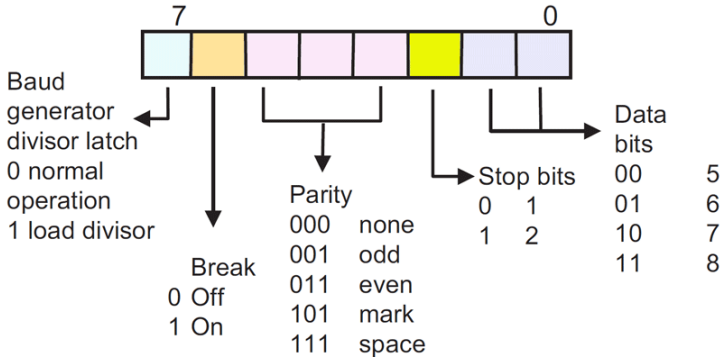
يتم منه قراءة البيانات الواردة عبر القطب RxD وإرسال البيانات الصادرة عبر القطب TxD.



مسجل التحكم بالخط (LCR, Line Control Register):

COM1: 0x3FB | COM2: 0x2FB

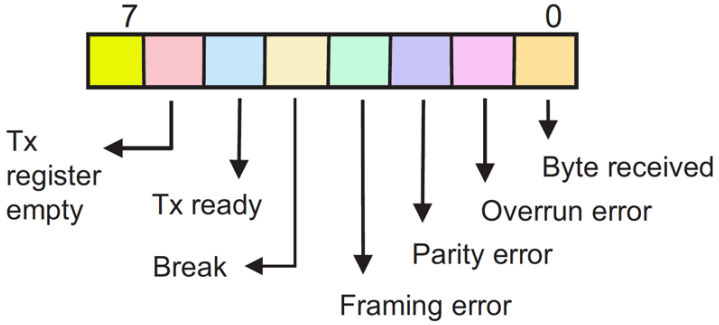
يتم فيه تعيين إعدادات (بارامترات) إطار البيانات.



مسجل حالة الخط (LSR, Line Status Register):

COM1: 0x3FD | COM2: 0x2FD

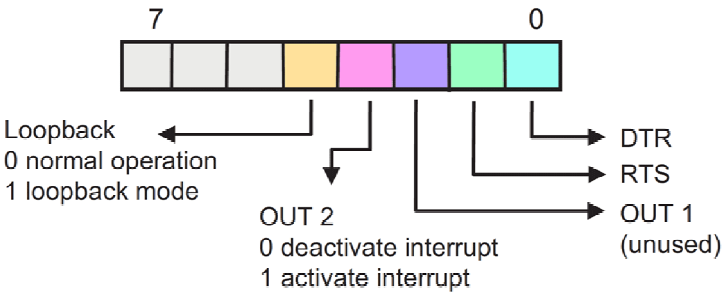
يتم منه قراءة حالة العمليات الجارية على الخط من أجل كشف الأخطاء والاستعلام عن حالة مسجل الإرسال.



مسجل التحكم بالمودم (MCR, Modem Control Register):

COM1: 0x3FC | COM2: 0x2FC

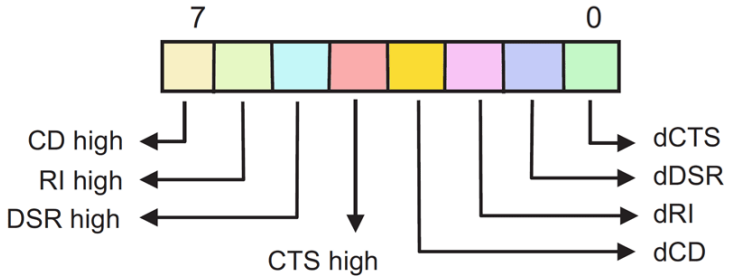
يتم فيه تعيين إعدادات (بارامترات) مصافحة التخاطب بين المرسال والمستقبل والتحكم بعمل الشريحة 8250.



مسجل حالة المودم (MSR, Modem Status Register):

COM1: 0x3FE | COM2: 0x2FE

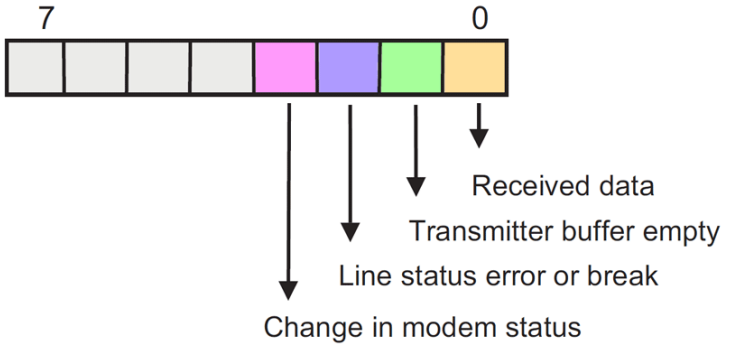
يتم منه قراءة حالة خطوط التحكم حيث أن "dxxx=1" إذا كانت حالة خطوط التحكم قد تغيرت منذ آخر عملية قراءة.



مسجل تفعيل المقاطعات (IER, Interrupt Enable Register):

COM1: 0x3F9 | COM2: 0x2F9

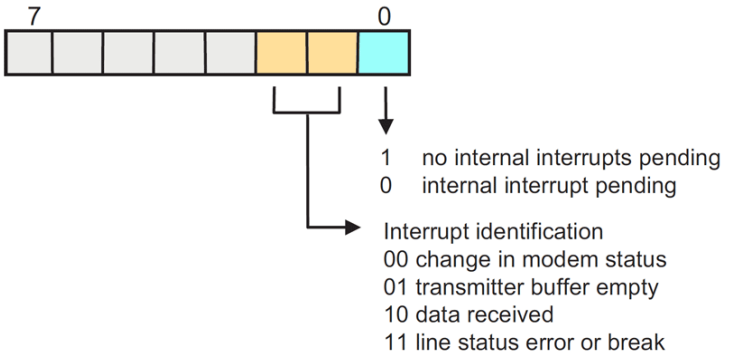
تملك النافذة التسلسلية COM أربعة مقاطعات داخلية (Active "1") موصلة إلى المعالج عن طريق أحد قطب مقاطعة المعالج، هذا القطب هو قطب المقاطعة IRQ4 للمنفذ COM1 وقطب المقاطعة IRQ3 للمنفذ COM2.



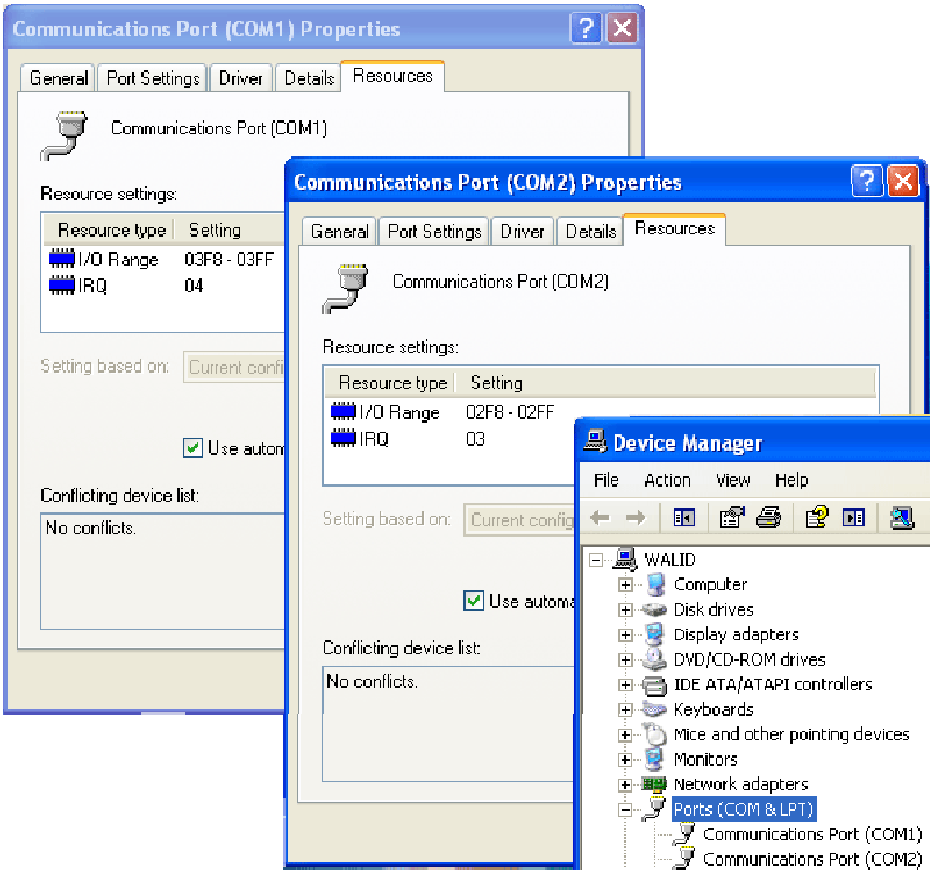
مسجل التعرف لهوية المقاطعة (IIR, Interrupt Identifcation Register):

COM1: 0x3FA | COM2: 0x2FA

يتم من خلاله معرفة نوع المقاطعة الحاصلة.



ملاحظة: في حال تواجد منفذ اتصالات COM3 مثلاً أو غيره، فيمكن الحصول على مجال عناوين مسجلات هذا المنفذ من إدارة أجهزة النظام في لوحة التحكم.



معدل النقل للنافذة التسلسلية:

يتم حساب قيمة معدل النقل استناداً إلى تردد هزاز كريستالي موجود على نفس الشريحة 8250 والذي يساوي إلى 1.8432MHZ، ومقسم "Divisor".

$$BAUD = \frac{1.8432 \times 10^6}{16 \times Divisor}$$

مثال: من أجل معدل نقل 9600bps أحسب قيمة Divisor؟

$$Divisor = \frac{1.8432 \times 10^6}{16 \times BAUD} = \frac{1.8432 \times 10^6}{16 \times 9600} = 12$$

إن القيمة D=12 هي قيمة المقسم ويجب تحميلها إلى النافذة UART8250 كمايلي:

- تفعيل Bit7=1 من مسجل التحكم بالخط (LCR).
- كتابة النبل الأدنى (LSB) من قيمة بايت المقسم إلى العنوان (0x3F8).
- كتابة النبل الأعلى (MSB) من قيمة بايت المقسم إلى العنوان (0x3F9).
- إلغاء تفعيل Bit7=0 من مسجل التحكم بالخط (LCR).

برمجة منفذ الاتصالات التسلسلي COM في بيئة VB, MVS2008.net:

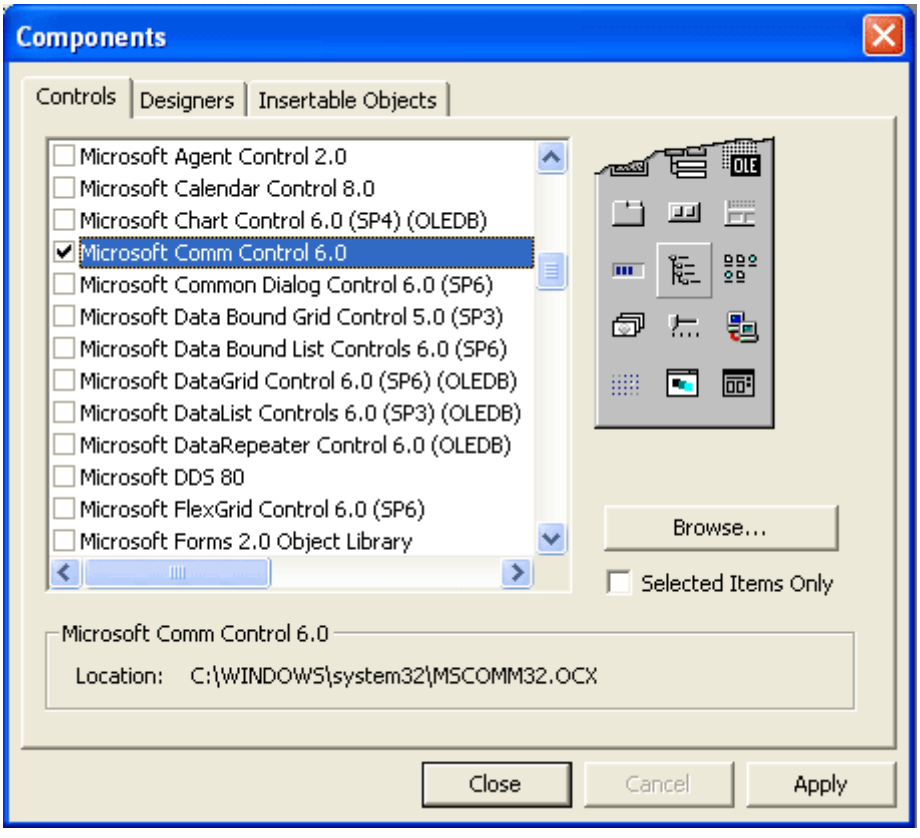
إن التعامل مع المسجلات بشكل مستقل يعتبر معقداً بعض الشيء، لذلك توفر البيئات البرمجية المرئية أدوات (ActiveX & OCX Components) تمكن المبرمج من القراءة والكتابة من مسجلات المنفذ بشكل مباشر كذلك استثمار المقاطعات والأحداث دون الحاجة إلى الوصول البرمجي المباشر لـ Bios، بالإضافة إلى إمكانية إعداد بارامترات المنفذ بشكل مبسط جداً.

إن هذه الأدوات تختلف باختلاف البيئة البرمجية المستخدمة أو الشركة المزودة.

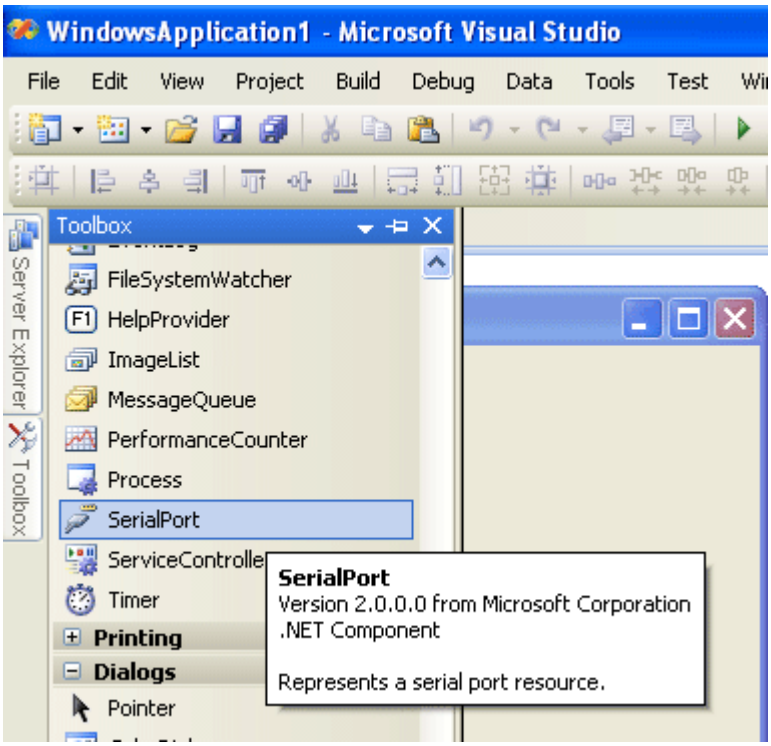
تمتلك بيئة VB6 أداة تسمى "MSComm" وهي عبارة عن "OCX" (MSCOMM32.ocx) تمكّن المستخدم من التخاطب مع منفذ الاتصالات التسلسلية COM بشكل مرن.

ملاحظة: إن هذه الأداة يجب تصيبها في مجلد النظام System32 ليتمكن المبرمج من التعامل معها، أو يمكن تنصيب مكتبات التحديث SP6 لبيئة VB6 وهي تحتوي على جميع الأدوات.

في حال كان المشروع الذي تم إنشاؤه هو "Standard EXE" فإنه يجب تحميل الأداة إلى شريط الأدوات في بيئة VB6 من مدير الأدواة كما في الشكل أدناه، أما في حال كان المشروع هو "Enterprice Edition" فسوف يتم تحميل جميع الأدوات المتقدمة والقياسية إلى شريط الأدوات.



أما بالنسبة للبرمجة في بيئة "Microsoft Visual Studio 2008" فالأمر مشابه تماماً لبيئة VB6 إلا أنّ الأداة أصبحت ضمن شريط الأدوات الأساسية وتدعى "SerialPort" كما أنها يمكن أن تستخدم في أي لغة برمجية داخل بيئة .net وذلك لأن الواجهة البرمجية والأدوات مشتركة وتختلف اللغة النصية فقط (VB.net, C#.net or C++.net).

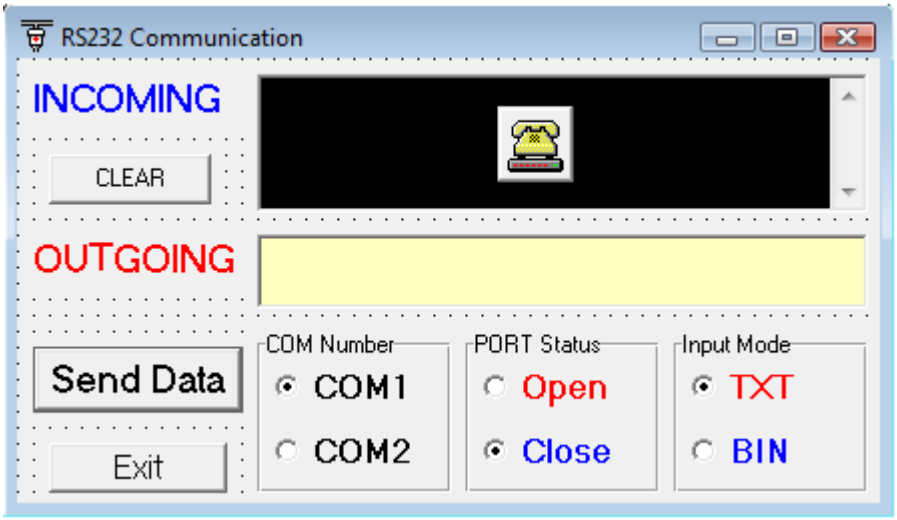


ملاحظة إن التعامل مع الموديل البرمجي للأداة SerialPort مشابه تماماً (إلا من تغييرات في شكل التعليمات) للموديل البرمجي في بيئة VB6.

أولاً: البرمجة في بيئة VB6:

سنقوم بإنشاء واجهة برمجية من أجل إرسال واستقبال البيانات بين حاسبين عبر منفذ الاتصالات التسلسلية (COM) وسوف نشرح التعليمات من خلال البرنامج الرئيسي.

الشكل جانباً يبين شكل الواجهة البرمجية (Test1/ProjRS232.vbp).



سوف يتم إرسال البيانات المكتوبة في مربع النص "OutGoing" عند الضغط على الزر "Send Data" كذلك سوف يتم استقبال جميع البيانات الواردة على النافذة االتسلسلية وإظهارها في مربع النص "InComing" بشكل آلي.

البرنامج:

```
Private Sub Form_Load()
    MSComm1.CommPort = 1
    MSComm1.Settings = "9600,N,8,1"
    MSComm1.RThreshold = 1
    MSComm1.InputLen = 0
    MSComm1.InBufferCount = 0
End Sub

Private Sub cmdClear_Click()
    txtOutput.Text = ""
    txtInput.Text = ""
End Sub

Private Sub optCOM1_Click()
    MSComm1.CommPort = 1
End Sub

Private Sub optCOM2_Click()
    MSComm1.CommPort = 2
End Sub

Private Sub optOpen_Click()
    MSComm1.PortOpen = True
End Sub

Private Sub optClose_Click()
    MSComm1.PortOpen = Flase
End Sub

Private Sub optTXT_Click()
    MSComm1.InputMode = comInputModeText
End Sub

Private Sub optBIN_Click()
    MSComm1.InputMode = comInputModeBinary
End Sub

Private Sub cmdSendData_Click()
    MSComm1.Output = txtOutput.Text & Chr(13)
End Sub

Private Sub cmdExit_Click()
    If MSComm1.PortOpen = True Then MSComm1.PortOpen = Flase
    End
End Sub
Private Sub MSComm1_OnComm()
Static sBuff As String
    If MSComm1.CommEvent = comEvReceive Then
        If optBIN.Value = True Then
            sBuff = sBuff & StrConv(MSComm1.Input, vbUnicode)
            txtInput.Text = sBuff
        Else
            txtInput.Text = txtInput.Text & MSComm1.Input
        End If
    End If
End Sub
```

شرح التعليمات الأساسية الخاصة بالأداة "MSComm":

```
MSComm1.CommPort = N
```

تعيين البوابة المطلوب برمجتها حيث "N" هو رقم البوابة.

```
MSComm1.Settings = "Baud,Parity,Bits,Stop"
```

تعيين بارامترات البوابة (معدل النقل، خانة الإيجابية، عدد بتات الإرسال، عدد بتات التوقف).

```
MSComm1.RThreshold = n
```

تحديد عدد المحارف التي يجب أن تتواجد في مسجل بفر الاستقبال قبل إطلاق الحدث "comEvReceive" (مقاطعة استقبال)، وفي حال كانت قيمة n=0 فسيتم إلغاء هذه المقاطعة.

```
MSComm1.InputLen = n
```

تحديد عدد المحارف التي سيتم إدخالها في كل عملية قراءة لبفر الاستقبال، وفي حال كانت قيمة n=0 فسيتم قراءة كامل محتوى البفر عند أول تعليمة قراءة.

```
MSComm1.InBufferSize = n
```

تحديد سعة مسجل بفر الاستقبال (1~1024).

```
MSComm1.OutBufferSize = n
```

تحديد سعة مسجل بفر الإرسال (1~1024).

```
MSComm1.InBufferCount = n
```

تعود بعدد المحارف الموجودة في مسجل بفر الاستقبال.

```
MSComm1.OutBufferCount = n
```

تعود بعدد المحارف الموجودة في مسجل بفر الإرسال.

```
MSComm1.PortOpen = True | Flase
```

فتح | إغلاق البوابة التسلسلية.

```
MSComm1.InputMode = comInputModeText | comInputModeBinary
```

تعيين شكل البيانات (محرفي | رقمي) التي سيتم قرائتها باستخدام التعليمة "Input" والموافقة لشكل البيانات المرسلة.

```
var = MSComm1.InPut
```

إدخال البيانات من مسجل بفر الاستقبال.

```
MSComm1.OutPut = var
```

إرسال البيانات إلى مسجل بفر الإرسال.

```
MSComm1.CommEvent = Value
```

تعود بقيمة تحدد آخر حدث أو خطأ تم في النافذة التسلسلية.

Value	الحدث
comEvCD	حدث تغير في حالة القطب CD
comEvCTS	حدث تغير في حالة القطب CTS
comEvDSR	حدث تغير في حالة القطب DSR
comEvRing	حدث كشف الرنين على القطب RI
comEvReceive	حدث اكتمال استقبال عدد المحارف المحدد في RThreshold في بفر الاستقبال.
comEvSend	حدث اكتمال تواجد عدد المحارف المحدد في SThreshold في بفر الإرسال.
comEvEOF	حدث كشف محرف نهاية الإرسال (vbCrLf).

```
MSComm1.DTREnable = True | Flase
```

تفعيل | إلغاء | قراءة حالة القطب DTR. من أجل (True) فإن القطب سيصبح "1" عندما يكون المنفذ مفتوح، و "0" عندما يكون المنفذ مغلق. من أجل (Flase) فإن حالة القطب ستكون "0" بشكل دائم.

```
MSComm1.Handshaking = comNone|comRTS|comXOnXoff|comRTSXOnXOff
```

تحديد نمط عمل المصافحة للنافذة التسلسلية.

```
MSComm1.RTSEnable = True | Flase
```

تفعيل | إلغاء | قراءة حالة القطب RTS من أجل نمط مصافحة Hardware. فمن أجل (True) فإن القطب سيصبح "1" عندما يكون المنفذ مفتوح، و "0" عندما يكون المنفذ مغلق. من أجل (Flase) فإن حالة القطب ستكون "0" بشكل دائم.

طرق قراءة محتويات مسجل الاستقبال:

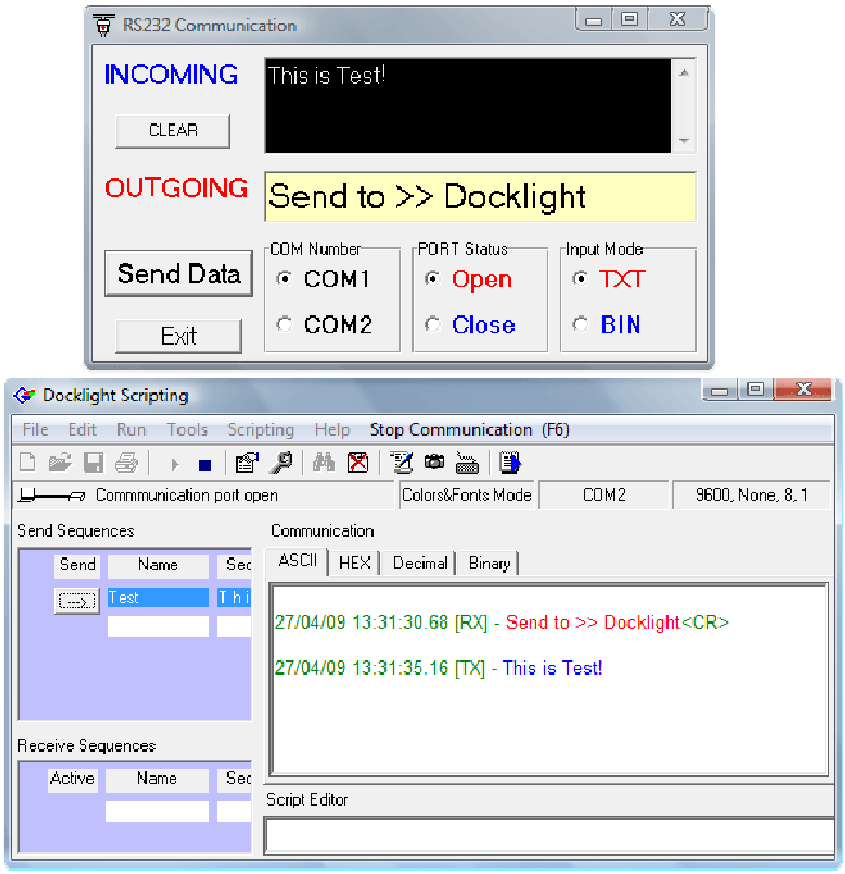
يوجد طريقتان لقراءة البيانات من مسجل الاستقبال للنافذة التسلسلية:

- الفحص الدوري للمسجل (Poling the Port): تتم هذه الطريقة باستخدام مؤقت زمني بحث أنه كلما تحقق حدث المؤقت يتم فحص محتوى مسجل البيانات للنافذة التسلسلية وفي حال وجدت بيانات يتم قرائتها.

هذه الطريقة مفيدة جداً في حال معرفة أطوال بلوكات البيانات التي يتم إرسالها مختلفة ولكنها تبدأ ببايت تعريف بداية البلوك (Header Byte) وتنتهي ببايت تعريف نهاية البلوك (Footer Byte).

- باستخدام مقاطعات الأحداث (OnComm() event): تتم هذه الطريقة باستخدام أحداث النافذة التسلسلية OnComm حيث يتم القفز إلى برنامج تحقق أحد أحدث النافذة ويتم تنفيذ البرنامج لموافق لحالة الحدث.

هذه الطريقة مفيدة جداً في حال معرفة أطوال بلوكات البيانات التي سيتم استلامها، كما انها أفضل باعتبار أن المعالج لن ينشغل بتفحص المسجلات بشكل دائم.



ثانياً: البرمجة في بيئة Matlab:

يوجد في بيئة البرنامج Matlab تعليمات برمجية تمكن المبرمج من التعامل مع المنفذ التسلسلي، حيث أن هذه التعليمات هي عبارة عن موديولات برمجية تم بنائها أصلاً في نفس البيئة

التعليمات الأساسية:

```
obj = serial('Port','PropertyName',PropertyValue,...)
Ser = serial('COM1','BaudRate',9600,'DataBits',8, 'Parity','non');
```

تحديد بارامترات المنفذ التسلسلي (رقم المنفذ، معدل النقل، عدد بتات الإرسال).

```
fopen(obj)
fopen(Ser);
```

فتح المنفذ التسلسلي.

```
fclose(obj)
fclose(Ser);
```

إغلاق المنفذ التسلسلي.

```
delete(obj)
delete(Ser);
```

تحرير البارامترات من الذاكرة.

```
fprintf(fid, format, A, ...)
fprintf(Ser,'This is Test');
```

إرسال البيانات بشكل محرفي (TXT) إلى مسجل الإرسال.

```
fwrite(fid, format, A, ...)
fwrite(Ser,4);
```

إرسال البيانات بشكل ثنائي (BIN) إلى مسجل الإرسال.

```
A = fscanf(fid, format)
A = fscanf(Ser);
```

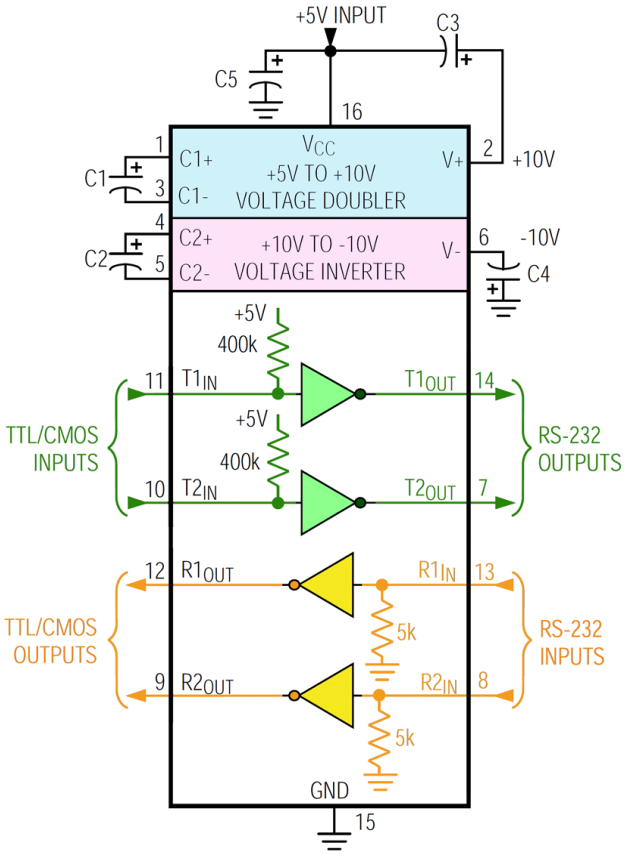
قراءة البيانات بشكل محرفي (TXT) من مسجل الاستقبال.

```
A = fread(fid)
A = fread(Ser);
```

قراءة البيانات بشكل ثنائي (BIN) من مسجل الاستقبال.

البرنامج:

```
ser = serial('COM1','BaudRate',9600,'DataBits',8);
fopen(ser)

fprintf(ser,'This is Test')
A = fscanf(ser);
fprintf(ser,A)

for i=1:5
    fwrite(ser,i);
end
A = fread(ser);
fwrite(ser,A);

fclose(ser)
delete(ser)
clear ser
```

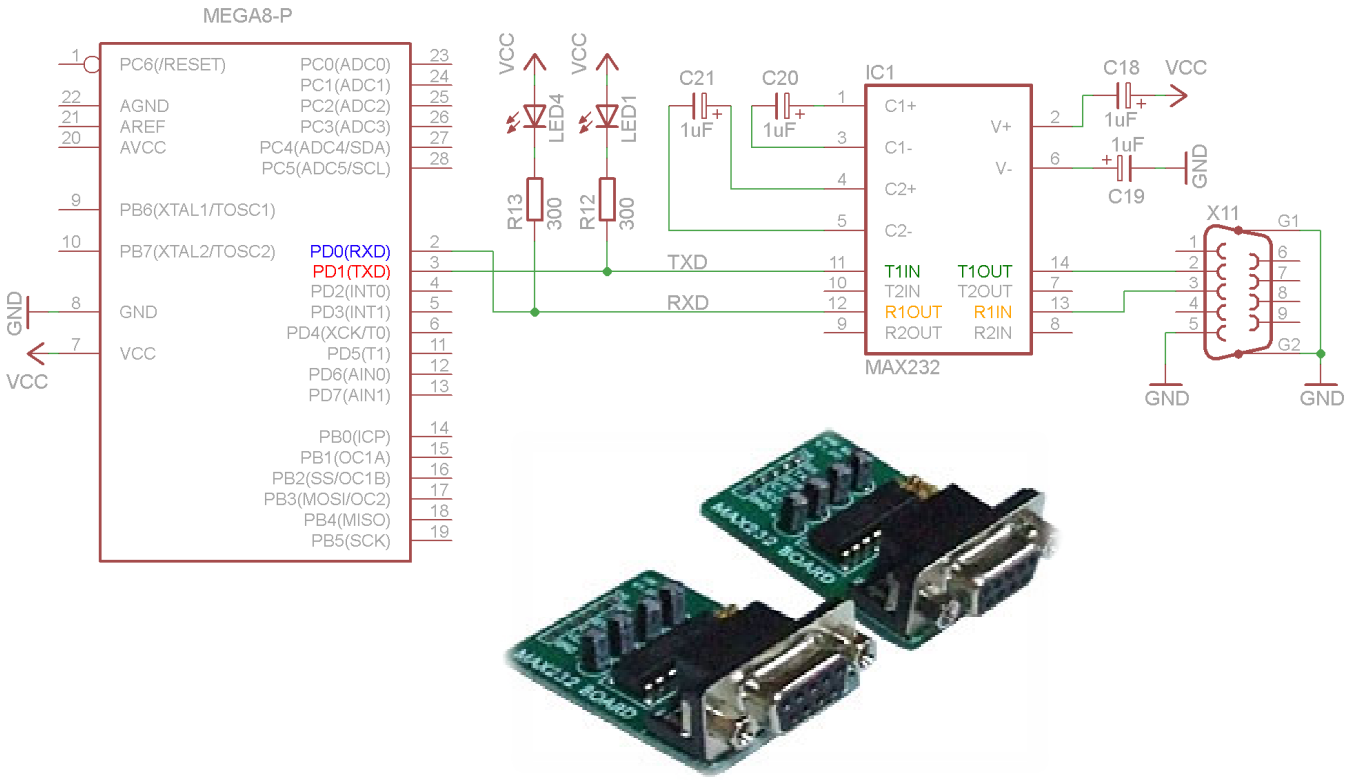


دارات الملائمة TTL <> RS232:

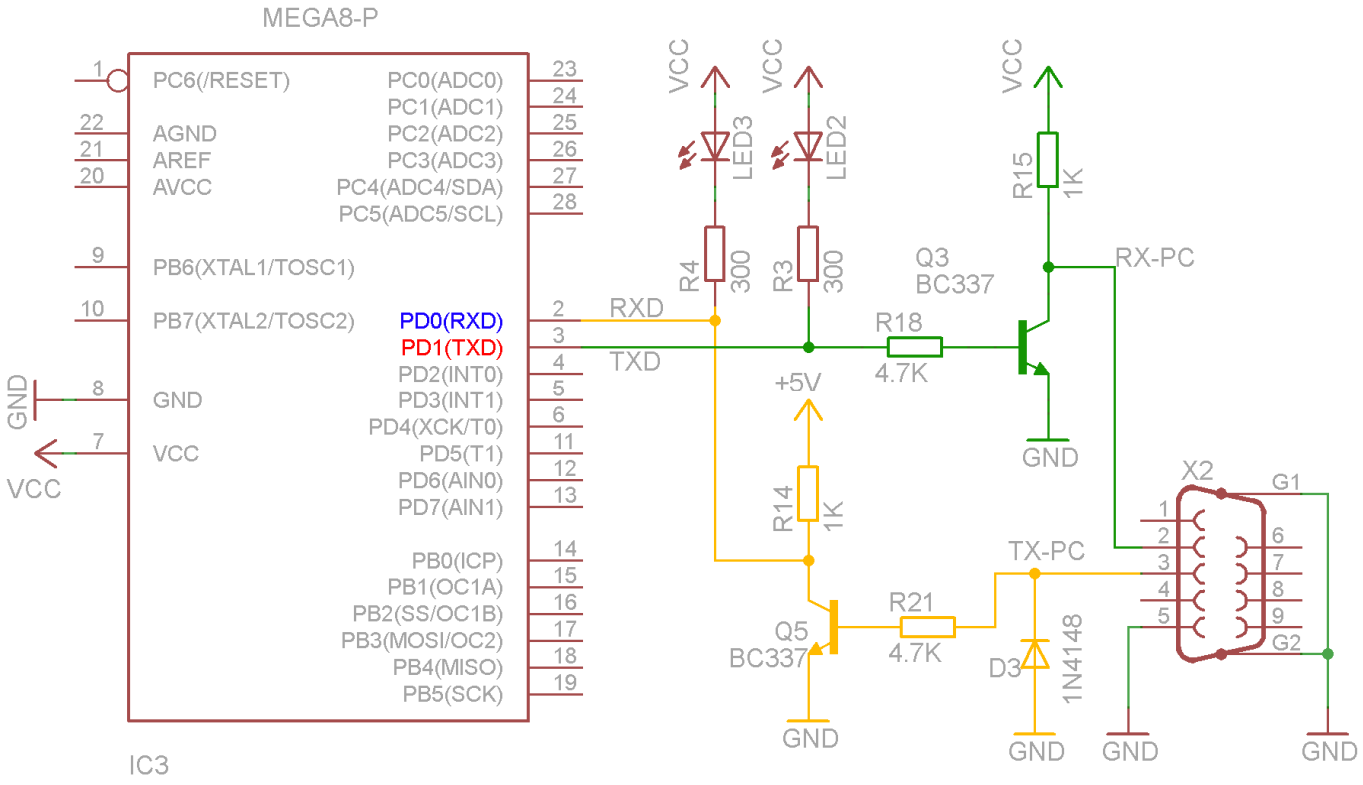
إن المستويات المنطقية للبروتوكول RS232 تختلف عن المستويات المنطقية للمتحكمات المصغرة وللدارات الرقمية الأخرى التي تعتمد المنطق TTL في عملها، وبالتالي نحتاج إلى دارة وسيطية (Adapter) من أجل الملائمة بين الطرفين.

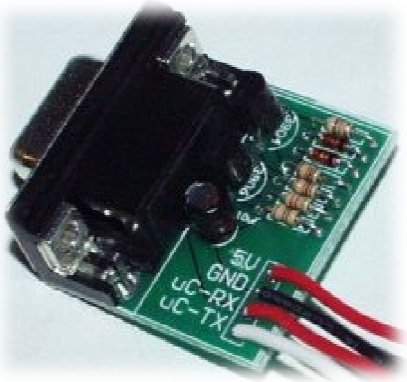
تستخدم الدارة المتكاملة Max232 كدارة تحويل وعزل TTL<>RS232.

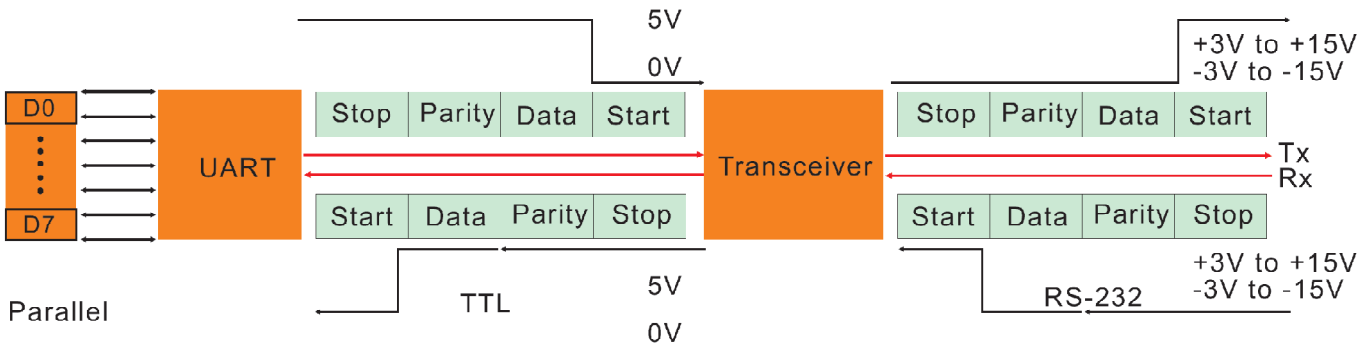
الشكل التالي يبين طريقة تحقيق دارة ملائمة TTL<>RS232 بين منفذ الحاسب التسلسلي (RS232) وبين نافذة تسلسلية (UART) لمتحكم مصغر باستخدام الدارة المتكاملة Max232.



الشكل التالي يبين طريقة تحقيق دارة ملائمة TTL<>RS232 بين منفذ الحاسب التسلسلي (RS232) وبين نافذة تسلسلية (UART) لمتحكم مصغر باستخدام وصلة مفاتيح ترانزستورية.





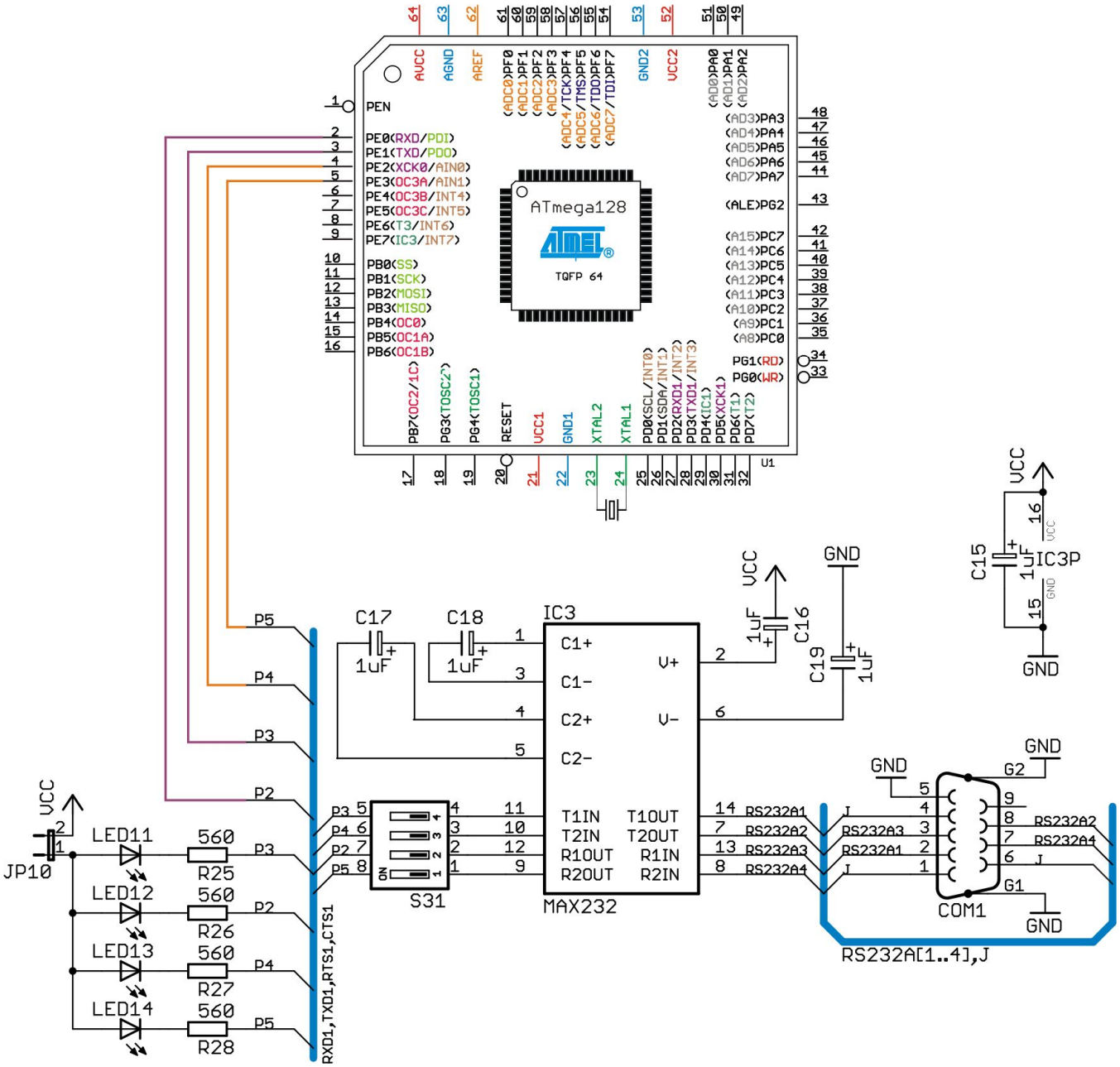


Exp.27: UART Interface التجربة السابعة والعشرون: النافذة التسلسلية UART

الغاية من التجربة:

استثمار وبرمجة النافذة التسلسلية UART.

مخطط التوصيل:



متطلبات التوصيل:

يجب إغلاق المفتاحين 2,4 في S31 من أجل وصل القطبين TxD, RxD.

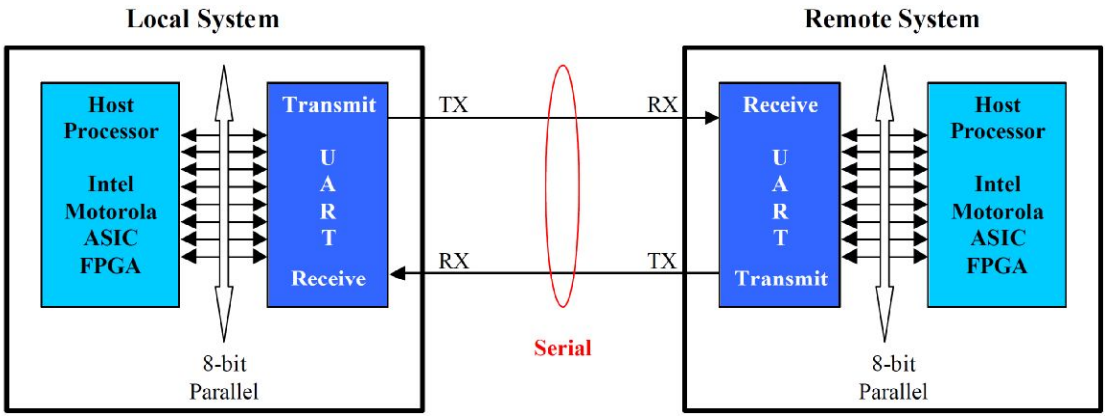
شرح عمل الدارة:

تحوي الدارة أعلاه على دارة ملائمة بين المتحكم المصغر ومنفذ الاتصال التسلسلي RS232 للحاسب. سوف نقوم بكتابة برنامج للقراءة والكتابة إلى النافذة التسلسلية UART.

النافذة التسلسلية UART (Universal Asynchronous Receiver and Transmiter Int erface):

تعتبر هذه النافذة من أكثر نوافذ الاتصال التسلسلي استخداماً في الأنظمة الرقمية ومبدأ عملها وكذلك بروتوكولها متوافق تماماً مع البروتوكول RS232 إلا أن المستويات المنطقية فيها وفق المنطق TTL، لذلك تستخدم دارات التحويل والملائمة كوسيط بين المنفذ التسلسلي RS232 وبين النافذة التسلسلية UART.

تتميز بسهولة وبساطة استخدامها بالإضافة إلى الكلفة المنخفضة للربط بين متحكمين (MCU-MCU)، أو الربط بين حاسب ومتحكم (MCU-PC).

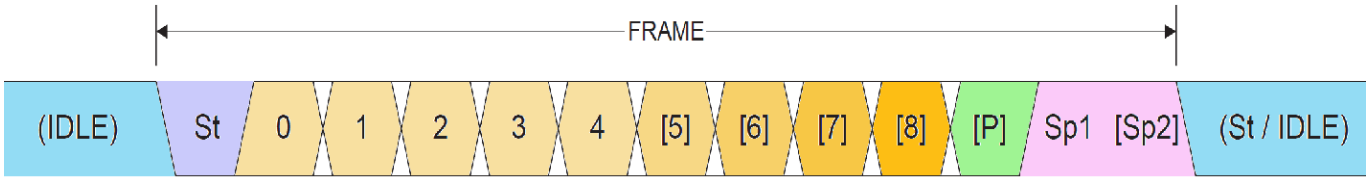


تملك النافذة التسلسلية في متحكمات العائلة AVR على ميزات عديدة وهي تعمل في نمطين مستقلين:

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

بنية إطار البيانات (UART Frame Format):

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



St: Start bit, always low.

Data bits: (0 to 8).

P: Parity bit (Can be odd or even)

Sp: Stop bit, always high.

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

حساب قيمة مسجل معدل النقل (Baud Rate Register):

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

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

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

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

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

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

يمكن حساب الخطأ من العلاقة التالية:

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

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

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

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

التعليمات الجديدة:

التعليمة	شرح التعليمة
`$baud = Var`	تحديد معدل النقل العام للنافذة التسلسلية (Hardware) UART0.
`Baud = Var`	تغيير معدل النقل للنافذة التسلسلية (Hardware) UART0 أثناء تنفيذ البرنامج.
`$BAUD1 = var`	تحديد معدل النقل العام للنافذة التسلسلية (Hardware) UART1.
`Baud1 = Var`	تغيير معدل النقل للنافذة التسلسلية (Hardware) UART1 أثناء تنفيذ البرنامج.
`Baud #x , BaudRate`	تعيين معدل نقل لنافذة تسلسلية برمجية (Software) UART0 رقم قناة الاتصال x
`Baud1 #x , BaudRate`	تعيين معدل نقل لنافذة تسلسلية برمجية (Software) UART1 رقم قناة الاتصال x
`Print Var ; "const"`	إرسال البيانات عبر النافذة التسلسلية UART0.
`Print [#channel,] Var; "const"`	إرسال البيانات عبر النافذة التسلسلية UARTx ورقم قناة اتصالها هو channel.
`Printbin Var [; Varn]`	إرسال البيانات بصيغة ثنائية عبر النافذة التسلسلية UARTx. `[; Varn]`: خيار من أجل تحديد عدد البايتات المراد إرسالها (مصفوفة).
`Printbin #channel, Var [;Varn]`	إرسال البيانات عبر النافذة التسلسلية UARTx ورقم قناة اتصالها هو channel. `[; Varn]`: خيار من أجل تحديد عدد البايتات المراد إرسالها (مصفوفة).
`Input ["prompt"] , Var [,Varn]`	قراءة البيانات الواردة من اتصال RS232 على النافذة التسلسلية UART0. `["prompt"]`: خيار يقوم بإرسال رسالة نصية قبل قراءة محتوى النافذة. `Var`: المتحول الذي سيتم إدخاله (رقمي، محرفي). `[,Varn]`: خيار من أجل إدخال أكثر من متحول بنفس التعليمة (n=1,2...).
`Input #ch, Var [,Varn ]`	قراءة البيانات الواردة من اتصال RS232 على النافذة UART0 ورقم قناتها هو ch.
`Inputbin Var1 [,Var2]`	قراءة البيانات الواردة من اتصال RS232 على النافذة UART0 بصيغة ثنائية. `[,Var2]`: خيار من أجل تحديد عدد البايتات المراد إدخالها (مصفوفة).
`Inputbin #channel, Var1[,Var2]`	قراءة البيانات الواردة من اتصال RS232 على النافذة UARTx ورقم قناتها هو ch.
`Inputhex ["prompt"],Var[,Varn]`	قراءة البيانات الواردة من اتصال RS232 على النافذة UART0 بالصيغة HEX.
`var = INKEY()`	تعود بقيمة الـ Ascii لأول محرف في مسجل bufer النافذة التسلسلية UART0.
`Var = Inkey(#channel)`	تعود بقيمة الـ Ascii لأول محرف في مسجل bufer النافذة التسلسلية UARTx ورقم قناة اتصالها هو channel. إذا كان المسجل فارغاً تعود بالقيمة "0".
`var = WAITKEY()`	ينتظر وصول أول محرف إلى مسجل bufer النافذة التسلسلية UART0 ويعود بقيمة الـ Ascii له.
`Var = Waitkey(#channel)`	ينتظر وصول أول محرف إلى مسجل bufer النافذة التسلسلية UARTx ورقم قناة اتصالها هو channel ويعود بقيمة الـ Ascii له.
`Var = Ischarwaiting()`	يفحص محتوى bufer مسجل النافذة التسلسلية UART0 ويعود بالقيمة "1" إذا كان هناك أي محرف، وإلا فسوف يعود بالقيمة "0"، مع العلم أن هذه التعليمة تفحص محتوى المسجل ولا تؤثر على محتواه!

`Var = Ischarwaiting(#channel)`	يفحص محتوى bufer مسجل النافذة التسلسلية UARTx ويعود بالقيمة "1" إذا كان هناك أي محرف وإلا فسوف يعود بالقيمة "0".
`$Timeout = value`	تفعيل مدة انقضاء زمني للنافذة التسلسلية UART1,2 عند استخدام التعليمة Input، وعند انتهاء الفترة المحددة تقوم بتجاوز التعليمة Input أو Waitkey() وإن لم يكتمل الاستقبال، وتعمل فقط في حال عدم تعريف bufer النافذة التسلسلية.
`Echo On|off`	تفعيل|إلغاء إعادة طباعة المتحولات المدخلة عند استخدام التعليمة Input.
`Config Input = Term,` `Echo = Echo | Noecho`	يقوم بتوجيه المترجم (compiler) إلى تغيير محرف التحكم الأخير الذي يتم إرساله بعد كل بايت (CR, LF, CRLF or LFCR) ليتم به إنهاء القراءة لمحتوى الإرسال عند استخدام التعليمة Input.
`$serialinput2lcd`	يقوم بإظهار جميع البيانات المستلمة أو المرسلة على النافذة التسلسلية على شاشة الإظهار الكريستالية بدلاً من إظهارها في نافذة Terminal.

```
Config Serialin|Serialin1|Serialin2|Serialin3 = Buffered, Size = Size
[, Bytematch = All|byte|none]
[, Cts = Pin, Rts = Pin, Threshold_full = Num, Threshold_empty = Num]
```

إعداد مسجل تجميع لدخل (Input Bufer) النافذة التسلسلية المحددة بـ Serialin يتم حجزه في ذاكرة SRAM حيث أنّ:

$Size_{MAX}$= **255**

- *SERIALIN* : 1st UART Hardware Interface > UART0
- *SERIALIN1* : 2nd UART Hardware Interface > UART1
- *SERIALIN2* : 3rd UART Hardware Interface > UART2
- *SERIALIN3* : 4th UART Hardware Interface > UART3

من أجل "Bytematch=byte" يتم فيه تحديد قيمة ما "ASCII" فإذا تطابقت مع بايت وارد على النافذة يتم القفز إلى برنامج فرعي لتنفيذه.

هذا البرنامج يجب أن يتوضع عند لافتة محددة حسب رقم النافذة التسلسلية المستخدمة على الشكل التالي:

- Serial0CharMatch (for SERIALIN or the 1st UART/UART0)
- Serial1CharMatch (for SERIALIN1 or the 2nd UART/UART1)
- Serial2CharMatch (for SERIALIN2 or the 3rd UART/UART2)
- Serial3CharMatch (for SERIALIN3 or the 4th UART/UART3)

من أجل "Bytematch=all" يتم القفز إلى برنامج فرعي لتنفيذه (يتوضع عند لافتة محددة) كلما ورد بايت على النافذة التسلسلية.

- Serial0ByteReceived (for SERIALIN or the 1st UART/UART0)
- Serial1ByteReceived (for SERIALIN1 or the 2nd UART/UART1)
- Serial2ByteReceived (for SERIALIN2 or the 3rd UART/UART2)
- Serial3ByteReceived (for SERIALIN3 or the 4th UART/UART3)

من أجل "Bytematch=none" لن يتم استدعاء أي برنامج فرعي ولا يوجد أي لافتة يتم تحديدها.

`Cts = Pin`: تحديد القطب الذي سيتم توصيل القطب "CTS" معه من أجل نمط المصافحة.

`Rts = Pin`: تحديد القطب الذي سيتم توصيل القطب "RTS" معه من أجل نمط المصافحة.

`Threshold_full = Num`: تحديد عدد البايتات التي ستجعل حالة القطب "RTS=1" من أجل إعلام المرسل أن بفر المستقبل ممتلئ ويجب التوقف عن إرسال المزيد من البايتات.

`Threshold_empty = Num`: تحديد عدد البايتات التي يجب أن تتوفر كمساحة حرة في البفر قبل أن تعود حالة القطب "CTS=0" من جديد ويمكن الآن إكمال الإرسال.

يجب إعداد كلا بفري دخل وخرج النافذة التسلسلية من أجل العمل في نمط المصافحة بوجود القطبين "CTS-RTS".

```
Config Serialout|Serialout1|Serialout2|Serialout3 = Buffered, Size = Size
```

إعداد مسجل تجميع لخرج (Output Bufer) النافذة التسلسلية المحددة بـ Serialout يتم حجزه في ذاكرة SRAM حيث أنّ:

$Size_{MAX}$= **255**

SERIALOUT : 1st UART Hardware Interface > **UART0**

SERIALOUT1 : 2nd UART Hardware Interface > **UART1**

SERIALOUT2 : 3rd UART Hardware Interface > **UART2**

SERIALOUT3 : 4th UART Hardware Interface > **UART3**

```
Config Comx = Baud , Synchrone = 0|1 , Parity = None|disabled|even|odd ,
Stopbits = 1|2 , Databits = 4|6|7|8|9 , Clockpol = 0|1
```

تحديد بارامترات الإعدادات المتقدمة للنافذة التسلسلية UART حيث أن "x" هو رقم قناة التسلسلية.

`Baud`: تحديد معدل النقل، ويمكن كتابة `'dummy'` من أجل استخدام نفس القيمة المحددة بالتعليمة $Baud.

`Synchrone = 0|1`: تحديد نمط العمل "متواقت | غير متواقت".

`Parity = None|disabled|even|odd`: تحديد بارامتر فحص خانة الإيجابية (فردي | زوجي | إلغاء | عدم فحصها).

`Stopbits = 1|2`: تحديد عدد بتات التوقف.

`Databits = 4|6|7|8|9`: تحديد عدد بتات بايت البيانات.

`Clockpol = 0|1`: تحديد جبهة التزامن في حال اختيار نمط التزامن.

```
Serin  Var , Bts , Port , Pin , Baud , Parity , Dbits , Sbits
Serout Var , Bts , Port , Pin , Baud , Parity , Dbits , Sbits
```

تتيح هاتين التعليمتين قراءة | إرسال البيانات من نافذة تسلسلية برمجية ديناميكية مع إمكانية استخدام نفس القطب في كلا التعليمتين ليكون نفسه قطب إرسال أو استقبال، وبالتالي يمكن إرسال البيانات باستخدام "Serout" ومن ثم استقبال البيانات الواردة على نفس القطب باستخدام "Serin"، بالإضافة إلى إمكانية تغيير بارامترات التعليمة أثناء تنفيذ البرنامج لأن هذه النافذة ديناميكية.

`Var`: تحديد المتحول الذي سيتم استقباله | إرساله.

`Bts`: تحديد عدد البايتات التي سيتم استقبالها | إرسالها.

`Port`: تحديد اسم البوابة.

`Pin`: تحديد القطب المستخدم من البوابة المذكورة.

`Baud`: تحديد معدل النقل.

`Parity`: تحديد بارامتر فحص خانة الإيجابية (NONE=0, EVEN=1, ODD=2).

`Dbits`: تحديد عدد بتات بايت البيانات (7,8).

`Sbits`: تحديد عدد بتات التوقف (1,2).

ملاحظة: باعتبار أن كل البارامترات متوفرة وتستخدم نفس القطب فإن هاتين التعليمتين سوف تحتاجان إلى مساحة أكبر من الذاكرة!

ملاحظة: يجب إلغاء المقاطعات الخارجية أثناء استخدام هاتين التعليمتين كي لا تؤثر على التزامن.

ملاحظة: سوف يتم استخدام القطب المحدد في نمط المجمع المفتوح وبالتالي يمكن توصيل عدد كبير من المعالجات عبر الناقل الرئيسي (Data Bus) ووضع مقاومة رفع خارجية.

ملاحظة: من أجل استقبال أو إرسال بيانات محرفية، يجب وضع القيمة `Bts=0`.

`Var = Bufspace(n)`	يعود بكمية المساحة المتوفرة في البفر المحدد بـ n حيث: n=0 : output buffer 1st UART | n=1 : input buffer 1st UART n=2 : output buffer 2nd UART | n=3 : input buffer 2nd UART

`Clear Serialin` `Clear Serialout`	إفراغ محتوى بفر النافذة التسلسلية (بفر الدخل | بفر الخرج).

`Open "device" For Mode As #channel`
 	
Examples:
 	
`'open hardware UART`
`Open "com1:" For Binary As #1`
 	
`'open a Transmit channel for output (software UART)`
`Open "comd.1:9600,8,n,1" For Output As #2`
 	
`'open a Receive channel for input (software UART)`
`Open "comd.0:9600,8,n,1" For Input As #3`
	فتح قناة الاتصال لنافذة تسلسلية (UART) حيث أن:
	"device":
	إذا كانت النافذة "Hardware": يتم اختيار رقم المنفذ Comx ("COM1:").
	إذا كانت النافذة "Software": يتم تحديد مواصفات وقطب الاتصال كما يلي:
	"COMpin:Speed,N,Parity,Stopbits[,Inverted]"
	حيث أن:
	COMpin: القطب الذي سيتم استخدامه كنافذة تسلسلية (دخل|خرج).
	Speed: معدل النقل (Baud).
	N: عدد بتات البايت الذي سيتم إرساله أو استقباله (6,7,8 or 9).
	Parity> N|O|E (زوجي | فردي | بدون).
	Stopbits: عدد بتات التوقف (1|2).
	[,Inverted]: خيار يتيح إمكانية عكس المستوى المنطقي للإشارة.
	Mode: يوجد خيارين وهما:
	إذا كانت النافذة "Hardware": Binary or Random من أجل Com1, Com2.
	إذا كانت النافذة "Software": **Input** أو **Output** لتحديد اتجاه النافذة.
	#channel: هو رقم قناة الاتصال.

`Close #channel`	إغلاق قناة الاتصال للنافذة التسلسلية البرمجية (software UART)
`Get #channel, Var`	قراءة بايت من مسجل نافذة تسلسلية (HW or SW) ذات قناة اتصال محددة.
`Put #channel , Var`	كتابة بايت إلى مسجل نافذة تسلسلية (HW or SW) ذات قناة اتصال محددة.
`Enable | Disable Urxc` `On Urxc Rx_isr`	تفعيل | إلغاء تفعيل مقاطعة اكتمال استقبال البيانات للنافذة التسلسلية UART. القفز إلى برنامج خدمة المقاطعة عند تحقق مقاطعة اكتمال الاستقبال.
`Enable | Disable Utxc` `On Utxc Tx_isr`	تفعيل | إلغاء تفعيل مقاطعة اكتمال إرسال البيانات للنافذة التسلسلية UART. القفز إلى برنامج خدمة المقاطعة عند تحقق مقاطعة اكتمال الإرسال.
`Enable | Disable Udre` `On Udre Empty_isr`	تفعيل | إلغاء تفعيل مقاطعة فراغ مسجل البيانات للنافذة التسلسلية UART. القفز إلى برنامج خدمة المقاطعة عند تحقق مقاطعة فراغ مسجل البيانات.
`Enable | Disable Serial`	تفعيل | إلغاء تفعيل جميع مقاطعات النافذة التسلسلية UART السابقة.
`$dbg` `Dbg`	تفعيل مشخص الأخطاء (Debugging) وإرسال البيانات على النافذة التسلسلية.

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

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

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

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

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

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

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

ملاحظة: عند استخدام تعليمة الإرسال على قناة محددة (**Print** #channel) أو القراءة على قناة محددة (**Input** #channel) فإنه يجب استخدام التعليمتين OPEN & CLOSE من أجل فتح القناة قبل الإرسال أو الاستقبال وإغلاقها عند الانتهاء.

ملاحظة: عند استخدام تعليمة (**Open** "device") فإن COM1 هو المنفذ الافتراضي ولا حاجة لتعريفه أو فتحه وإغلاقه باستخدام التعليمتين OPEN & CLOSE.

ملاحظة: في التعليمة **$Timeout** = value فإن القيمة value في التعليمة ليس لها واحدة زمنية، وإنما بالتجريب وجد أنها تعطي التأخيرات التالية:

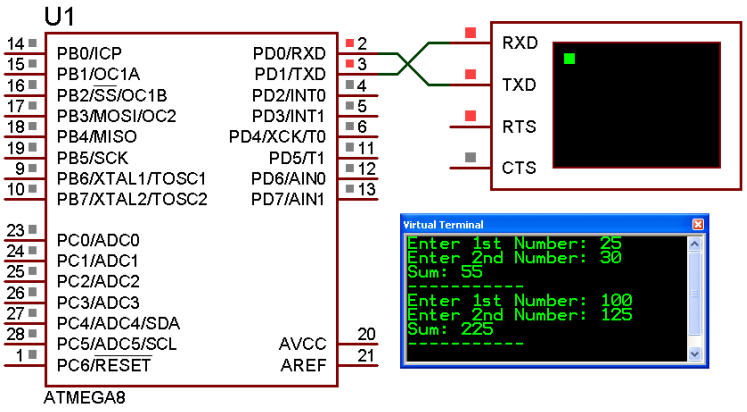
fosc	**$Timeout** = value			
	Value = 100	**Value = 1000**	**Value = 10000**	**Value = 100000**
1MHZ	1.6ms	16ms	160ms	1600ms
2MHZ	0.8ms	8ms	80ms	800ms
4MHZ	0.4ms	4ms	40ms	400ms
8MHZ	0.2ms	2ms	20ms	200ms
16MHZ	0.1ms	1ms	10ms	100ms

الجدول التالي يبين قيم ASCII للوحة المفاتيح.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENQ 5	ACK 6	BEL 7	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
1	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31
2	SPC 32	! 33	" 34	# 35	$ 36	% 37	38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127
8	€ 128	پ 129	‚ 130	ƒ 131	„ 132	… 133	† 134	‡ 135	ˆ 136	‰ 137	ٹ 138	‹ 139	Œ 140	چ 141	ژ 142	ڈ 143
9	گ 144	‘ 145	’ 146	“ 147	” 148	• 149	– 150	— 151	ک 152	™ 153	ڑ 154	› 155	œ 156	157	158	ں 159
A	160	ء 161	¢ 162	£ 163	¤ 164	¥ 165	¦ 166	§ 167	¨ 168	© 169	ھ 170	« 171	¬ 172	- 173	® 174	¯ 175
B	° 176	± 177	² 178	³ 179	´ 180	µ 181	¶ 182	· 183	¸ 184	¹ 185	؛ 186	» 187	¼ 188	½ 189	¾ 190	؟ 191
C	ہ 192	ء 193	آ 194	أ 195	ؤ 196	إ 197	ئ 198	ا 199	ب 200	ة 201	ت 202	ث 203	ج 204	ح 205	خ 206	د 207
D	ذ 208	ر 209	ز 210	س 211	ش 212	ص 213	ض 214	× 215	ط 216	ظ 217	ع 218	غ 219	ـ 220	ف 221	ق 222	ك 223
E	à 224	ل 225	â 226	م 227	ن 228	ه 229	و 230	ç 231	è 232	é 233	ê 234	ë 235	ى 236	ي 237	î 238	ï 239
F	ً 240	ٌ 241	ٍ 242	َ 243	ô 244	ُ 245	ِ 246	÷ 247	ّ 248	ù 249	ْ 250	û 251	ü 252	253	254	ے 255

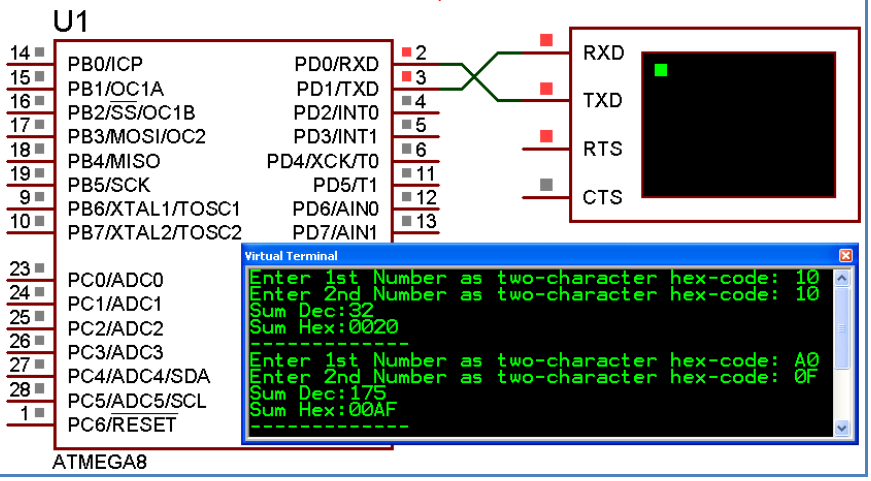
إدخال البيانات باستخدام التعليمة Input:

```
$regfile = "m8def.dat"
$crystal = 4000000
$baud = 9600
'----------------------
Dim Num1 As Integer
Dim Num2 As Integer
Dim Sum As Integer
'----------------------
Do
   Num1 = 0 : Num2 = 0
   Input "Enter 1st Number: " , Num1
   Input "Enter 2nd Number: " , Num2

   Sum = Num1 + Num2
   Print "Sum: " ; Sum
Loop
End
```



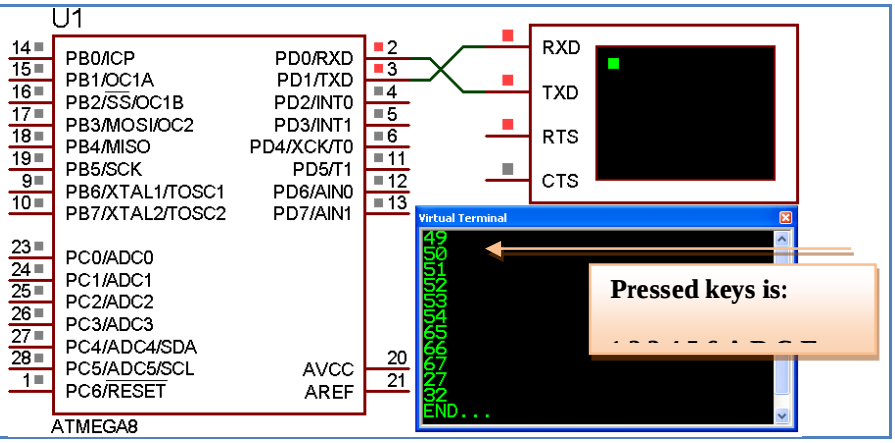
إدخال البيانات باستخدام التعليمة InputHex:

```
$regfile = "m8def.dat"
$crystal = 4000000
$baud = 9600
'----------------------
Dim Num1 As Byte, Num2 As Byte, Sum As Word
'----------------------
Do
   Num1 = 0 : Num2 = 0
   Inputhex "Enter 1st Number as two-character hex-code: " , Num1
   Inputhex "Enter 2nd Number as two-character hex-code: " , Num2

   Sum = Num1 + Num2
   Print "Sum Dec:" ; Sum
   Print "Sum Hex:" ; Hex(sum)
   Print "-------------"
Loop
End
```



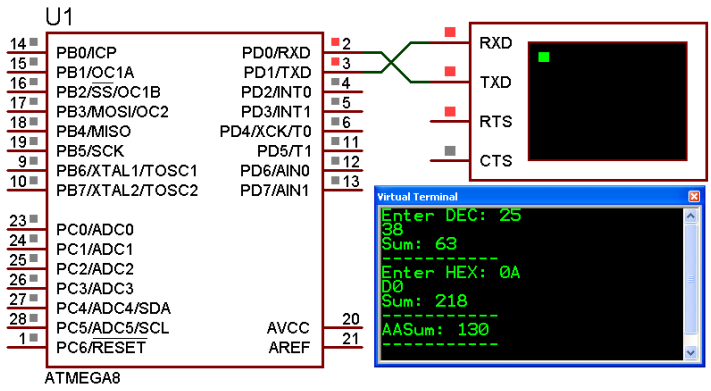
إدخال البيانات باستخدام التعليمة Waitkey():

```
$regfile = "m8def.dat"
$crystal = 4000000
$baud = 9600
'----------------------
Dim Inchar As Byte
'----------------------
Do
   Inchar = Waitkey()
   Print Inchar
Loop Until Inchar = " "
Print "END..."
End
```



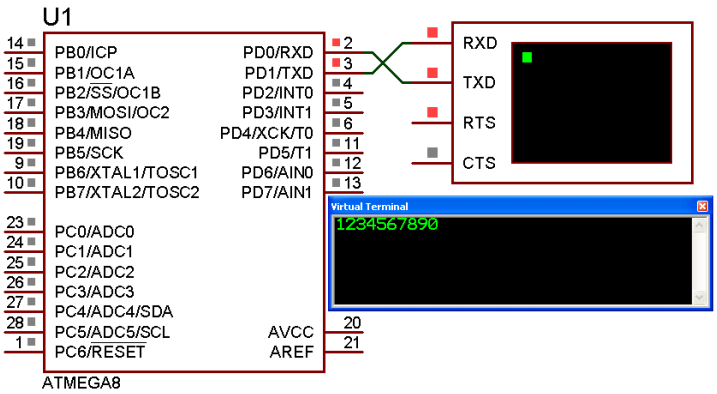
إدخال البيانات بشكل مصفوفي باستخدام التعليمة Input, Inputhex:

```
$regfile = "m8def.dat"
$crystal = 4000000
$baud = 9600
'----------------------
Dim Num1 As Byte, Num2 As Byte
Dim Sum As Integer, Arr(2) As Byte
'----------------------
Do
   Num1 = 0 : Num2 = 0
   Input "Enter DEC: " , Num1 , Num2
   Sum = Num1 + Num2
   Print "Sum: " ; Sum
   Print "-----------"

   Inputhex "Enter HEX: ", Num1, Num2
   Sum = Num1 + Num2
   Print "Sum: " ; Sum ;
   Print "-----------"
Loop
End
```



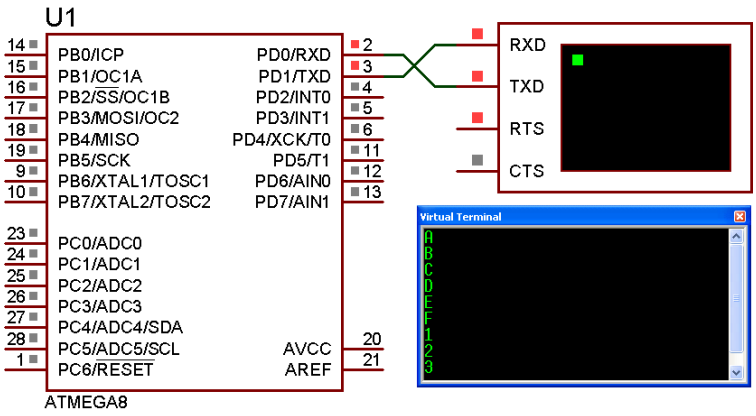
تفعيل بفر دخل وبفر خرج للنافذة التسلسلية UART0:

```
$regfile = "m8def.dat"
$crystal = 4000000
$baud = 19200
'----------------------
Config Serialin = Buffered , Size = 10
Config Serialout = Buffered , Size =10
Enable Interrupts
'----------------------
Dim Arr(10) As Byte
'----------------------
Baud = 9600
Do
   If Ischarwaiting() = 1 Then
      Inputbin Arr(1) , 10
      Printbin Arr(1) , 10
      Waitms 10
      Clear Serialin
      Clear Serialout
   End If
Loop
End
```



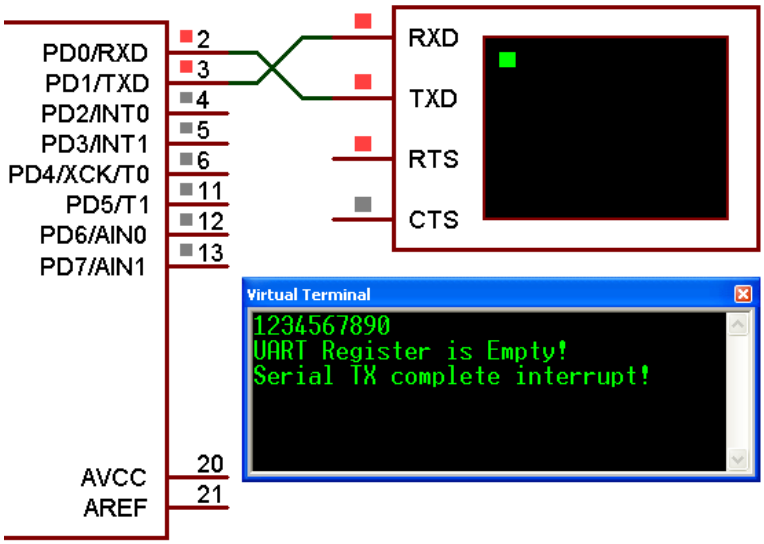
مقاطعة اكتمال استقبال البيانات:

```
$regfile = "m8def.dat"
$crystal = 4000000
$baud = 9600
'----------------------
Enable Urxc
On Urxc Getchar
Enable Interrupts
'----------------------
Dim Inchar As String * 1
'----------------------
Do
   nop
Loop Until Inchar = " "
'----------------------

Getchar:
   Inchar = Inkey() : Print Inchar
Return
```



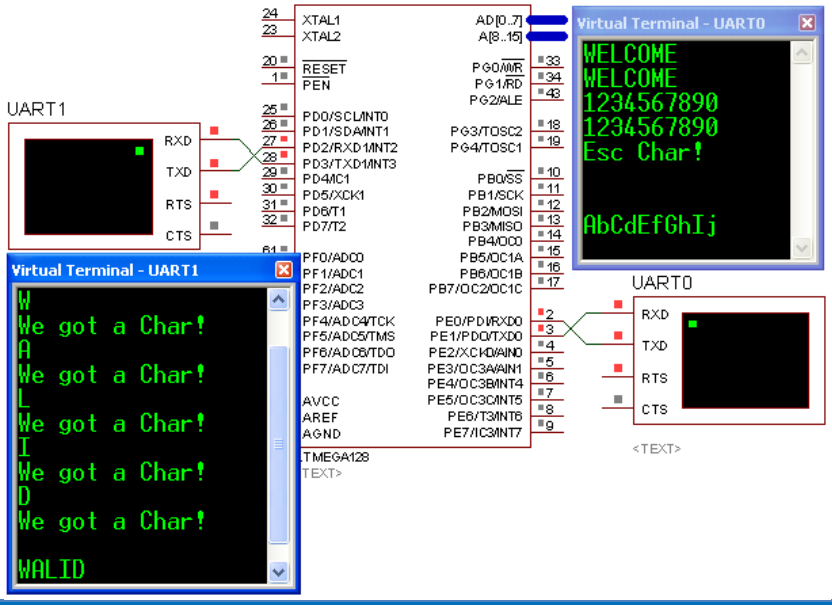
مقاطعات النافذة التسلسلية Urxc, Utxc, Udre:

```
$regfile = "m8def.dat"
$crystal = 4000000
$baud = 9600
'----------------------
Enable Urxc
Disable Utxc
Disable Udre

On Urxc Getchar
On Utxc Finish
On Udre Empty

Enable Interrupts
'----------------------
Dim Inchar(10) As Byte
Dim Flag As Bit
'----------------------
Do
   nop
Loop Until Flag = 1
End
'----------------------
Getchar:
   Disable Urxc
   Enable Udre
   Enable Utxc

   Inputbin Inchar(1) , 10
   Printbin Inchar(1) ; 10
Return
'----------------------
Finish:
   Disable Utxc : Set Flag
   Print "Serial TX complete interrupt!"
Return
'----------------------
Empty:
   Disable Udre : Print ""
   Print "UART Register is Empty!"
Return
```



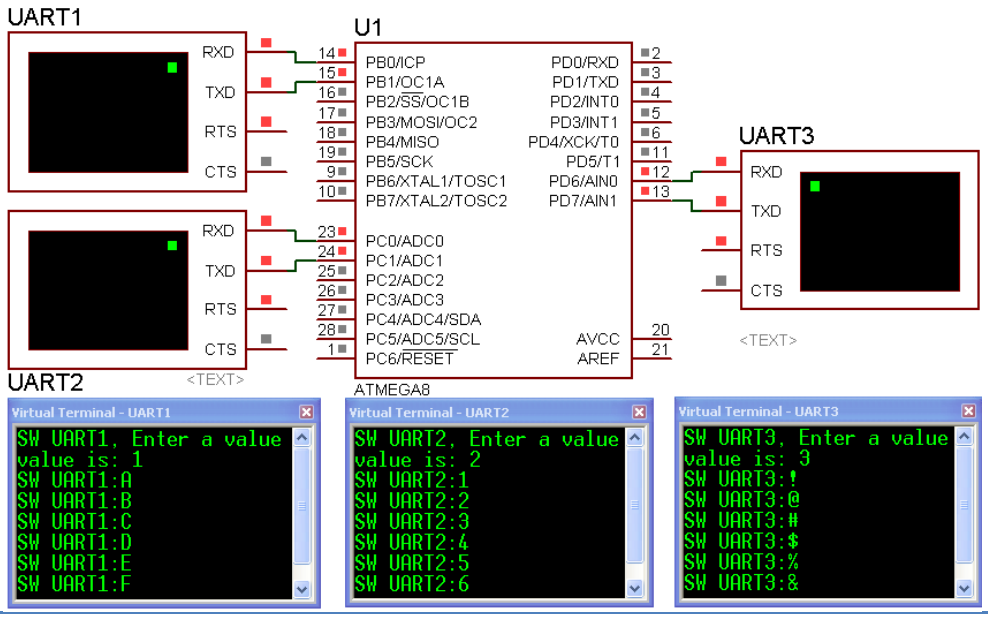
برمجة النافذة التسلسلية UART0, UART1 وتحرير الإعدادات المتقدمة للنافذة:

```
$regfile = "m128def.dat"
$crystal = 4000000
$baud = 9600
$baud1 = 9600
'-------------------------
Config Com1=Dummy, Synchrone =0, Parity = None, Stopbits =1, Databits =8, Clockpol=0
Config Com2=Dummy, Synchrone =0, Parity = None, Stopbits =1, Databits =8, Clockpol=0

Open "com1:" For Binary As #1
Open "com2:" For Binary As #2

Config Serialin = Buffered , Size = 20 , Bytematch = 27
Config Serialin1 = Buffered , Size = 20 , Bytematch = All

Config Serialout = Buffered , Size = 20
Config Serialout1 = Buffered , Size = 20

Enable Interrupts
'-------------------------
Dim Msg As String * 10
'-------------------------
Do
   If Ischarwaiting() = 1 Then
      Input Msg : Print Msg
   End If

   If Ischarwaiting(#2) = 1 Then
      Input #2 , Msg : Print #2 , Msg
   End If
Loop
End
'-------------------------
Serial0charmatch:
   Print "Esc Char!"
Return
'-------------------------
Serial1bytereceived:
   Print #2 , " "
   Print #2 , "We got a Char!"
Return
'-------------------------
Close #1
Close #2
```



النافذة التسلسلية البرمجية:

```
$regfile = "m8def.dat"
$crystal = 4000000
'------------------------
Ucsrb = 0 : Wait 1
Dim Value As Byte
'------------------------
Open "comb.0:9600,8,n,1" For Output As #1
Open "comb.1:9600,8,n,1" For Input As #2

Open "comc.0:9600,8,n,1" For Output As #3
Open "comc.1:9600,8,n,1" For Input As #4

Open "comd.6:9600,8,n,1" For Output As #5
Open "comd.7:9600,8,n,1" For Input As #6
'------------------------
Print #1 , "SW UART1, " ; "Enter a value"
   Input #2 , Value
      Print #1 , "value is: " ; Value
Print #3 , "SW UART2, " ; "Enter a value"
   Input #4 , Value
      Print #3 , "value is: " ; Value
Print #5 , "SW UART3, " ; "Enter a value"
   Input #6 , Value
      Print #5 , "value is: " ; Value
'------------------------
Get # 2 , A : Put # 1 , A
Do
   Value = Inkey(#2)
      If Value > 0 Then Print #1 , "SW UART1:" ; Chr(value)

   Value = Inkey(#4)
      If Value > 0 Then Print #3 , "SW UART2:" ; Chr(value)

   Value = Inkey(#6)
      If Value > 0 Then Print #5 , "SW UART3:" ; Chr(value)

Loop Until Inkey(#2) = 1 Or Inkey(#4) = 1 Or Inkey(#6) = 1
'------------------------
Close #6 : Close #5 : Close #4
Close #3 : Close #2 : Close #1
End
```



برمجة المتحكمات المصغرة

التجارب العملية

الجلسة العاشرة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

WALID BALID, Tuesday, December 15, 2009

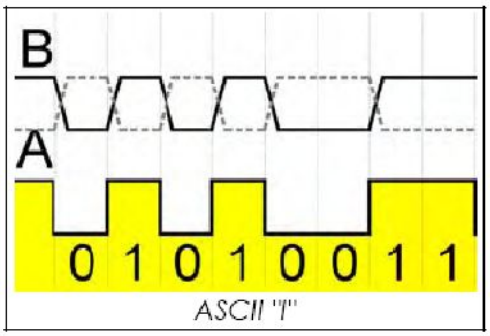
الغاية من التجربة:

دراسة بروتوكول الاتصال التسلسلي RS485 وتطبيقاته في أنظمة التحكم الرقمي.

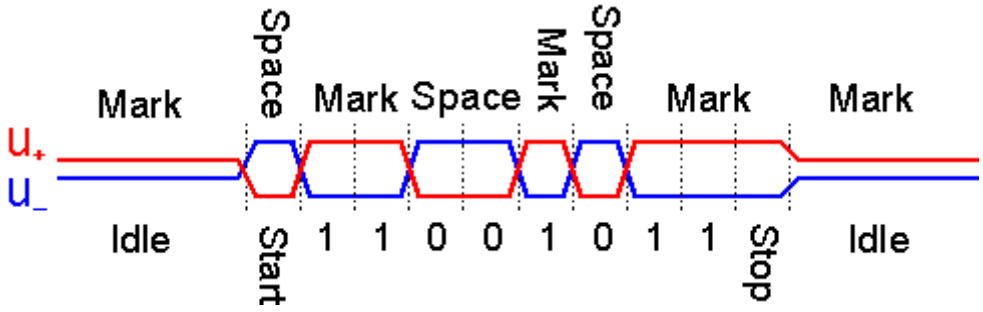
مبدأ عمل البروتوكول RS485:

يعتبر البروتوكول RS485 من أكثر بروتوكولات الاتصال التسلسلي متعدد النقاط انتشاراً في التطبيقات الصناعية في وقتنا الحاضر.

يعتمد هذا البروتوكول على مبدأ فرق الجهد التفاضلي بين خطي النقل (A,B)، حيث أنه في حال كون الجهد VA<VB تكون الحالة المنطقية على خط النقل "1" وفي حال كون الجهد VA>VB تكون الحالة المنطقية على خط النقل "0". في حال عدم وجود بيانات على خط النقل فإن الحالة المنطقية على الخط هي "1".



إن البروتوكول RS485 ذو ممانعة عالية للضجيج وذلك لأنه يعتمد على مبدأ الجهد التفاضلي على الخط لتعيين الحالة المنطقية لكل بت، فمثلاً إن وجود ضجيج على الخط A سوف يحدث ضجيجاً مماثلاً على الخط B، وباعتبار أن هذا البروتوكول يقارن فرق الجهد بين الخطين، فإن أي زيادة مماثلة أو نقصان لن يغير نتيجة المقارنة.



الشركات المصنعة لشرائح RS485:

الجدول التالي يبين الشركات الأساسية المصنعة لدرايفرات البروتوكول RS485.

ST	TI	MAXIM	LTC
ST485	SN75176	MAX485	LTC485

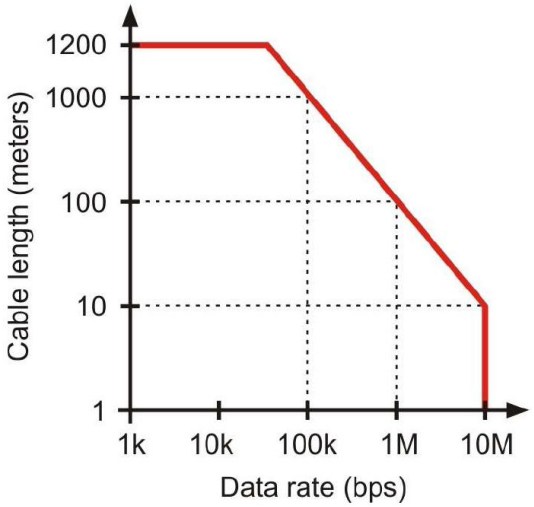
مواصفات البروتوكول RS485:

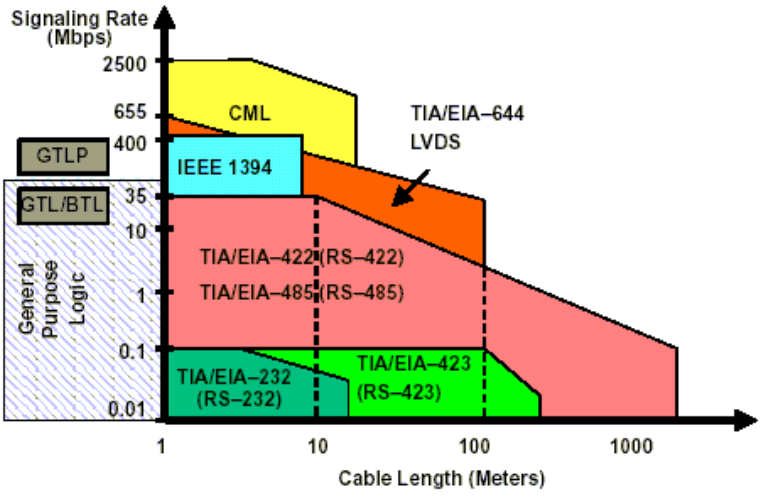
الجدول التالي يبين المواصفات التقنية للبروتوكول RS485 ومقارنة مع برتوكولات مشابهة.

SPECIFICATIONS		RS232	RS423	RS422	RS485
Mode of Operation		SINGLE-ENDED	SINGLE-ENDED	Differential	Differential
Total Number of Drivers and Receivers on One Line		**1**DRIVER **1**RECVR	**1**DRIVER **10**RECVR	**1**DRIVER **10**RECVR	**32**DRIVER **32**RECVR
Maximum Cable Length		50 FT	4000 FT	4000 FT	4000 FT
Maximum Data Rate		20kb/s	100kb/s	10Mb/s 100Kb/s	10Mb/s 100Kb/s
Maximum Driver Output Voltage		±25V	±6V	-0.25V to +6V	-7V to +12V
Driver Output Signal Level (Loaded Min.)	*Loaded*	±5V to ±15V	±3.6V	±2.0V	±1.5V
Driver Output Signal Level (Unloaded Max)	*Unloaded*	±25V	±6V	±6V	±6V
Driver Load Impedance (Ohms)		3k to 7k	>=450	100	54
Max. Driver Current in High Z State	*Power On*	N/A	N/A	N/A	±100uA
Max. Driver Current in High Z State	*Power Off*	±6mA @ ±2v	±100uA	±100uA	±100uA
Slew Rate (Max.)		30V/µS	Adjustable	N/A	N/A
Receiver Input Voltage Range		±15V	±12V	-10V to +10V	-7V to +12V
Receiver Input Sensitivity		±3V	±200mV	±200mV	±200mV
Receiver Input Resistance (Ohms)		3k to 7k	4k min	4k min	>=12k

العلاقة بين معدل النقل وطول خط النقل:

الشكل اليميني يوضح العلاقة بين معدل النقل وطول الكابل لبعض بروتوكولات الاتصال التسلسلي الصناعية، بينما يوضح الشكل اليساري العلاقة بين معدل النقل وطول الكابل للبروتوكول RS485.

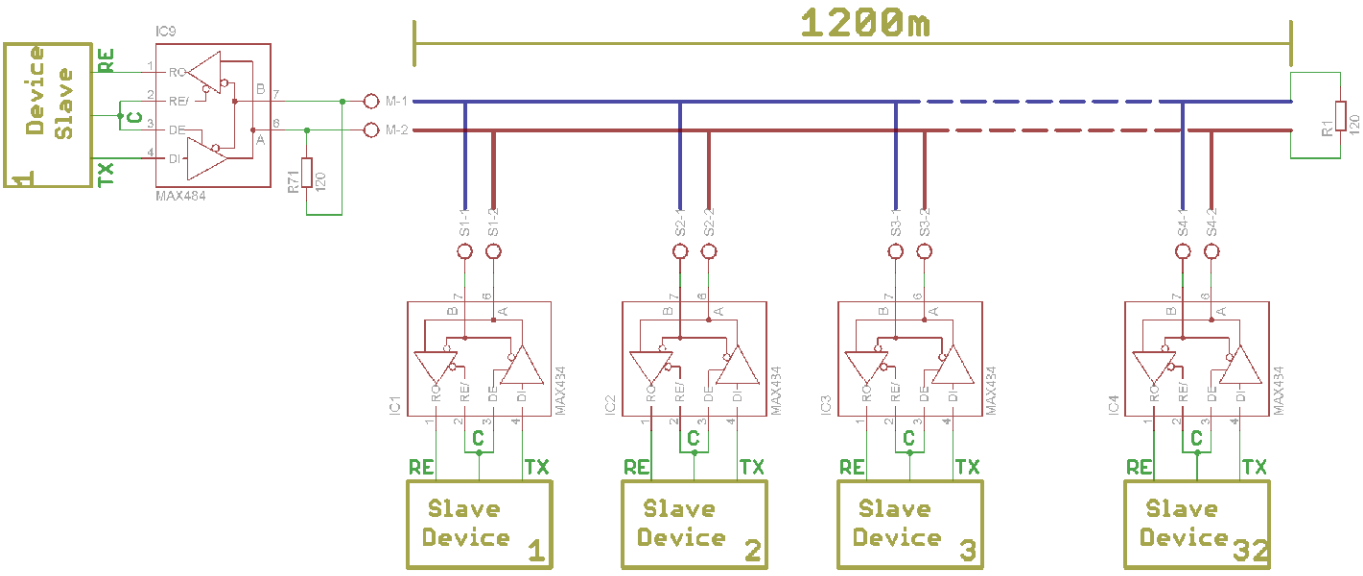




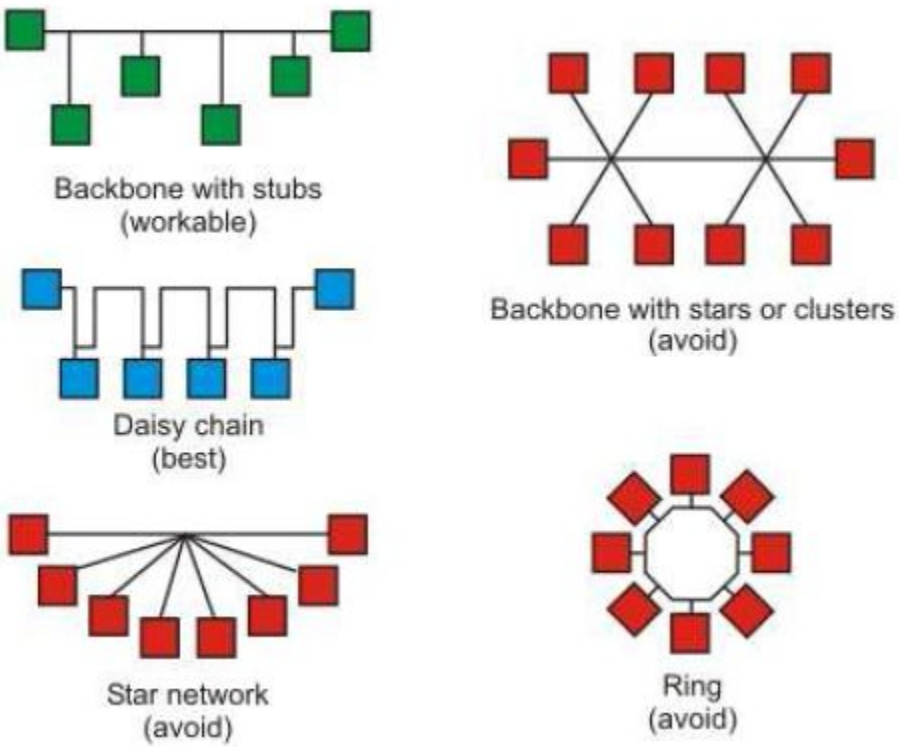
ملاحظة: بشكل عام فإنه بازدياد طول خط النقل يتم تخفيض معدل النقل لتخفيض الضجيج، ويمكن الحصول على مسافة نقل أعظمية عند معدلات نقل منخفضة.

توصيل شبكة RS485 باستخدام خطين:

يتم توصيل الشبكة بحيث يتم وصل جميع النقاط **A** مع بعضها وكذلك جميع النقاط **B**، ويتم وضع مقاومات تحميل طرفية في بداية ونهاية الخط كما في الشكل التالي:



ملاحظة: إن التفريعات الصادرة عن الخط الرئيسي يجب أن لا تتجاوز 15Meter، والأفضل أن تستخدم طريقة الوصل Daisy Chain بدلاً من طريقة الوصل Backbone with Stubs.



ملاحظة هامة: إن الاعتقاد الشائع بأن البروتوكول RS485 يحتاج إلى خطين فقط ولا يحتاج إلى خط تأريض هو اعتقاد خاطئ، فيجب توصيل خط التأريض بين الوحدات أو وصله إلى نقطة التأريض. الجدول التالي مثال عن الكابلات المستخدمة.

Belden P/N	Pairs	AWG	mm	Shield/drain wire	Imp	Cap
9841	1	24	0.6	Yes, 24AWG	120ohm	42pf/m
9842	2	24	0.6	Yes, 24AWG	120ohm	42pf/m
8132	2	28	0.4	Yes, 28AWG	120ohm	36pf/m

التحكم بشبكة RS485 باستخدام خطين:

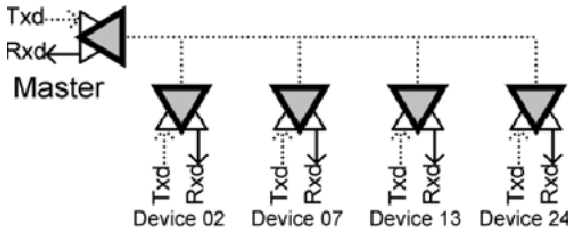
يتم التحكم بهذه الشبكة وفق مبدأين:

- PASSIVE DUPLEX CONTROL (*AUTOMATIC*)
- ACTIVE DUPLEX CONTROL (*RTS Pin*)

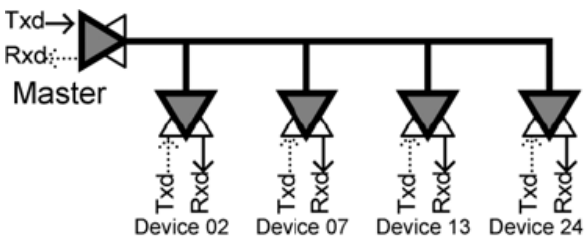
أولاً: نمط التحكم Passive Duplex: في هذه الحالة يستخدم خطين فقط (RO, DI) للتحكم بـ RS485 Driver ولا حاجة لقطب التحكم بالاتجاه (DE/RE)، حيث أن المستقبلات تفحص بشكل دائم خط النقل وعندما تستقبل أمرً من المرسل تقوم بالرد عليه، حيث أن التنسيق يتم برمجياً.

مبدأ العمل:

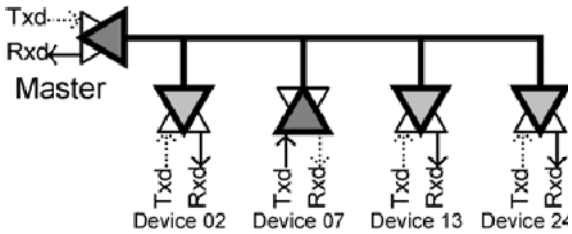
- **لا يوجد بيانات للإرسال** (No data to send): في هذه الحالة تكون الحالة المنطقية على خط النقل هي "1" وجميع الطرفيات الثانوية تكون في حالة انتظار استقبال بيانات من الوحدة الرئيسية للاستقبال (حالة توقف Idle).



- **إرسال أمر من الوحدة الرئيسية** (Master sends a request): تقوم الوحدة الرئيسية بإرسال أمر طلب على الناقل، فتكتشف الوحدات الثانوية بت بدء الإرسال (Start Bit) وتبدأ بقراءة البيانات الموجودة على الناقل، وعندما ينتهي الإرسال من قبل الوحدة الرئيسية ببت التوقف (Stop Bit)، تقوم الوحدات بتفحص الخطأ (Check sum) وفي حال حصوله لا يتم إرسال أي استجابة، وتعود جميع الوحدات إلى نمط الاستقبال ويصبح الناقل في حالة توقف (Idle).



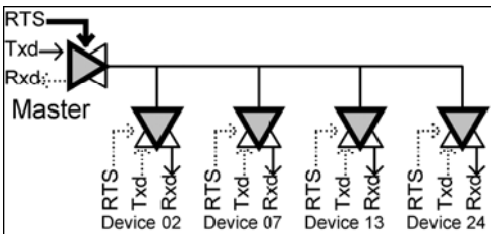
§ **إرسال رد من الوحدة الثانوية** (Slave sends a response): بفرض أن عنوان الجهاز الذي تم طلبه من قبل الوحدة الرئيسية هو add = 07 وبالتالي ستقوم الوحدة الثانوية السابعة بالرد وتتحول إلى وحدة إرسال بنفس الوقت الذي تكون فيه الوحدة الرئيسية في حالة استقبال وانتظار الرد، حالما تنتهي هذه الوحدة من الإرسال تعود إلى حالتها كمستقبل ويصبح خط النقل في حالة توقف من جديد.



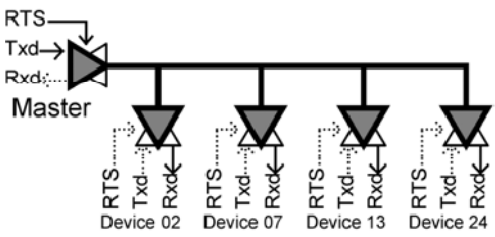
Ü **ثانياً: نمط التحكم Active Duplex:** في هذه الحالة يستخدم ثلاث خطوط للتحكم بشريحة RS485 Driver، حيث يستخدم الخط الثالث (RTS Control) كخط تحكم لتحديد اتجاه وحدة الإرسال/الاستقبال.

مبدأ العمل:

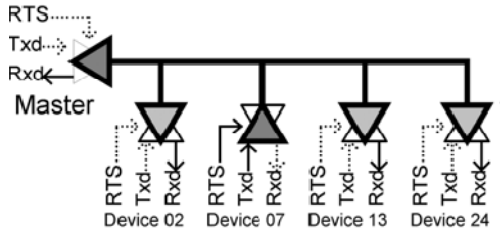
§ **لا يوجد بيانات للإرسال** (No data to send): في هذه الحالة تقوم الوحدة الرئيسية بتفعيل إشارة خط التحكم بالاتجاه الخاص بها (RTS) مما يجبر خط النقل إلى التحول إلى حالة التوقف (الحالة المنطقية على خط النقل هي "1"). كل إشارات التحكم بالاتجاه الخاصة الوحدات الثانوية تكون غير مفعلة والوحدات في حالة ترقب.



§ **إرسال أمر من الوحدة الرئيسية** (Master sends a request): تقوم الوحدة الرئيسية بإرسال أمر طلب على الناقل فتكتشف الوحدات الثانوية بت بدء الإرسال (Start Bit) وتبدأ بقراءة البيانات الموجودة على الناقل، وعندما ينتهي الإرسال من قبل الوحدة الرئيسية ببت التوقف (Stop Bit) تقوم الوحدات بتفحص الخطأ (Check sum) وفي حال حصوله لا يتم إرسال أي استجابة، وبعدها تقوم الوحدة الرئيسية بإلغاء تفعيل خط التحكم بالاتجاه وتعود جميع الوحدات إلى نمط الاستقبال ويصبح الناقل في حالة توقف من جديد (Idle).



إرسال رد من الوحدة الثانوية (Slave sends a response): بفرض أن عنوان الجهاز الذي تم طلبه من قبل الوحدة الرئيسية هو add = 07 وبالتالي ستقوم الوحدة الثانوية السابعة بتفعيل خط التحكم بالاتجاه الخاص بها وتتحول إلى وحدة إرسال بنفس الوقت الذي تكون فيه الوحدة الرئيسية في حالة استقبال وانتظار الرد، وحالما تنتهي هذه الوحدة من الإرسال تقوم بإلغاء تفعيل خط التحكم بالاتجاه الخاص بها وتعود إلى حالتها كمستقبل، حيث أن وحدة التحكم الرئيسية تقوم بإعادة تفعيل خط التحكم بالاتجاه الخاص بها حالما تدرك أنه تم انتهاء الإرسال بالتعرف على بت التوقف.



الميزات والمساوئ لبرتوكول الاتصال RS232:

المساوئ (Disadvantages)	المحاسن (Advantages)
× مناسب فقط من أجل الربط بين System-to-System أكثر من كونه قابلاً للربط بين Chip-2-Chip أو من أجل تطبيقات Chip-2-Sensor. × يحتاج إلى وحدة تبديل مستوى منطقي TTL<>RS485، بالإضافة إلى كابل مزدوج ملفوف ومقاومات طرفية مما يزيد من كلفة النظام.	ü بروتوكول اتصال شائع الاستخدام في كثير من التطبيقات ومعتمد من قبل العديد من الشركات. ü مسافة الاتصال طويلة جداً حوالي 1200 متر عند معدل إرسال 100kb/s. ü مناعة ضد الضجيج بسبب الجهد التفاضلي. ü سهل البناء والبرمجة ومتوفر برمجياً وكعيان صلب.

التخاطب باستخدام البنية البرمجية MODBUS وتطبيقاتها في الاتصالات التسلسلية RS485:

يعتبر التشكيل MODEBUS لرسالة البيانات أحد أهم البروتوكولات البرمجية المستخدمة في الاتصالات التسلسلية متعددة النقاط (Multi Master/Slave)، حيث أن هذا التشكيل لرسالة المعلومات المرسلة أو المستقبلة يتيح مرونة كبيرة في تحديد الجهاز المراد التخاطب معه وتعيين الوظيفة التي سيقوم بها، بالإضافة لكشف حدوث الأخطاء.

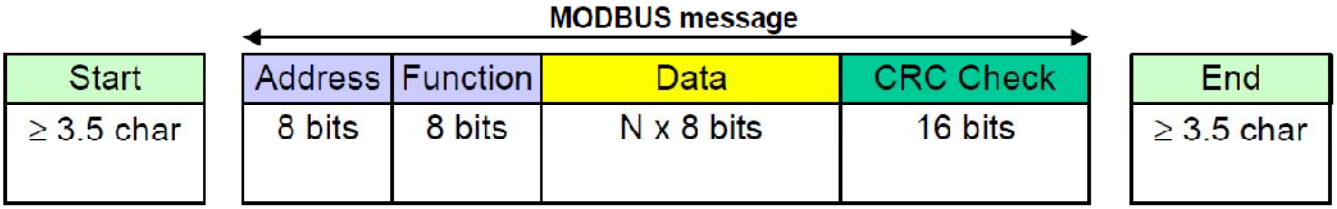
يوجد تشكيلين متقاربين إلى حد ما للبروتوكول البرمجي MODEBUS، أحدهما للقائد (Master Device) والآخر للمقاد (Slave Device).

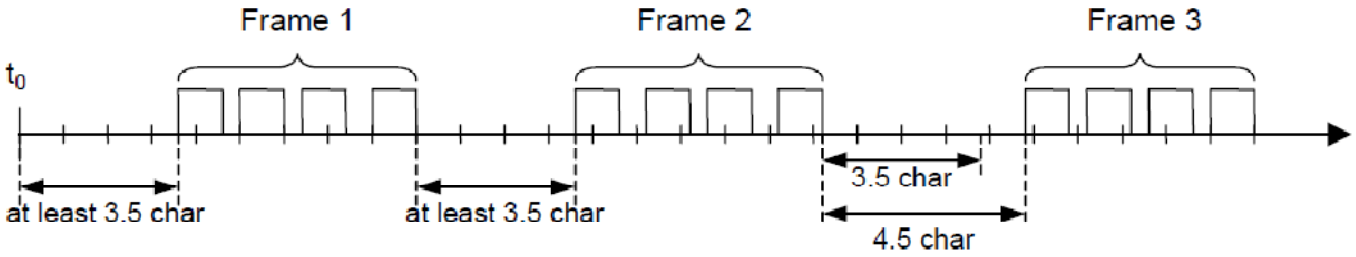
التشكيل MODEBUS لرسالة الاستعلام (Query) المرسلة من القائد (Master Device) للمقاد (Slave Device):

Slave Address	Function Code	Byte Count	Start Address	Data Stream	CRC CRC Low	CRC Hi
1 byte	1 byte	1 byte	2 byte	0 ~ 252 byte	1~2 byte	

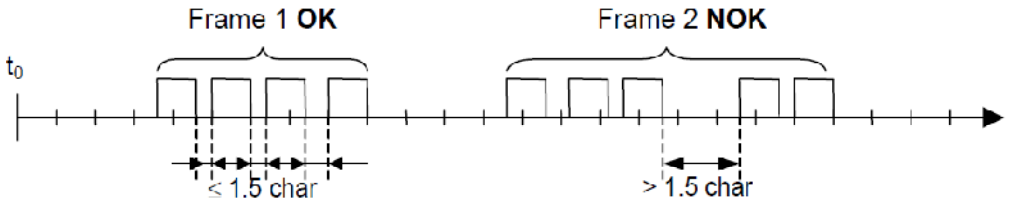
- Slave Address: يحدد عنوان الوحدة الطرفية الثانوية (1~252).
- Function code: يحدد الوظيفة المطلوبة من الإرسال.
- Byte Count: عدد البايتات التي سيتم إرسالها.
- Start Address: عنوان البايت الأول في ذاكرة المستقبل.
- Data Stream: وتحوي على حزمة البيانات المرسلة متضمنة العنوان ($255\ byte_{max}$).
- CRC (Cyclic Redundancy Check): كود تفحص خطأ الإرسال.

في حال أريد إرسال حزمتي بيانات متلاحقة، فإنه يجب أن يفصل بينهما بزمن يسمى Inter-frame-time وهو يساوي $t_{3.5}$ بناءً على الشكل التالي:





في حال إرسال عدد كبير من البايتات (Data) فيجب أن لا تكون الفترة الفاصلة بين إرسال محرفين أكثر من $t_{1.5}$، وإلا فعلى المستقبل أن يلغي العملية ويشير إلى حصول خطأ!



إن كشف هذا التأخير أو تعيينه عند سرعات نقل عالية يصبح أمراً صعباً جداً على المعالج، لذلك يتم يتعين قيم افتراضية ثابتة وهي:

Inter-character time-out (t1.5) = 750µs
Inter-frame delay (t3.5) = 1.750ms

بافتراض أن معدل الإرسال لن يتجاوز 9600bps، كما أن عدد البايتات الأعظمي في كل حزمة لن يتجاوز 70_{Byte}، وبالتالي يمكن افتراض فترة تأخير زمني ثابتة بين حزم البيانات المتلاحقة، كما أن هذا الزمن هو زمن تعطيل الاستقبال للنافذة التسلسلية للوحدات الثانوية الغير معنية بالعنوان Slave address.

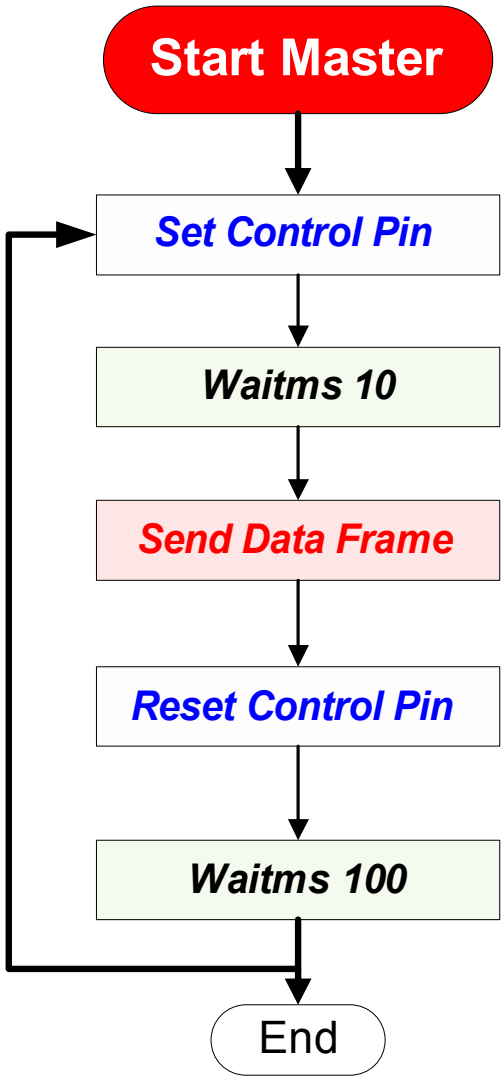
يمكن حساب زمن الإرسال تبعاً لمعدل نقل البيانات بالشكل التالي:

$$70_{Byte} \times 8_{Bit} = 560_{Bit}$$

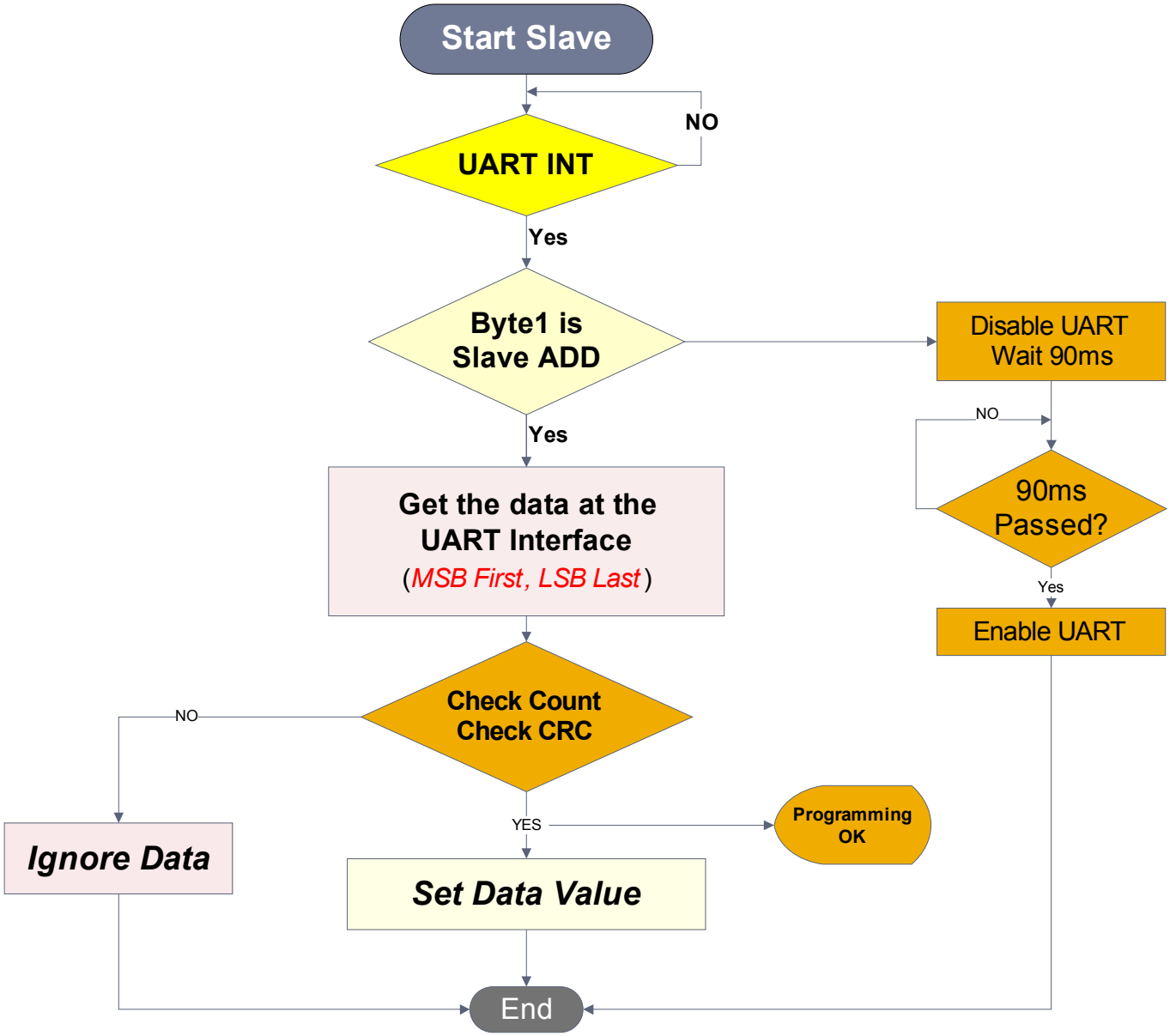
$$\frac{560}{9600} \times 1000 = \mathbf{58.3\ ms}$$

وبالتالي فإن الوحدات الثانوية الغير معنية يجب أن تعطل الاستقبال زمناً أكبر من 58ms وبعدها تعود لوضع الاستقبال، وأيضاً يجب على الوحدة الرئيسية القائدة أن لا ترسل أي حزمة جديدة إلا بعد انقضاء زمن الإرسال مضافاً إليه الزمن 1.75ms =t3.5.

الشكل التالي مخطط خوارزمية الإرسال من الوحدة الرئيسية.



الشكل التالي مخطط خوارزمية الاستقبال للوحدات الفرعية.



التشكيل MODEBUS لرسالة الاستجابة (Responce) المرسلة من المقاد (Slave) للقائد (Master):

بعد أن يقوم القائد بإرسال رسالة الاستعلام، يقوم المقاد المعني بالعنوان Slave بالرد على رسالة الاستعلام، وبالتالي فإن رسالة الرد تقتصر على تأكيد الرسالة أو ربما يتوجب على المقاد إرسال بيانات مطلوبة منه، وبالتالي فإن تشكيل رسالة الاستجابة يمكن أن يكون له عدة أشكال تبعاً لعمل النظام.

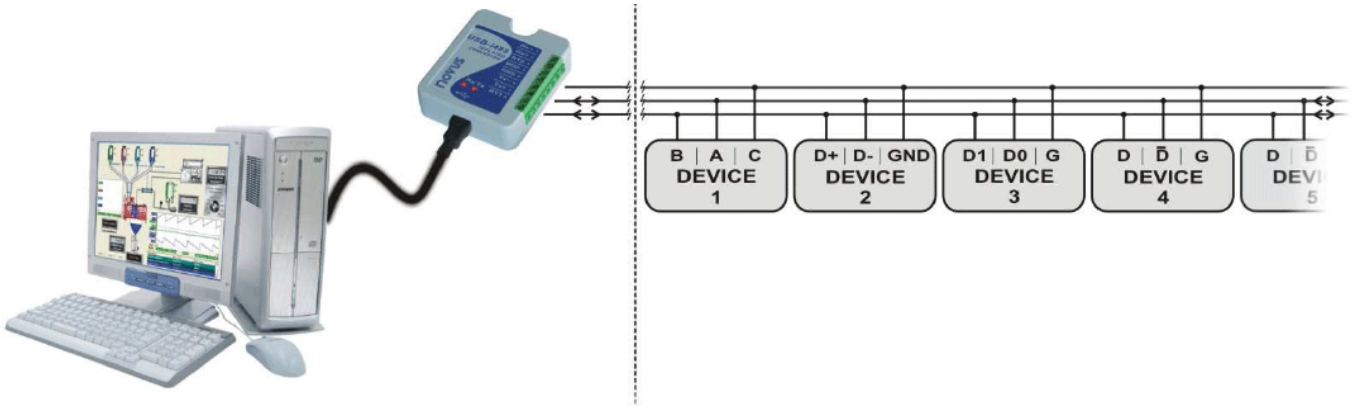
Slave Address	Function Code	Byte Count	CRC	
			CRC Low	CRC Hi
1 byte	1 byte	1 byte	1~2 byte	

مثلاً: في رسالة الرد أعلاه يقوم المقاد بإرسال عنوانه (Slave) والوظيفة التي استلمها وعدد البايتات التي تلقاها ويحسب قيمة الـ CRC.

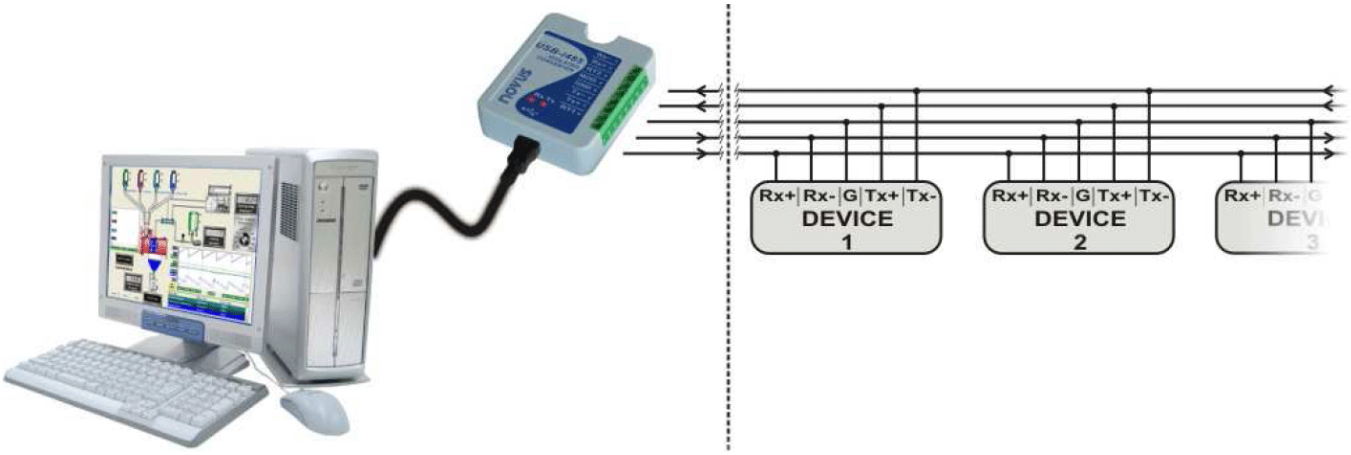
نمط الإرسال أحادي | ثنائي الاتجاه للبروتوكول RS485:

إن البروتوكول RS485 يمكن أن يعمل في نمطين: أحادي الاتجاه (Half-Duplex) وثنائي الاتجاه (Full-Duplex)، وقد تم شرح المبدأ في المحاضرة السابقة.

الشكل التالي توصيل شبكة أحادية الاتجاه.



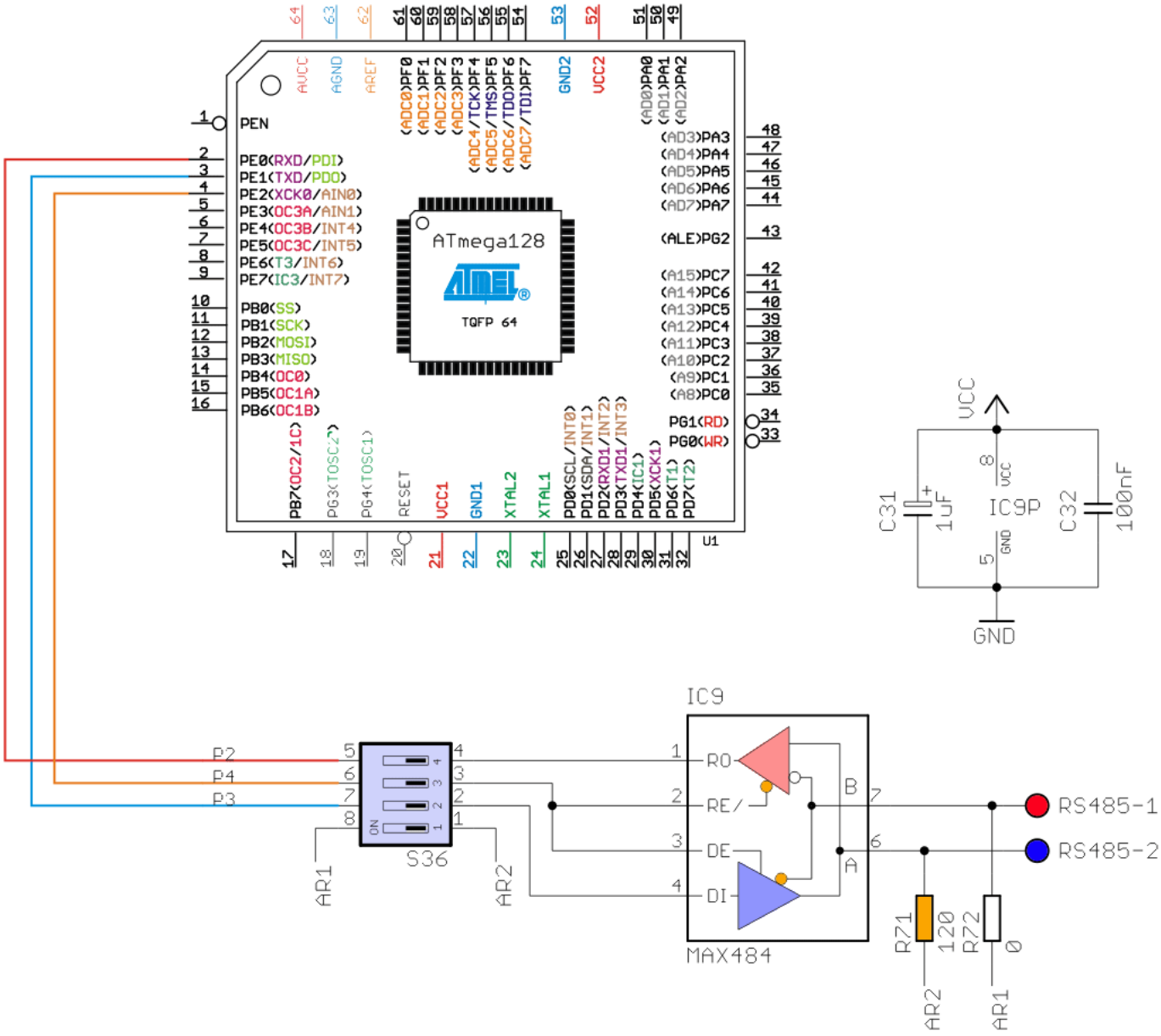
الشكل التالي توصيل شبكة ثنائية الاتجاه.



إن الشبكات RS485 أحادية الاتجاه أكثر شيوعاً، ولكن في بعض التطبيقات التي تتطلب تحكماً ومراقبةً للنظام في الزمن الحقيقي تستخدم الشبكات ثنائية الاتجاه.

مخطط دارة الاستقبال والإرسال عن طريق المتحكم المصغر (UART<>RS458):

الشكل التالي توصيل متحكم مصغر مع الدارة المتكاملة Max485.



التعليمات الجديدة:

شرح التعليمة	التعليمة
تعريف قطب التحكم بالاتجاه لدارة المحول RS485.	`Config Print0 = Porte.2 , Mode = Set`

ملاحظة: إن التعليمة السابقة (`Config Print0`) سوف تقوم بتطبيق "1" على القطب "DE" عند طلب الإرسال باستخدام التعليمة "DE"، ومن ثم تعيد حالة القطب المذكور إلى "0" فور انتهاء عملية الإرسال.

يمكن إجراء عملية التحكم على القطب "DE" بشكل يدوي والاستغناء على التعليمة السابقة وخصوصاً عندما يراد أن تكون الحالة المنطقية عائمة!

برنامج تشغيل الدارة كقائد (Microcontroller is Master, PC is Slave):

التوجيهات.

تعريف البوابة الموصل معها قطب التحكم DE.

تعريف المتحولات

```
$regfile = "m128def.dat"
$crystal = 8000000
$baud = 9600
'------------------
Config Print0 = Porte.2 , Mode = Set
Config Pine.2 = Output

Config Pine.4 = Input
Key1 Alias Pine.4
Porte.4 = 1
'------------------
Dim Msg As String * 100
'------------------
Do
   Debounce Key1 , 0 , Sendall , Sub
Loop
End
'------------------
Sendall:
   Print "This is RS485 Test Program"
   Print "RS458 Protocol Done Based-on MAX485"
   Print "University of Aleppo - Syria"
   Print "Faculty of Alectrical & Electronic Engineering"
   Print "Control Department"
   Print "Fourth Year Students"
   Print "Computer Aided Design Session"
   Print "Unfutunality This Was The Last Session Lab"
   Print "2nd Semester"
   Print "See You Next Semester Guys ;)"
   Print "Kindest Regards"
   Print "Walid BALID"
   Print ":) :) :) :) :) :) :) :) :) :) :) :)"
   Wait 1
Return
```

برنامج تشغيل الدارة كمقاد (Microcontroller is Slave, PC is Master):

التوجيهات.	`$regfile = "m128def.dat"`
	`$crystal = 8000000`
	`$baud = 9600`

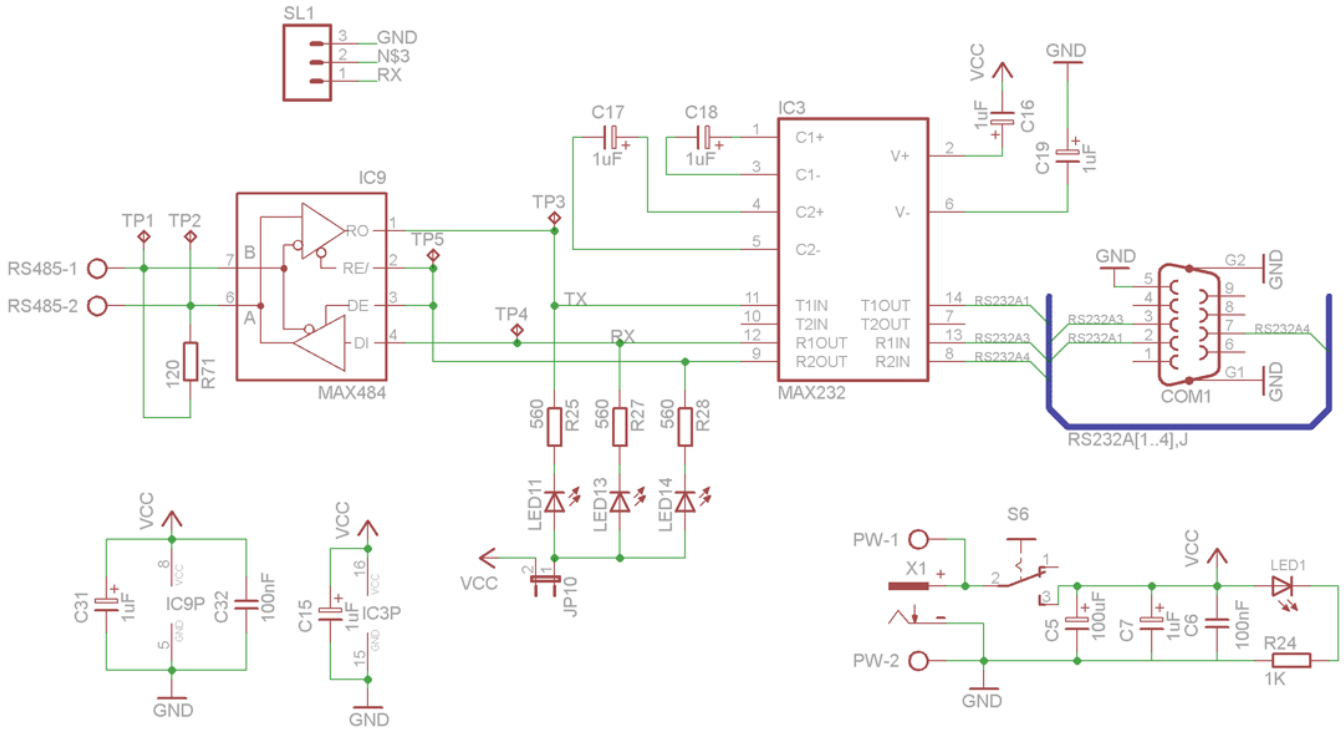
تعريف البوابة الموصل معها قطب التحكم DE.	`'-------------------------`
	`Config Porte.2 = Output`
	`Config Print = Porte.2 , Mode = Reset`

تعريف نافذة تسلسلية UART برمجية.	`Open "comd.3:9600,8,n,1" For Output As #1` `Open "comd.2:9600,8,n,1" For Input As #2`
تعريف مسجل بفر لدخل النافذة التسلسلية	`Config Serialin = Buffered , Size = 250` `Enable Interrupts`
تعريف المتحولات	`'-------------------------` `Dim Msg As String * 250`

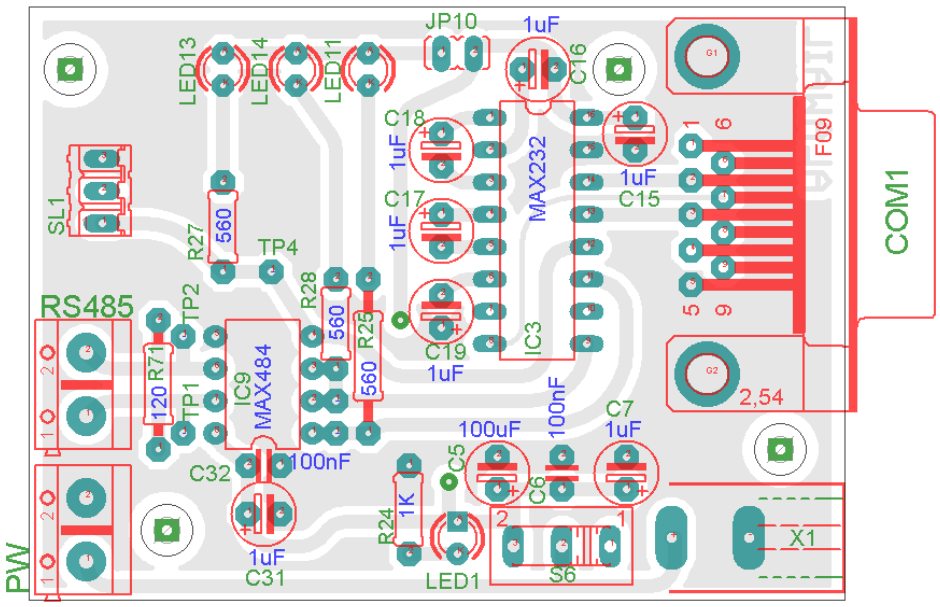
حلقة البرنامج الرئيسي يتم فيها:
فحص مسجل دخل النافذة UART الرئيسية.
قراءة وطباعة جميع البيانات الواردة على النافذة UART البرمجية.
	`'-------------------------`
	`Do`
	`  If Ischarwaiting() = 1 Then`
	`  Input Msg`
	`  Print #1 , Msg`
	`  End If`
	`Loop`
	`End`

مخطط دارة الاستقبال والإرسال عن طريق الحاسب (RS232<>RS458):

الشكل التالي المخطط النظري لدارة تحويل RS458<>RS232.

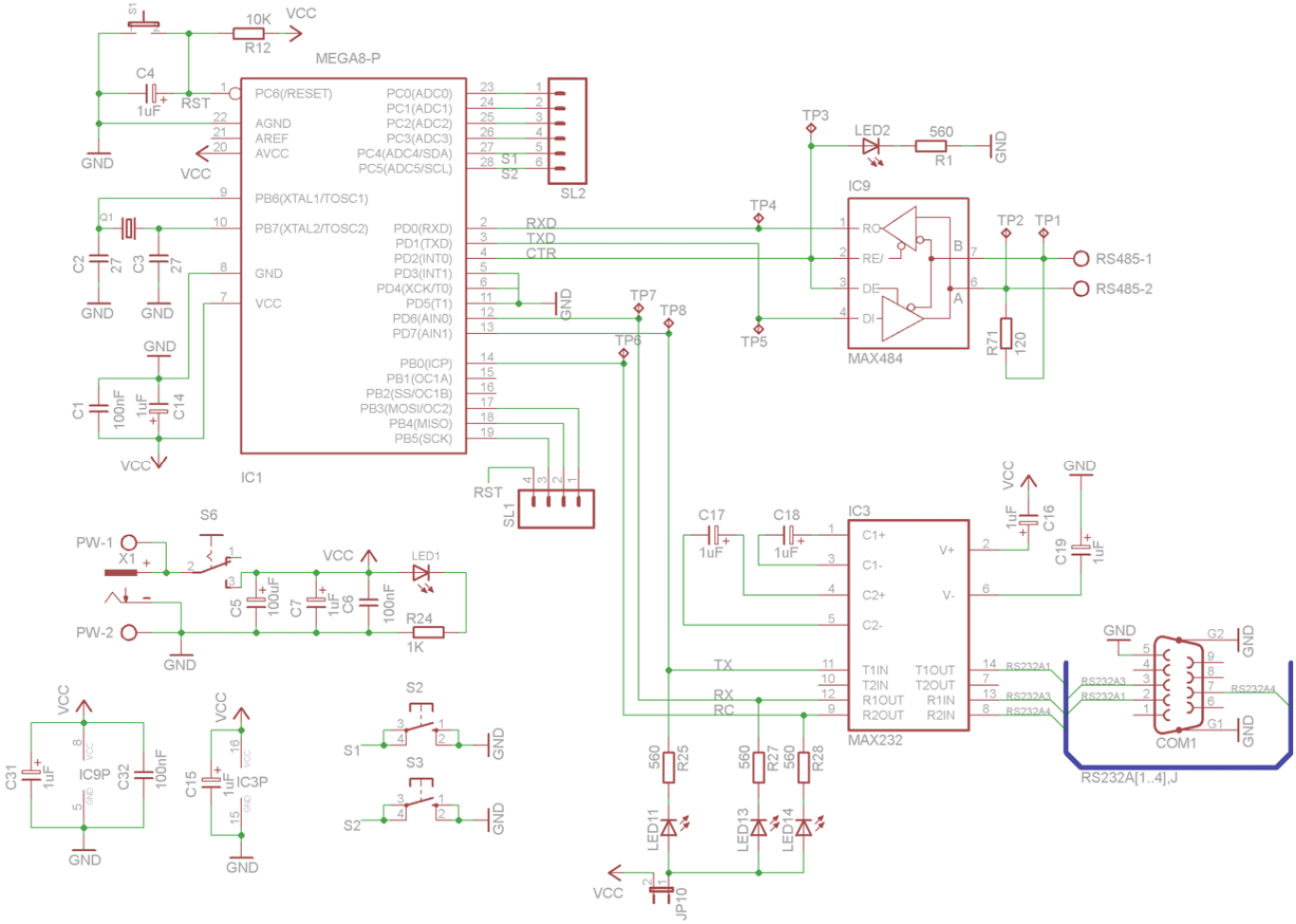


الشكل التالي مخطط الدارة المطبوعة وتوزع العناصر لدارة تحويل RS458<>RS232.

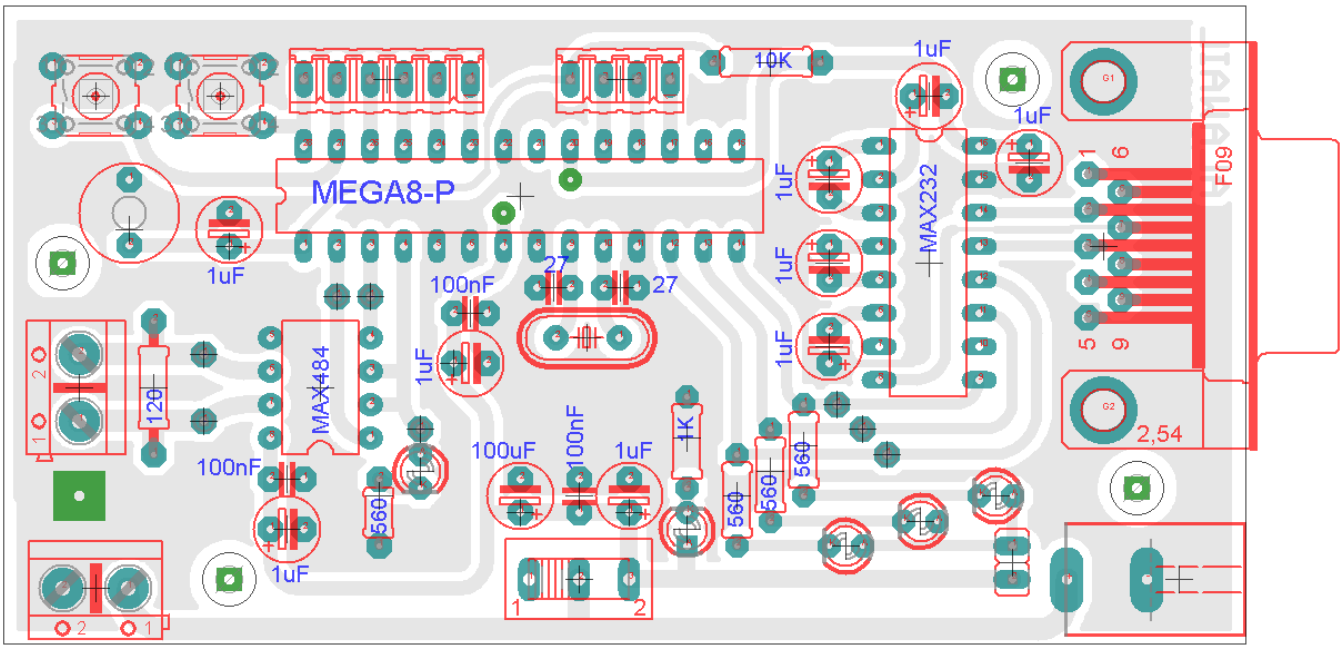


مخطط لوحة اختبار لدارة استقبال وإرسال عن طريق الحاسب ومتحكم (RS232<>UART<>RS458):

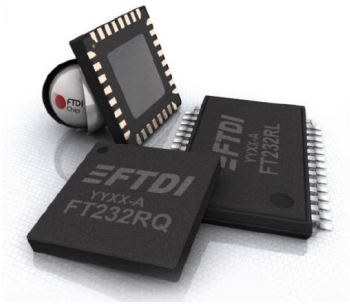
الشكل التالي المخطط النظري لدارة تحويل RS232<>UART<>RS458. في هذه الدارة يتعامل المتحكم مع الشريحة Max485 الموصولة مع النافذة UART الأساسية، ويتعامل بنفس الوقت مع الشريحة Max232 الموصولة مع نافذة UART برمجية، وبالتالي يمكن للمتحكم أن يكون وسيطاً بين البروتوكولين.



الشكل التالي مخطط الدارة المطبوعة وتوزع العناصر لدارة التحويل RS232<>UART<>RS458.



التجربة التاسعة والعشرون: البروتوكول USB Exp.29:Serial Communications USB



الغاية من التجربة:

برمجة واستخدام بروتوكول الاتصال التسلسلي USB في أنظمة التحكم الرقمي.

شرح عمل الدارة:

إن بنية البروتوكول USB شديدة التعقيد، لذلك لن نناقش بالتفصيل في هذه العجالة مبدأ عمل البروتوكول USB وإنما سوف نقدم إحدى طرق استثمار المنفذ لأغراض الربط مع الحاسب.

ميزات منفذ الاتصالات التسلسلي USB:

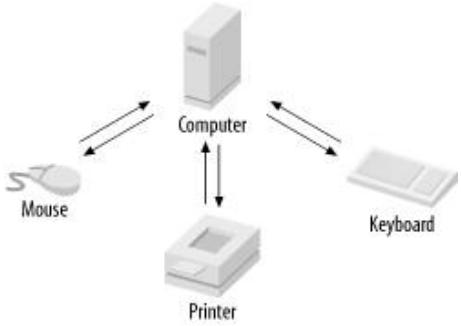
- السرعة العالية التي تصل إلى 480Mb/s.
- دعمه لتقنية Plug & Play حيث يمكن وصل الجهاز دون الحاجة لإعادة إقلاع الجهاز.
- عدد أقل من الخطوط، لأن التقنية تسلسلية تستخدم فقط أربع خطوط.
- إمكانية ربط عدة طرفيات حتى 127 طرفية على نفس الممر.

معايير البروتوكول USB:

يوجد هناك عدة معايير للبروتوكول USB: USB 3.0 & USB 2.0 & USB 1.1 & USB 1.0 تدعم أربع سرعات:

- السرعة المنخفضة Low Speed بسرعة 1.5 Mbits/s في الإصدار USB 1.0.
- السرعة الكاملة Full Speed بسرعة 12 Mbits/s في الإصدار USB 1.1.
- السرعة العالية High Speed بسرعة 480 Mbits/s في الإصدار USB 2.0.
- الجيل الجديد بسرعات عالية جداً 4800 Mbits/s في الإصدار USB 3.0.

بنية ممر USB:



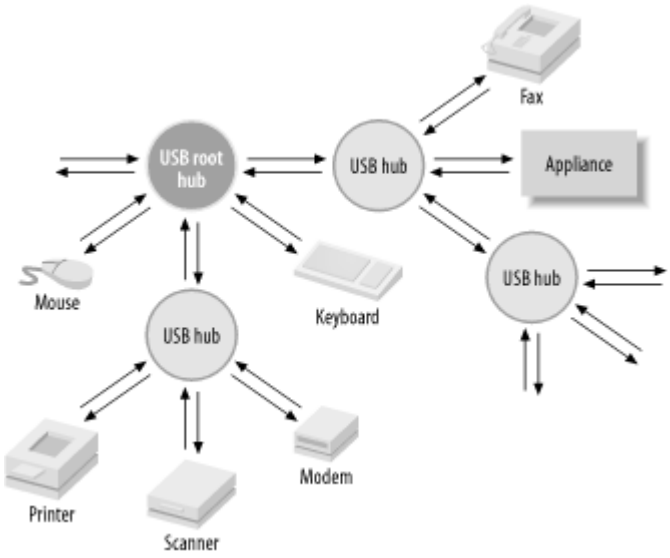
يتصف ممر USB بأنه "Host Controlled" أي يوجد هناك جهاز مضيف واحد على الممر (وهو عبارة عن جهاز يتحكم بالممر وببقية الأجهزة الموصولة معه أثناء العمل) يقع على عاتقه عدة مسؤوليات مثل تهيئة عمليات النقل وتوزيع عرض الحزمة، ولا تسمح تقنية USB بوجود أكثر

من جهاز مضيف واحد على الممر، إلا أنه صدر معيار جديد يدعى"On-The-Go" يسمح بإمكانية وجود جهازين يتم التفاوض فيما بينهما لتبادل دور الجهاز المضيف.

يتمتع ممر USB ببنية نجمية مركزها الجهاز المضيف، تتصف هذه البنية بعدة مزايا مثل:

- إمكانية مراقبة تغذية كل طرفية على حدى.
- إمكانية إطفاء الطرفية في حال استجرارها لتيار كبير بدون أن تتأثر بقية الطرفيات.

لوصل أكثر من جهاز إلى منفذ USB واحد نقوم بوصل تلك الأجهزة إلى موزع HUB والذي بدوره يوصل إلى هذا المنفذ، ويتوفر حالياً العديد من الطرفيات التي تحوي على موزعات مدمجة فيها مثل الشاشات و لوحات المفاتيح.



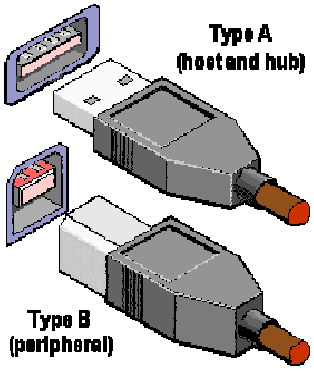
تقنية Plug& Play:

يدعم ممر USB تقنية Plug & Play مع إمكانية تحميل وإزالة ديناميكية لبرنامج القيادة، حيث يقوم المستخدم بوصل الطرفية على ممر USB فيقوم الجهاز المضيف بكشف اتصال هذه الطرفية، و يقوم بالبحث عن برنامج القيادة المناسب لهذا الجهاز ومن ثم يقوم بتنصيبه. إن حجز موارد المقاطعة وعناوين المنافذ يتم بشكل آلي، وكل هذا بدون الحاجة لإعادة إقلاع الحاسب.

بمجرد أن يقوم المستخدم بإزالة هذه فإن الجهاز المضيف سيكشف غياب هذه الطرفية ويزيل برنامج التعريف الخاص بها. تتم عملية تحميل برنامج القيادة المناسب باستخدام رقمي VID, PID .

يدعم ممر USB عدة أنواع من أنماط النقل يتميز كل نمط عن الآخر بإمكانية كشف الأخطاء و تصحيحها، وعرض الحزمة المخصص، و زمن الوصول فيما إذا كان ثابت أم لا.

المواصفات الميكانيكية والكهربائية لمنفذ USB:

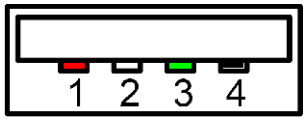
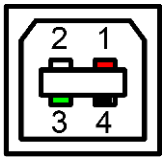


تحدد معايير USB نوعين من المآخذ :

- مأخذ نوع A: يوجد على الحاسب المضيف.
- مأخذ نوع B: يوجد على الطرفية.

يمنع هذا الاختلاف في الشكل بين المأخذين من وصل طرفيتين أو جهازي حاسب مع بعضهما البعض.

يستخدم ممر USB أربع خطوط: خطي تغذية GND & +5v، خطي معطيات D+ & D-.

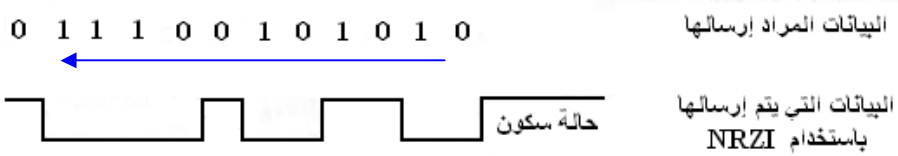
1 VCC (أحمر) 2 D- (أبيض) 3 D+ (أخضر) 4 GND (أسود)		

يستخدم ممر USB الخطين D+&D- لإرسال البيانات بشكل تفاضلي، فمثلاً لإرسال '1' منطقي بشكل تفاضلي على الممر يجب وضع '1' منطقي على القطب D+، '0' منطقي على القطب D- وذلك في حال السرعة الكاملة والمنخفضة للممر، وينعكس هذا التشفير في حال السرعة العالية.

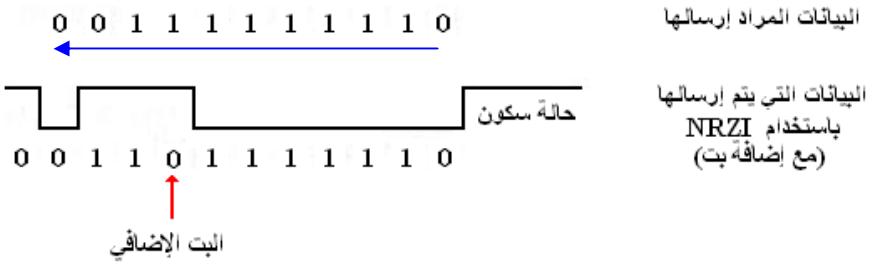
تقنية التشفير NRZI (Non Return to Zero Invert):

يستخدم ممر USB تقنية NRZI لتشفير البيانات المرسلة على الممر للحماية من الضجيج و للمحافظة على التواقت بين المرسل والمستقبل

تعتمد هذه التقنية على تغيير حالة الممر من '0' إلى '1' أو بالعكس عندما يراد إرسال '0' أما إذا كان البت المراد إرساله هو '1' فتبقى حالة الممر كما هي.



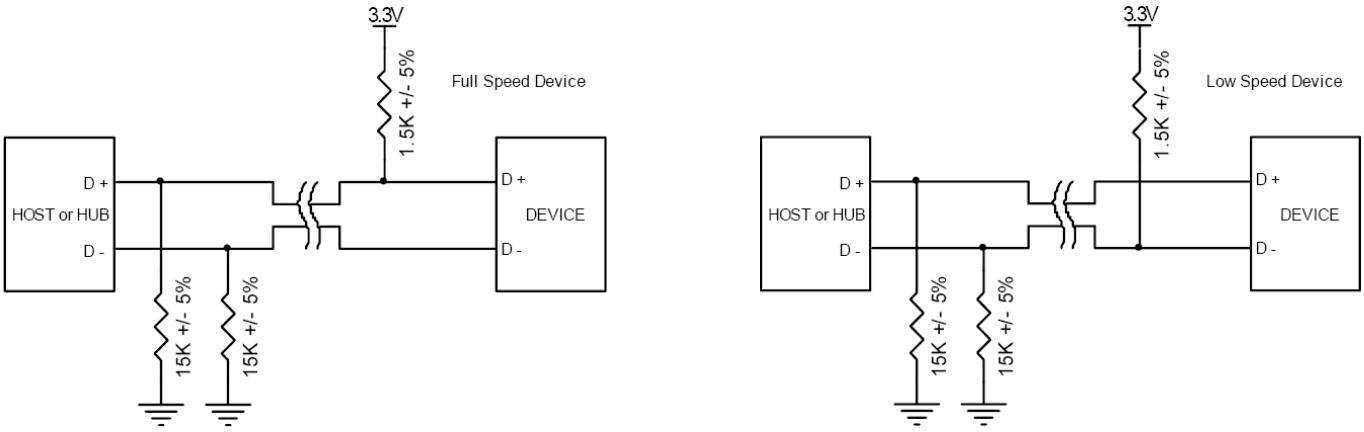
عندما يراد إرسال أكثر من ستة خانات بقيمة '1' منطقي فإنه يتم الفصل بين كل ستة من هذه الخانات بخانة إضافية تحمل القيمة صفر، و ذلك لضمان استمرار عملية التزامن بين المرسل و المستقبل.



تحديد سرعة الطرفية:

يتمكن الحاسب المضيف من تحديد سرعة الطرفية المربوطة بممر USB عن طريق مقاومة شد تربط إلى أحد قطبي البيانات D+ D- في الطرفية، ونستنتج الحالات التالية:

1. الطرفية تدعم السرعة المنخفضة: في هذه الحالة تربط هذه المقاومة إلى القطب D-.
2. الطرفية تدعم السرعة الكاملة: في هذه الحالة تربط هذه المقاومة إلى القطب D+.
3. الطرفية تدعم السرعة العالية: تكون طريقة الربط مماثلة لحالة السرعة الكاملة.



بنية البروتوكول USB:

يحوي برتوكول USB على العديد من المصطلحات الجديدة المرتبطة به، ولذلك لابد لنا من التعرف على هذه المصطلحات قبل الخوض في شرح بروتوكول ممر USB.

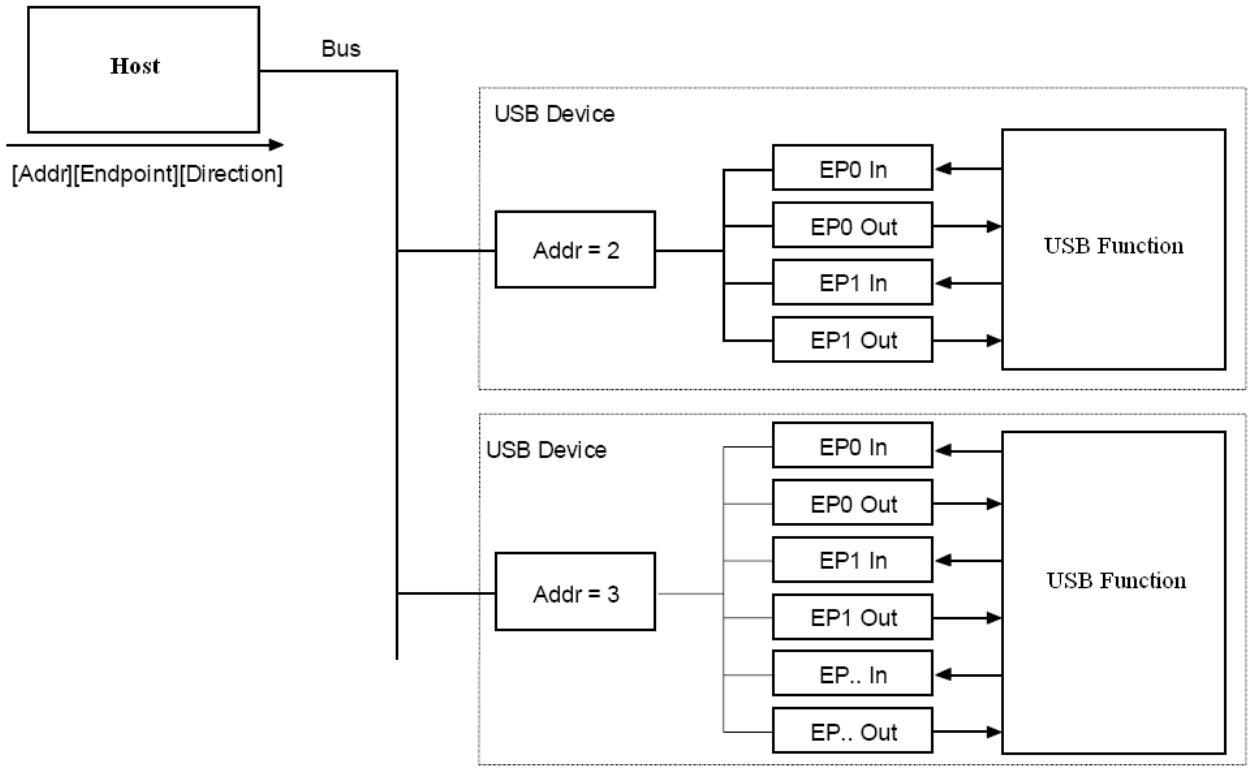
جهاز USB (USB Device): هو تعبير عام قد يشير إلى وحدة طرفية أو حاسب مضيف أو موزع أو إلى دارة متحكم USB في المضيف (Host Controller IC) ولتجنب هذه العمومية تم استخدام مصطلح (USB Function) حيث يقوم المضيف بتخصيص عنوان فريد لكل جهاز USB أثناء عملية التهيئة.

USB Function: عبارة عن مصطلح يشير إلى جهاز USB يقوم بوظيفة معينة مثل الطابعة أو الماسح الضوئي.

النقطة النهائية (Endpoint): يحوي USB Functionعلى عدد من المسجلات التي تمثل صلة الوصل بين الجهاز المضيف وبين USB Function، يدعى كل مسجل من هذه المسجلات بالنقطة النهائية (Endpoint)، و تكون هذه النقاط إما كمصدر للبيانات (Out) أو كمتلقي لها (In) وذلك من وجهة نظر المضيف.

تملك كل نقطة نهائية عنوان فريد ضمن جهاز الـ USB نفسه، و هذا العنوان هو عبارة عن رقم هذه النقطة ونوعها، واختصاراً نرمز للنقطة النهائية الثانية مثلاً من النوع متلقي بالرمز EP2 In.

إن كل عملية اتصال تنشأ بين USB Function والمضيف تكون عملياً بين أحد النقاط النهائية والمضيف كما هو موضح في الشكل التالي:



على سبيل المثال يتمكن المضيف من الوصول إلى النقطة النهائية EP1 In في جهاز USB1 بإرسال عنوان جهاز USB1 ورقم النقطة النهائية ونوعها كما يلي:

[Addr : 2] [Endpoint : EP1] [In]

Pipe: يصطلح على تسمية قناة الاتصال بين أحد النقاط النهائية والمضيف بـ Pipe والذي يمتلك عدة خواص منها: عرض الحزمة المخصص له، نوع البيانات التي ستنقل عليه (تحكم، مقاطعة، كتلية)، اتجاه حركة البيانات، حجم رزمة البيانات الأعظمي. فمثلاً: Pipe الافتراضي هو Pipe0 ثنائي الاتجاه ويتألف من: EP0 In & EP0 Out وبيانات النقل هي للتحكم.

أنواع النقاط النهائية:

يرتبط مفهوم نوع النقاط النهائية بمفهوم نوع النقل الذي يتم عبر Pipe لذلك فإن أنواع النقاط النهائية هي نفسها أنواع النقل والتي تقسم إلى:

- عمليات النقل الخاصة بالتحكم (Control Transfers).

- عمليات النقل الخاصة بالمقاطعة (Interrupt Transfers).
- عمليات النقل التي تتم بشكل دوري وبزمن ثابت (Isochronous Transfers).
- عمليات النقل الخاصة بنقل الكتل (Bulk Transfers).

طريقة عمل البروتوكول USB:

يتبادل منفذ USB البيانات من خلال إطارات (Frame)، لكل إطار فترة زمنية ثابتة تتغير بحسب نوع المنفذ (USB1.1, USB2.0) والطرفية المستخدمة، فمثلاً: يكون طول الإطار (1ms) في حالة كانت السرعة المستخدمة هي سرعة منخفضة أو كاملة، في حين يكون عند السرعة العالية (125µs).

يحوي كل إطار عدد من عمليات تداول البيانات (Transaction) وكل عملية تداول تتألف من مجموعة من رزم البيانات.

تقسم هذه الرزم إلى الأنواع التالية:

Token Packet: ترسل في بداية كل عملية تداول للبيانات وتحوي معلومات عن عملية التداول المراد إجرائها وتقسم إلى أربعة أنواع:

In Token: تقوم هذه الرزمة بإخبار الطرفية بأن المضيف يريد قراءة بيانات، وتتألف من خمسة حقول:

1. Synchronization: تتألف من ثمانية بتات وتوجد في بداية كل أنواع الرزم، وتستخدم لمواقتة المرسل مع المستقبل.
2. PID (Packet Identity): يتألف من ثمانية بتات تستخدم للدلالة على نوع الرزمة المرسلة، وتكون ثاني أربع خانات هي متممة لأول أربع خانات وذلك للتأكد من أن البيانات صحيحة.
3. Address: يحوي هذا الحقل عنوان الطرفية المطلوبة ويتألف من سبع خانات مما يسمح بعنونة 128جهاز.
4. Endpoint: يتألف من أربع خانات تدل على رقم النقطة النهائية المراد مخاطبتها.
5. CRC (Cyclic Redundancy Check): بطول 5 بت من أجل كشف الأخطاء.

Out Token: تقوم هذه الرزمة بإخبار الطرفية بأن المضيف يريد إرسال بيانات، وتتألف من نفس حقول الرزمة السابقة (In Token).

Setup Token: تستخدم هذه الرزمة للدلالة على بداية عملية تهيئة Control Transfer وهي تتألف من نفس حقول رزمة In Token.

Start of frame Token : يقوم المضيف بإرسال هذه الرزمة في بداية كل إطار أي كل 1ms±500ns ، وتحتوي هذه الرزمة على رقم الإطار ضمن حقل (Frame number) المؤلف من 11 خانة.

In	SYNC 0x01	PID 0x96	Address 7 bits	End point 4 bits	CRC 5 bits
Out	SYNC 0x01	PID 0x1E	Address 7 bits	End point 4 bits	CRC 5 bits
Setup	SYNC 0x01	PID 0xD2	Address 7 bits	End point 4 bits	CRC 5 bits
Start of frame	SYNC 0x01	PID 0x5A	Frame number 11 bits		CRC 5 bits

رزمة البيانات (Data Packet): وتقسم إلى نوعين:

1. رزمة بيانات من نوع Data0.
2. رزمة بيانات من نوع Data1.

لكلا نوعي رزم البيانات حقل بيانات بطول 1024bit وحقل CRC بطول 16bit وذلك بسبب كبر هذه الرزمة.

Data 0	SYNC 0x01	PID 0x3C	Data 0-1023 bits	CRC 16 bits
Data 1	SYNC 0x01	PID 0xB4	Data 0-1023 bits	CRC 16 bits

رزمة المصافحة (Handshake Packet): ولها ثلاثة أنواع:

Acknowledge: وتدل على أن عملية التداول قد تمت بنجاح وأنه تم استقبال البيانات بشكل صحيح.

Not acknowledge: ترسل هذه الرزمة للدلالة على أن الجهاز مشغول لا يمكنه بشكل مؤقت إرسال أو استقبال البيانات، وترسل أيضا للمضيف خلال عمليات التداول الخاصة بالمقاطعة لإخباره بعدم وجود بيانات.

Stall: تدل هذه الرزمة على أن النقطة النهائية في حالة توقف بسبب مشكلة ما وتتطلب تدخل المضيف أو أن الأمر المرسل غير مدعوم.

حلول التطوير باستخدام منفذ الاتصالات التسلسلي USB:

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

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

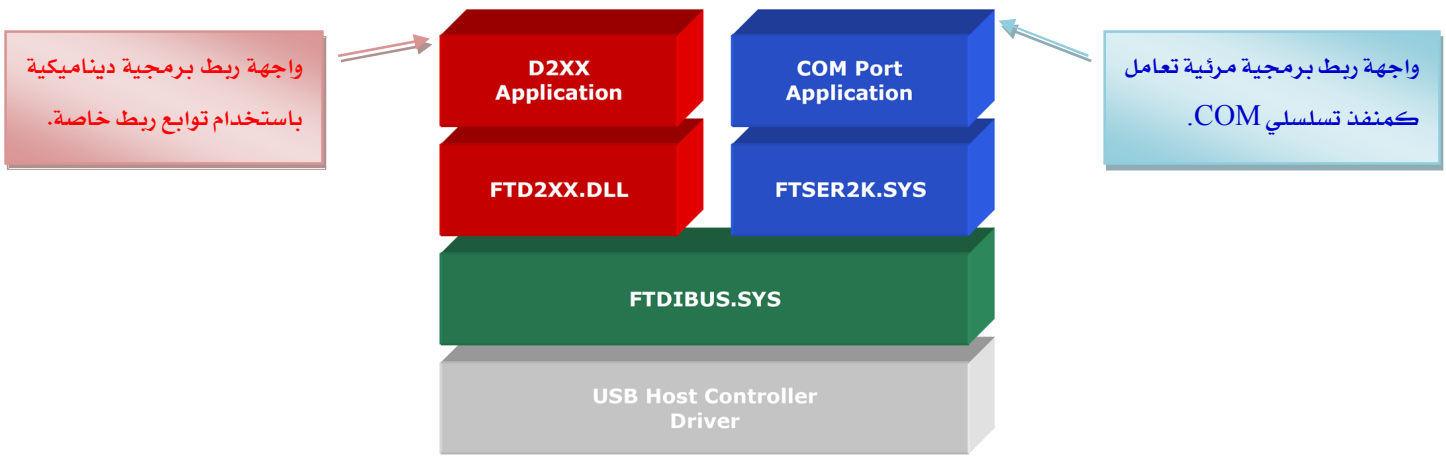
لذلك وبسبب الطلب المتزايد على هذه التقنية واقتحامها للسوق العالمية فإن هنالك الكثير من الشركات التي وفرت على المصممين عناء تصميم العتاد الالكتروني لينصّب اهتمامهم على كتابة برامج القيادة، لذلك كل ما يتوجب على المصمم هو الإطلاع على معايير USB بغرض فهم كيفية التعامل مع هذا العتاد الالكتروني.

تقدم بعض الشركات حلولاً للتعامل مع المنفذ USB باستخدام شرائح متكاملة تقوم على تحويل البروتوكول USB إلى نافذة تسلسلية UART تمكن المستخدم من توصيل المتحكم المصغر بشكل مباشرة مع هذه النافذة، بالإضافة إلى ذلك توفر هذه الشرائح حلولاً برمجية من خلال مكتبات ربط ديناميكية من أجل ربط نظام مع الحاسب عن طريق البروتوكول USB ومعالجة بارامترات النظام أو إرسال أوامر التحكم إلى النظام.

من أشهر وأكثر الشرائح انتشاراً واستخداماً هي الدارة المتكاملة FT232 التي هي عبارة عن دارة تحويل USB<>UART التي تنتجها شركة **FTDI**.

إن عملية تحويل البروتوكول USB تم بنائها في داخل هذه الشريحة ككيان صلب (Hardware) دون الحاجة إلى برمجة الشريحة، حيث تؤمن هذه الشريحة واجهتي ربط ديناميكي للتعامل برمجياً مع المنفذ باستخدام توابع خاصة وجاهزة موجودة في مكتبات الربط الديناميكي للشريحة دون الحاجة إلى بناء البروتوكول USB بشكل برمجي من البداية أو حتى فهم مبدأ عمله.

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



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

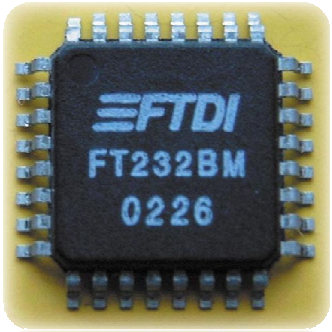
	VCP Driver	D2XX.DLL Driver
بساطة البرنامج	برنامج بسيط	برنامج معقد
السرعة	سرعة ثابتة لا يمكن تغيرها 300 KB/s	سرعة قابلة للتغيير تصل إلى 3MB
التحكم بالشريحة	لا يمكن التحكم بالشريحة	تحكم كامل ومباشر بالشريحة

VCP (Virtual Com Port): يعرف منفذ USB كمنفذ Com تسلسلي إضافي، مما يسمح لنا بالتخاطب مع منفذ USB كمنفذ Com معياري.

D2XX.DLL: يسمح هذا التعريف بالوصول المباشر إلى كامل مميزات هذه الشريحة عن طريق أوامر موجودة ضمن مكتبة ربط ديناميكيةDLL.

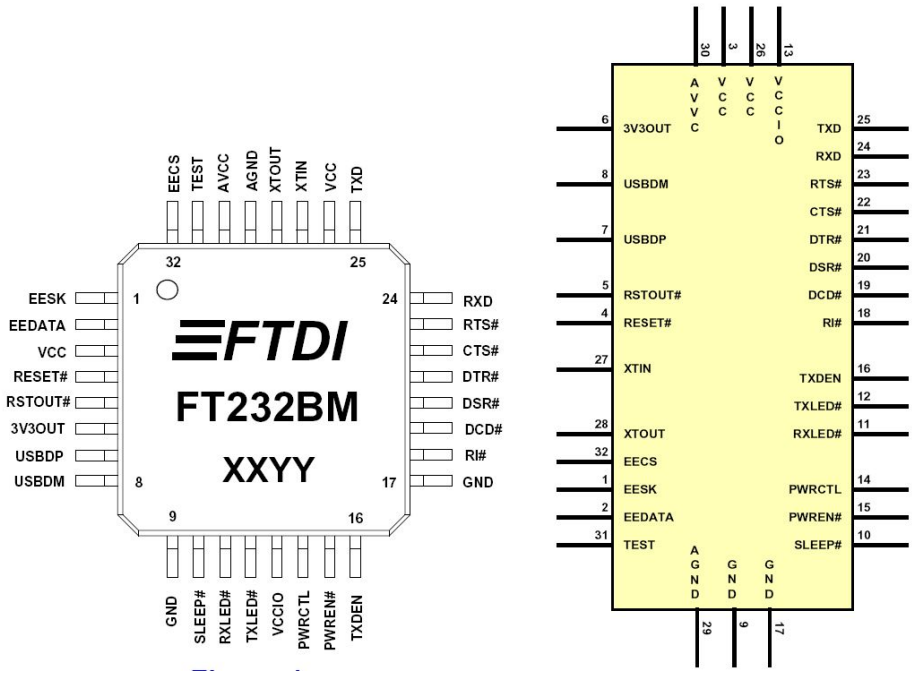
الشريحة FT232BM:

- توفر الشركة الصانعة برنامج القيادة لهذه الشريحة بشكل مجاني متوافق مع معظم أنظمة التشغيل.
- تقدم شركة FTDI برنامجي قيادة لشرائحها (VCP & D2XX.DLL).
- متوافقة مع المعيارين USB1.1, USB2.0.
- تدعم هذه الشريحة ملائمة كاملة لنظم الاتصالات التسلسلية.
- سرعة اتصال 300kb~2Mb بحسب نوع برنامج القيادة.
- ذاكرة استقبال وسيطية من نوع FIFO بطول 384 بايت.
- ذاكرة إرسال وسيطية من نوع FIFO بطول 128 بايت.
- رقمي VID, PID ورقم تسلسلي للمنتج ووصف لهذا الجهاز.
- توفر العديد من المقالات التقنية من الشركة المصنعة تقدم معلومات مفصلة عن طرق استخدام هذه الشريحة.

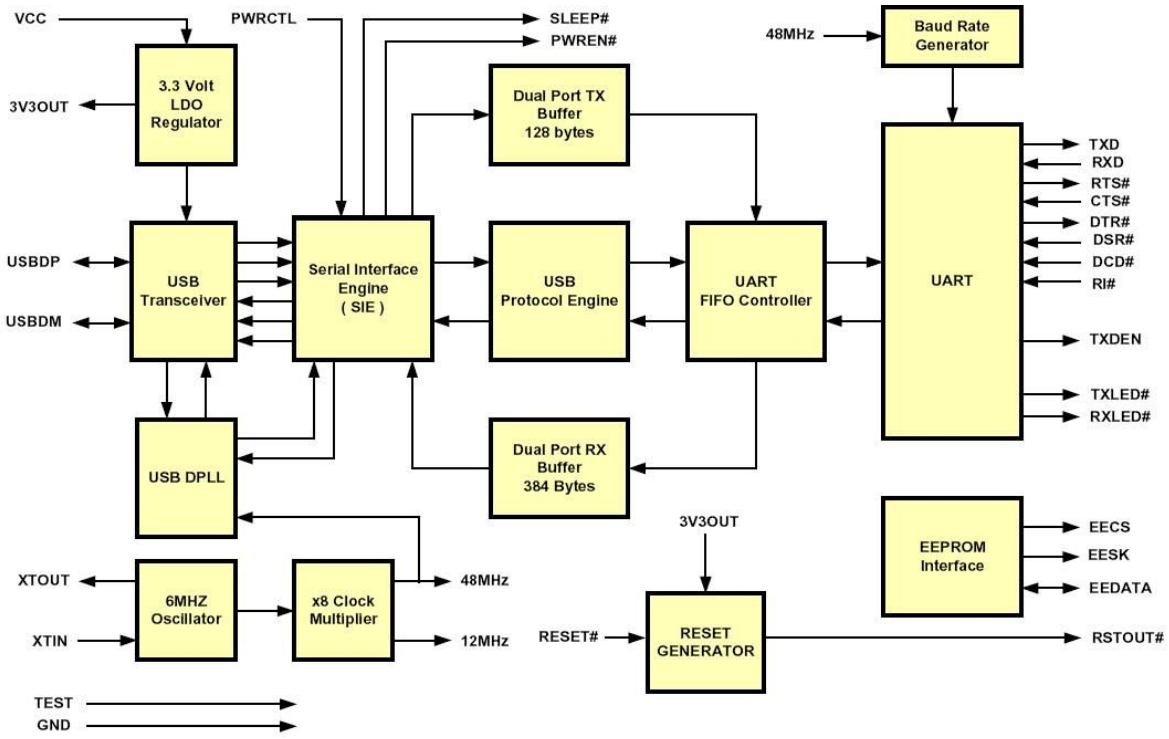


تلعب هذه الشريحة دور الملائم بين منفذ USB وبين النظام حيث تقوم باستقبال بيانات منفذ USB وتستخلص منها البيانات المطلوبة، كما تقوم بإرسال البيانات من المتحكم بشكلها التسلسلي إلى منفذ USB بعد إضافة الحقول اللازمة لتحقيق بروتوكول USB.

توزع أقطاب الشريحة:



المخطط العام لبنية هذه الشريحة مع العناصر الأساسية :



تمتلك هذه الشريحة 11 قطب للوصل مع النظام بحيث تؤمن تبادل البيانات مع الحاسب باستخدام بروتوكول المصافحة.

عند وصل الشريحة مع الحاسب وبعد أن يقوم نظام التشغيل بتحميل برنامج القيادة لها، تقوم هذه الشريحة بإعطاء صفر منطقي على القطب #PWREN وبالتالي يمكن استخدام هذا القطب لتشغيل الدارة الخارجية عند وصل النظام بالحاسب.

تدخل الشريحة في نمط الطاقة المنخفضة (Sleep mode) إذا لم تكن هناك عملية تبادل بيانات لمدة 3ms (أي بطول ثلاث إطارات)، في هذه الحالة تصبح حالة القطب "Sleep="1 ، وبالتالي فإن ربط هذا القطب مع المتحكم بطريقة مناسبة يمكن أن يدخله في نمط الطاقة التحتية أيضاً.

تمتلك هذه الشريحة قطب Wake up يسمح للنظام بإخراج الحاسب من الوضع الاحتياطي عند ورود جبهة صاعدة وذلك في حال كان النظام في حالة وضع احتياطي، أما إذا لم يكن كذلك فإن هذه الجبهة الصاعدة تؤدي إلى إرسال البيانات الموجودة في ذاكرة الاستقبال إلى الحاسب.

يتم تحقيق المتطلبات الخاصة لبرتوكول التخاطب مع منفذ USB في الشريحة من قبل وحدة الملائمة التسلسلية SIE (Serial Interface Engine) التي تقوم بالمهام التالية:

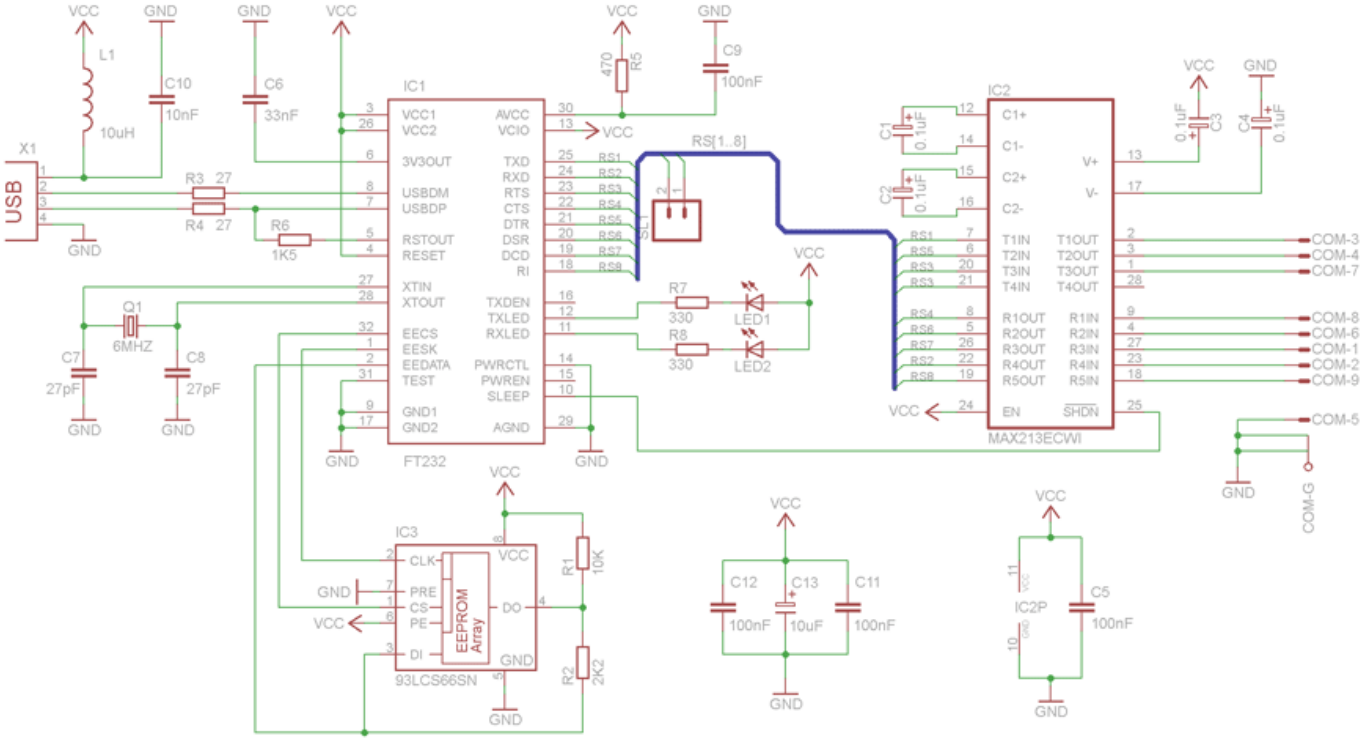
- كشف الرزم المستقبلة، وإرسال الرزم من وحدة بروتوكول USB إلى مرسل USB.
- فحص و توليد قيم CRC.
- كشف و توليد إشارات Start Of Packet, End Of Packet, Resume, Reset.
- تشفير وفك تشفير البيانات على الممر بحسب تقنية NRZI.
- فك تشفير وتوليد معرفات الرزم (PID).

تقوم وحدة (USB Transceiver) بملائمة الشريحة مع الممر وفقاً للشروط الكهربائية المحققة للمعايير USB1.1 ,USB2.0، بما في ذلك تعريف سرعة الشريحة على الممر باستخدام مقاومة شد على القطب D+ لأن سرعة الشريحة تصنف كسرعة كاملة.

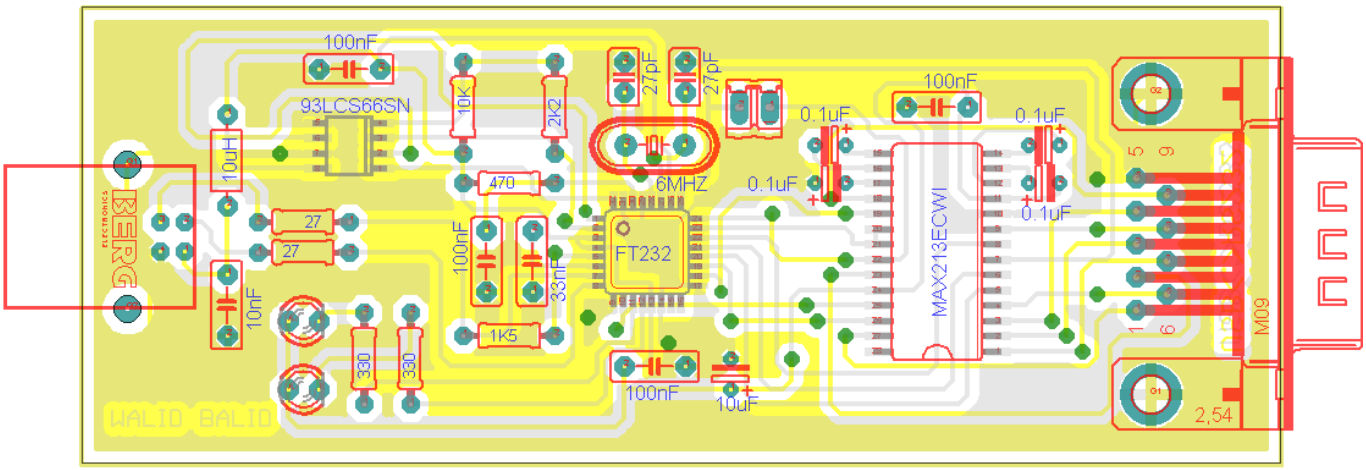
دارة ملائمة الشريحة FT232BM:

تحتاج هذه الشريحة تردد 48MHz تحصل عليه بواسطة ضارب داخلي لتردد 6MHz هزاز كريستالي خارجي.
يُخزَّن كلا رقمي PID, VID والرقم التسلسلي في ذاكرة نوع EEPROM خارجية، كما يخزن أيضاً معلومات أخرى عن صنف الجهاز والتيار الأعظمي الذي يحتاجه ومعلومات أخرى.

الشكل التالي المخطط التصميمي لدارة التحويل RS232<>UART<>USB.



الشكل التالي مخطط الدارة المطبوعة وتوزع العناصر لدارة التحويل RS232<>UART<>USB.



برمجة المتحكمات المصغرة

التجارب العملية

الجلسة الحادية عشرة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

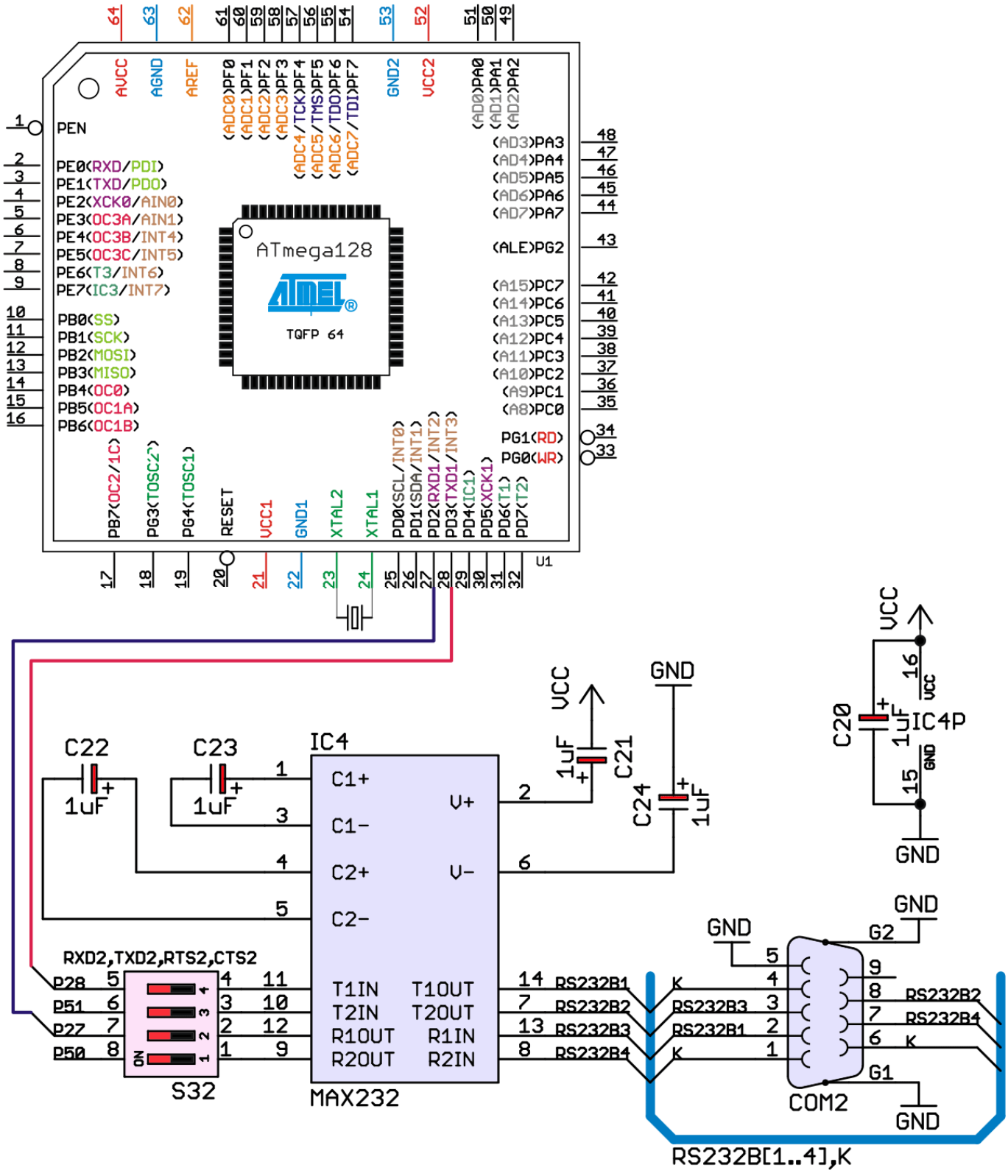
WALID BALID, Tuesday, December 15, 2009

Exp.30: Programming Internal EEPROM التجربة الثلاثون: برمجة الذاكرة الداخلية EEPROM

الغاية من التجربة:

استثمار وبرمجة ذاكرة المعطيات الداخلية (EEPROM) للمعالج.

مخطط الدارة:



شرح عمل الدارة:

تحوي الدارة أعلاه على دارة ملائمة بين النافذة التسلسلية UART1 للمتحكم المصغر ومنفذ الاتصال التسلسلي RS232 للحاسب. سوف نقوم بكتابة برنامج للقراءة والكتابة من وإلى ذاكرة المعطيات الداخلية للمعالج (EEPROM) باستخدام النافذة التسلسلية.

التعليمات الجديدة:

التعليمة	شرح التعليمة
Dim Var **As** [**xram**|Sram|**Eram**] Type [**at** Location][**overlay**] *Examples:* Dim Sram_var As Sram Byte At &H10 Dim Eprm_var As Eram Byte At &H80	تعريف متحول في ذاكرة (معطيات خارجية|معطيات داخلية|ذاكرة دائمة) مع خيار تحديد موقع المتحول في الذاكرة. **overlay**: خيار يجعل المتحول المعرف متحول خيالي في الذاكرة أي أنه لن يتم حجز مساحة في الذاكرة ولكن تكمن الخطورة في كتابة متحول على العنوان.
Writeeeprom Var , Address	كتابة قيمة (Var) إلى الذاكرة EEPROM عند العنوان Address.
Writeeeprom Var , Label	كتابة قيمة (Var) إلى الذاكرة EEPROM لتتوضع عند اللافتة Label.
Readeeprom Var , Address	قراءة قيمة (Var) من الذاكرة EEPROM عند العنوان Address.
Readeeprom Var , Label	قراءة قيمة (Var) من الذاكرة EEPROM المتوضعة عند اللافتة Label.
$eeprom	توجيه المترجم إلى تخزين البيانات الموجودة في التعليمة DATA والتي هي مباشرة بعد التوجيه في الذاكرة EEPROM، وسيتم توليد ملف ثنائي EPP.
$data	توجيه المترجم إلى أن البيانات التالية سيتم تخزينها في ذاكرة البرنامج.
$eepromhex	توجيه المترجم إلى تخزين البيانات في الملف EPP بصيغة Intel HEX. هذا التوجيه يجب أن يستخدم مع التعليمة $eeprom.
$eepleave	توجيه المترجم إلى عدم توليد ملف EPP وعدم محي الذاكرة EEPROM.
$default Sram | Xram | Eram *Examples:* $default Sram Dim A As Byte , B As Byte $default Eram Dim C As Byte , D As Byte	تعيين موقع حجز جميع المتحولات التالية للتوجيه $default.
$end $default	استعادة الإعدادات الافتراضية التي تم تغييرها باستخدام التوجيه السابق.
$noramclear	توجيه المترجم إلى عدم تصفير محتوى الذاكرة SRAM عند التهيئة.
$romstart = address	توجيه المترجم إلى تخزين البرنامج ابتداءً من العنوان المحدد.

ملاحظة: يوصى في الوثيقة الفنية لعائلة AVR بعد استخدام البايت الأول من الذاكرة EEPROM المتوضع عند العنوان صفر لأنه يمكن أن تتغير قيمة هذا البايت أثناء تصفير المعالج أو أي حالة عابرة.

برنامج تشغيل الدارة (1):

```
$regfile = "m128def.dat"
$crystal = 4000000
$baud = 9600
'------------------
Dim B  As Byte , I As Byte
Dim W  As Word , S As String * 5

Dim Eb As Eram Byte At 13
Dim Ei As Eram Integer At 14
Dim El As Eram Long At 16
Dim Es As Eram String * 5 At 20
'------------------
Do
   S = "ABCDE" : Es = S
   S = ""
   S = Es : Print S

   B = 10 : Eb = B
   B = 0
   B = Eb : Print B
'/------

   For I = 0 To 4
      Readeeprom B , I
      Print B
   Next I
'/------

   S = "abcde" : W = 10000
   Writeeeprom S , 5
   Writeeeprom W , 11

   S = "" : W = 0
   Readeeprom S , 5 : Print S
   Readeeprom W , 11 : Print W
'/------

   Restore Lbl
   Read B : Print B
   Read B : Print B
Loop
End
'------------------

Lbl:
Data 10 , 12
'------------------

$eeprom
   Data 1 , 2 , 3 , 4 , 5
$data
```

التوجيهات.

تعريف متحولات في الذاكرة SRAM

تعريف متحولات في الذاكرة EEPROM

إسناد قيمة متحول في الذاكرة SRAM إلى متحول في الذاكرة EEPROM.

قراءة قيمة عند عنوان محدد في الذاكرة EPROM إلى متحول في الذاكرة SRAM.

إسناد قيمة متحول في الذاكرة SRAM إلى عنوان محدد في الذاكرة EEPROM.

قراءة قيمة عند عنوان محدد في الذاكرة EPROM إلى متحول في الذاكرة SRAM.

تحميل قيم إلى الذاكرة SRAM عند لافتة محددة ومخزنة في ذاكرة البرنامج ROM.

تخزين قيم في الذاكرة ROM.

تخزين قيم في الذاكرة EEPROM.

برنامج تشغيل الدارة (2):

التوجيهات.

```
$regfile = "m128def.dat"
$crystal = 4000000
$baud = 9600
$eepromhex
$eepleave
'------------------
Dim Var As Sram Byte At &H200
'------------------
$eeprom
   Label1:
   Data 1 , 2 , 3 , 4 , 5

   Label2:
   Data 10 , 20 , 30 , 40 , 50
$data
'------------------
   Readeeprom Var , Label1
      Print Var
         Readeeprom Var
            Print Var
'/------


   Readeeprom Var , Label2
      Print Var
         Readeeprom Var
            Print Var
'/------


   Var = 100
   Writeeeprom Var , Label1
   Var = 101
   Writeeeprom Var

   Readeeprom Var , Label1
      Print Var
   Readeeprom Var
      Print Var
'/------


   Var = 0
   Writeeeprom Var , 3
      Readeeprom Var , 3
         Print Var
End
```

تخزين قيم في الذاكرة EEPROM عند لافتات محددة.

قراءة قيمة عند لافتة محددة في الذاكرة EPROM إلى متحول في الذاكرة SRAM.

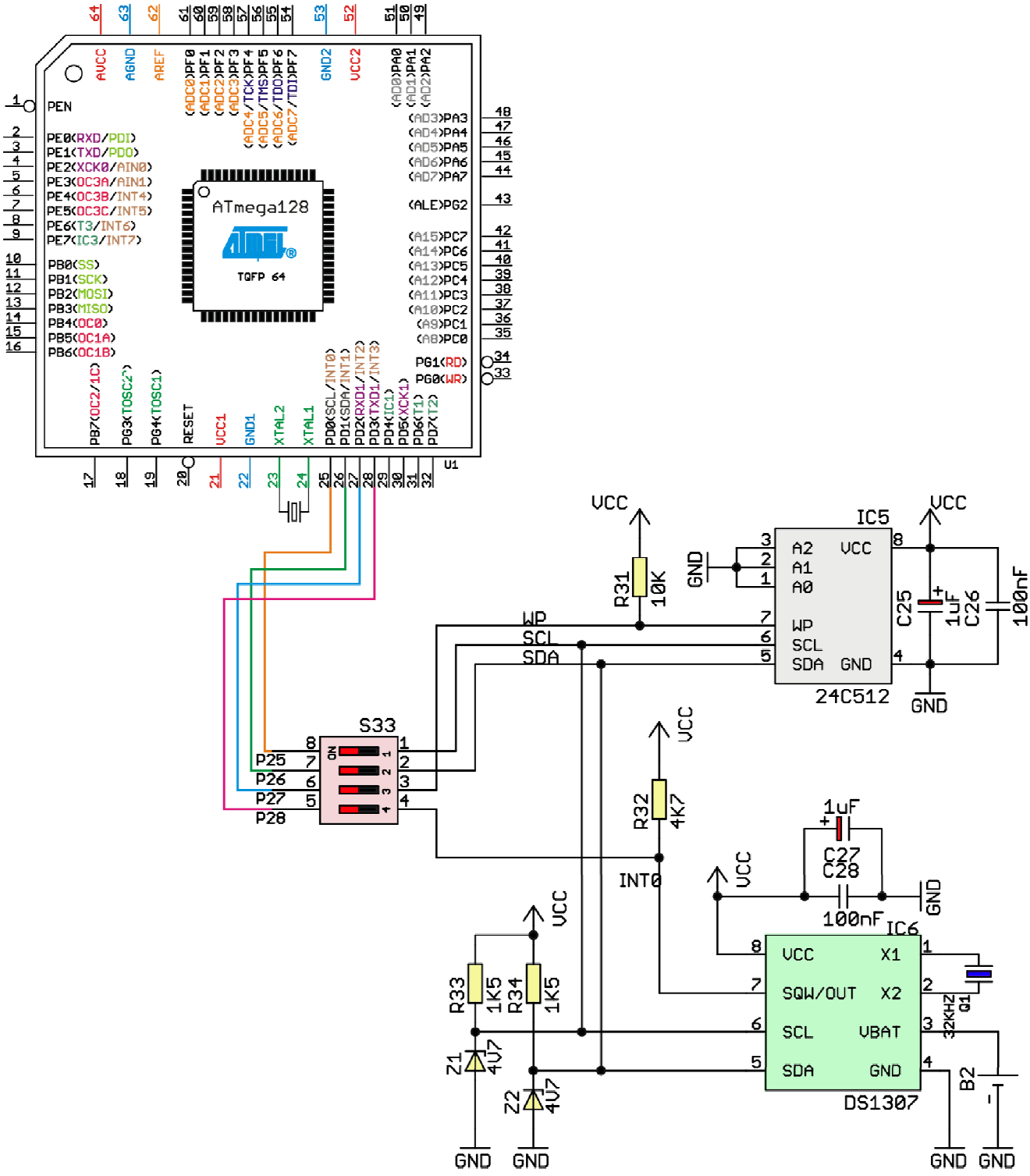
قراءة قيمة عند لافتة محددة في الذاكرة EPROM إلى متحول في الذاكرة SRAM.

إسناد قيمة متحول في الذاكرة SRAM إلى الذاكرة EEPROM.عند لافتة محددة.

الغاية من التجربة:

دراسة مبدأ عمل البروتوكول I²C من خلال برمجة ذاكرة معطيات تسلسلية EEPROM ودارة توليد الزمن الحقيقي RTC.

مخطط الدارة:



مبدأ عمل البروتوكول I^2C:

تم تطوير ويعتبر البروتوكول I^2C (Inter-Integrated Circuit) في أوائل عام 1980 من قبل شركة Philips بهدف تخفيض كلفة تصنيع المنتجات الإلكترونية (TV) ومن أجل ربط متحكم مع بعض المحيطيات في أجهزة التلفاز ويعتبر الأكثر استخداماً في الأجهزة الإلكترونية ويسمى أيضاً TWI (Two Wire Interface).

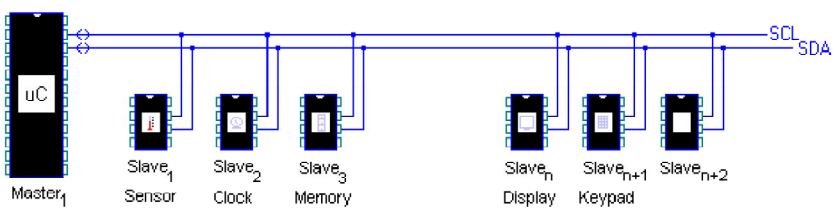
يصنف البروتوكول I^2C من بروتوكولات الاتصال التسلسلي المتزامنة التي تعتمد على خطين (SDA, SCK)، الخط SDA دائماً ثنائي الاتجاه، فإما أن يكون اتجاه البيانات من المرسل إلى المستقبل أو العكس، أما الخط SCK فهو أحادي الاتجاه في الأنظمة التي تعتمد مبدأ One-Master<>Multi-Slaves ويكون ثنائي الاتجاه في الأنظمة التي تعتمد على مبدأ Multi-Master<>Multi-Slaves.

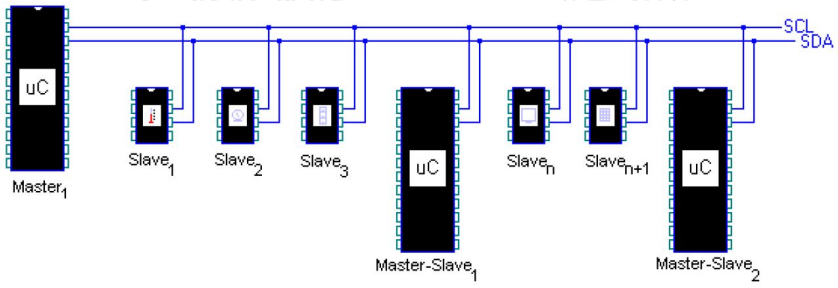
يستخدم هذا البروتوكول من أجل ربط Chip-to-Chip عند سرعات نقل منخفضة ومتوسطة، ويعمل من أجل Multi-Drop Bus حيث يمكن أن يتواجد على خط النقل قائد وحيد (Master) وأكثر من مقاد (Multi-Slave) ويتم التخاطب بين جهازين فقط معاً في نفس اللحظة (Master<>Slave)، كما يمكن إعداده من أكثر وجود أكثر من قائد (Multi-Master) وأكثر من مقاد (Multi-Slave).

خرج الأجهزة في البروتوكول I^2C هو من نوع Open collector، لذلك يتم ربط مقاومات رفع لكلا SDA, SCK، وفي حال البطالة تكون حالة كلا الخطين "1".

كل جهاز يعمل وفق البروتوكول I²C يملك عنوان فريد (Unique Address) مؤلف من 7-Bit (112 nodes)، أو يمكن أن يتوفر بعنوان 10-bit (1008 nodes) يستخدم هذا العنوان لتحديد الجهاز المقاد المراد التخاطب معه من قبل القائد.

إن عدد الأجهزة على خط النقل يعتمد مباشرة على سعة الخط حيث أن القيمة الأعظمية للسعة يجب أن لا تتجاوز 400pF، وغالباً تكون سعة كل جهاز بحدود 10pF.





المميزات والمساوئ لبرتوكول الاتصال I^2C:

المساوئ (Disadvantages)	المحاسن (Advantages)
× مسافة الاتصال قصيرة (3meter).	ü يمكن التخاطب مع أكثر من Slave باستخدام خطين.
× سرعات منخفضة لا تتجاوز 400Khz.	ü كلفة البناء منخفضة ، وسهل التطبيق.
× محدودية في عناوين الأجهزة.	ü شائع الاستخدام، ومتوفر ككيان صلب وبرمجياً.
	ü يمكن فصل ووصل أي جهاز من الناقل دون أي تأثير على النظام أو الحاجة لأي تغيير.

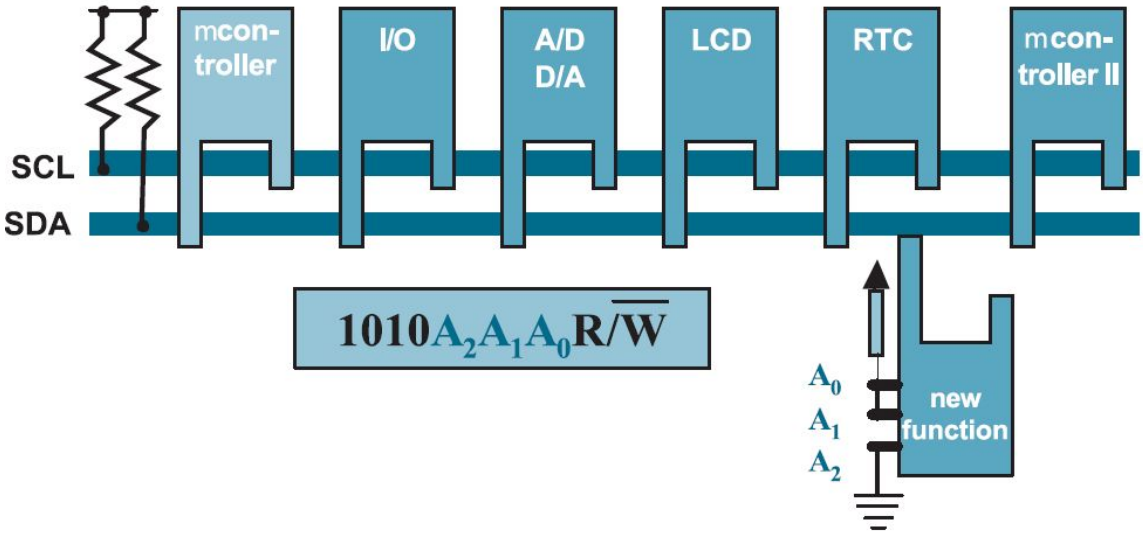
Interfacing I2C EEPROM

Description:

I^2C is an abbreviation of Inter Integrated Circuit and is a protocol for serial communication between Integrated Circuits, it is also called Two Wire Interface (TWI). The bus is used for communication between microcontrollers and peripheral devices like memories, temperature sensors and I/O expanders. An EEPROM is a Electrically Erasable and Programmable Read Only Memory.

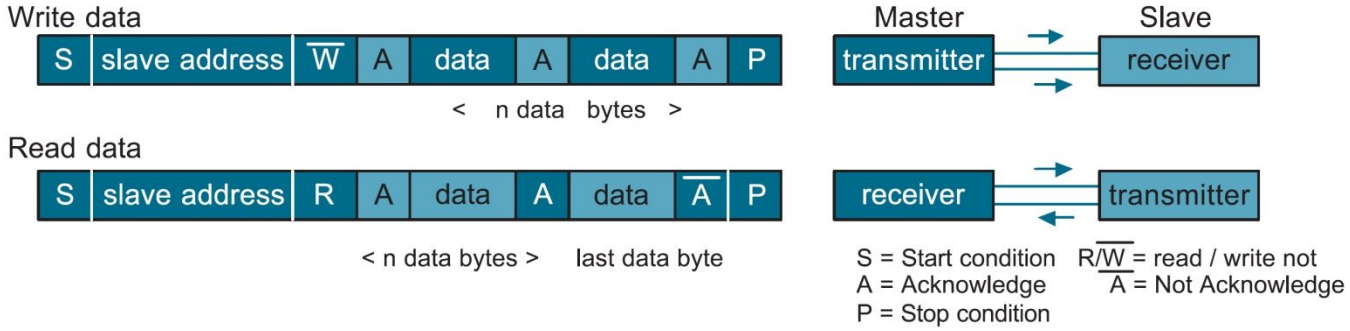
EEPROM Model	Size	Internally Organized	Address (hex)
AT24C01	128 Bytes	128 x 8 = 1024 bits	00000 >> 0007F
AT24C02	256 Bytes	256 x 8 = 2048 bits	00000 >> 000FF
AT24C04	512 Bytes	512 x 8 = 4096 bits	00000 >> 001FF
AT24C08	1 Kbyte	1024 x 8 = 8192 bits	00000 >> 003FF
AT24C16	2 Kbyte	2048 x 8 = 16384 bits	00000 >> 007FF
AT24C32	4 Kbyte	4096 x 8 = 32768 bits	00000 >> 00FFF
AT24C64	8 Kbyte	8192 x 8 = 65536 bits	00000 >> 01FFF
AT24C128	16 Kbyte	16384 x 8 = 131072 bits	00000 >> 03FFF
AT24C256	32 Kbyte	32768 x 8 = 262144 bits	00000 >> 07FFF
AT24C512	64 Kbyte	65536 x 8 = 524288 bits	00000 >> 0FFFF
AT24C1024	128 Kbyte	131072 x 8 = 1048576 bits	00000 >> 1FFFF

*The communication of the bus goes along two lines: **SDA** (Serial Data) and **SCL** (Serial Clock). Each I^2C device has a unique **7-bit address** (Device Select Code). The most significant bits are fixed and assigned to a specific device category (e.g. b1010 is assigned to serial EEPROMS). The three less significant bits (**A2,A1 and A0**) are programmable and used to address the device. The three bits allows eight different I2C address combinations and therefore allowing up to eight different devices of that type to operate on the same I2C-bus. The I2C address is send in the 1st byte, the lest significant bit of the first byte is used to indicate if the master is going to **write(0) or read(1)** from the slave.*



*The device that sends data along the bus is called **master**, a device that receives the data is called **slave**. The master starts the transmission with a start signal and stops the transmission with a stop signal on the SDA line. During the start and stop signals the SCL line has to be high. After the master has started the data-transmission with a start signal, the master writes a device address byte to the*

slave. Each data byte has to have a length of 8 bits. The slave has to acknowledge the reception of the data byte with a acknowledge-bit (ACK).

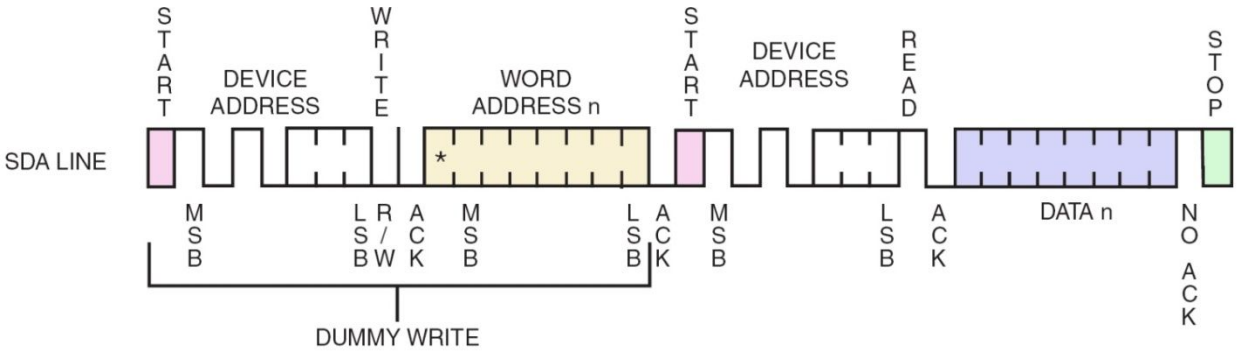


A write operation requires a device address bytes, two address bytes and the data-byte. Upon receive of the address the EEPROM sends an ACK and then clocks in the data-byte. The EEPROM sends again an ACK and the microcontrollers sends a stop-signal to terminate the write sequence.

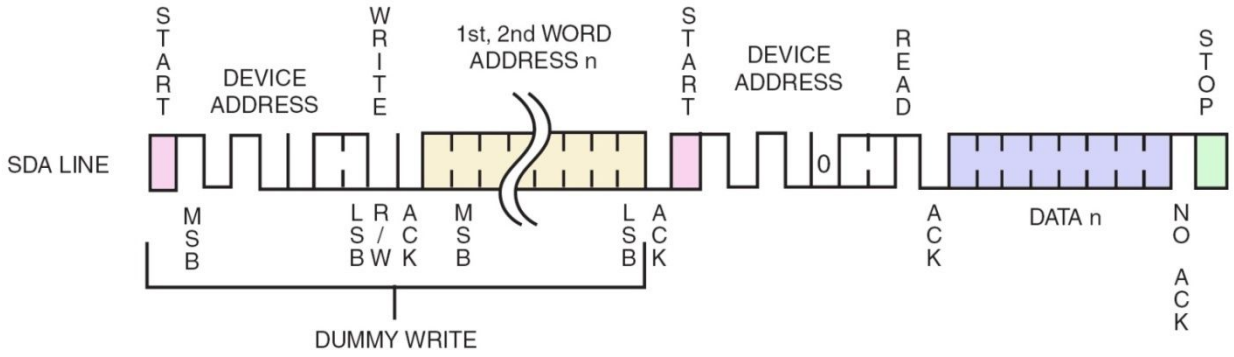
	Device Address / Page Byte	Word Address Bytes	Maximum Devices on Bus
1K/2K	1 0 1 0 A_2 A_1 A_0 R/W	1	8
4K	1 0 1 0 A_2 A_1 P_0 R/W	1	4
8K	1 0 1 0 A_2 P_1 P_0 R/W	1	2
16K	1 0 1 0 P_2 P_1 P_0 R/W	1	1
32K-512K	1 0 1 0 A_2 A_1 A_0 R/W	2	8
1M	1 0 1 0 A_2 A_1 P_0 R/W	2	4
	MSB LSB		

All devices from 32K – 512K will require no system changes and can be interchanged with only the page size differences to consider.

Low Density Random Read:



Medium and High Density Random Read:



AT24C32 (4 Kbyte)

*4096 * 8 = 32768 bits* ***0000 >> 0FFF*** *32 byte page*

&H0000	Saturday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H01FF
&H0200	Sunday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H03FF
&H0400	Monday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H05FF
&H0600	Tuesday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H07FF
&H0800	Wednesday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H09FF
&H0A00	Thursday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H0BFF
&H0C00	Friday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H0DFF
&H0E00	NON USED AREA *512Bytes*	&H0FFF

Software:

The BASCOM-AVR compiler is used to make a program that writes and reads one byte from the EEPROM. BASCOM has several embedded commands to control the I2C bus.

In BASCOM-AVR you first have to configure the ports you use for the SDA and SCL lines of the I2C bus. Then you send the device address to select the EEPROM that is connected to the I2C bus. After that you send two bytes to the EEPROM to select the address in the EEPROM to which you want to write the data. The last byte to send in a write sequence is the data byte.

```
$regfile = "m16def.dat"
$crystal = 2000000
$lib "I2C_TWI.LBX"
$baud = 9600
'----------------------------
Config Scl = Portc.0
Config Sda = Portc.1
Config Twi = 100000                                    '100KHZ
'----------------------------
Const Addressw = 160                                  '&B10100000 slave write address
Const Addressr = 161                                  '&B10100001 slave read address
'----------------------------
Dim Adres_h As Byte , Adres_l As Byte
Dim Rd_value As Byte , Wr_value As Byte
'----------------------------
Do
   Input "Wr_value:" , Wr_value
   Input "Adres_l:" , Adres_l
   Input "Adres_h:" , Adres_h

   Gosub Write_eeprom
   Gosub Read_eeprom

   Print "Error W: " ; Err
   print "Wr_value: " ; Wr_value

   Print "Error R: " ; Err
   Print "Rd_value: " ; Rd_value
Loop
End
'----------------------------
Write_eeprom:
    I2cstart                                           'Start condition
    I2cwbyte Addressw                                  'Slave address
    I2cwbyte Adres_h                                   'H address of EEPROM
    I2cwbyte Adres_l                                   'L address of EEPROM
    I2cwbyte Wr_value                                  'Value to write
    I2cstop                                            'Stop condition
    Waitms 10                                          'Wait for 10 milliseconds
Return
'----------------------------
Read_eeprom:
   I2cstart                                            'Generate start
   I2cwbyte Addressw                                   'Slave adsress
   I2cwbyte Adres_h                                    'H address of EEPROM
   I2cwbyte Adres_l                                    'L address of EEPROM
   I2cstart                                            'Repeated start
   I2cwbyte Addressr                                   'Slave address (read)
   I2crbyte Rd_value , Nack                            'Read byte
   I2cstop                                             'Generate stop
Return
'----------------------------
```