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.

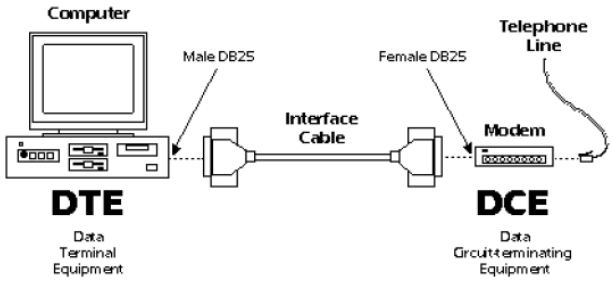
تستخدم الاتصالات التسلسلية على نطاق أوسع بكثير من الاتصالات التفرعية وتمتاز بمناعة عالية ضد الضجيج ونقل لمسافات بعيدة، كما أن حجم الناقل سيكون صغير وكلفته ضئيلة نسبياً مقارنة مع الناقل التفرعية.

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**

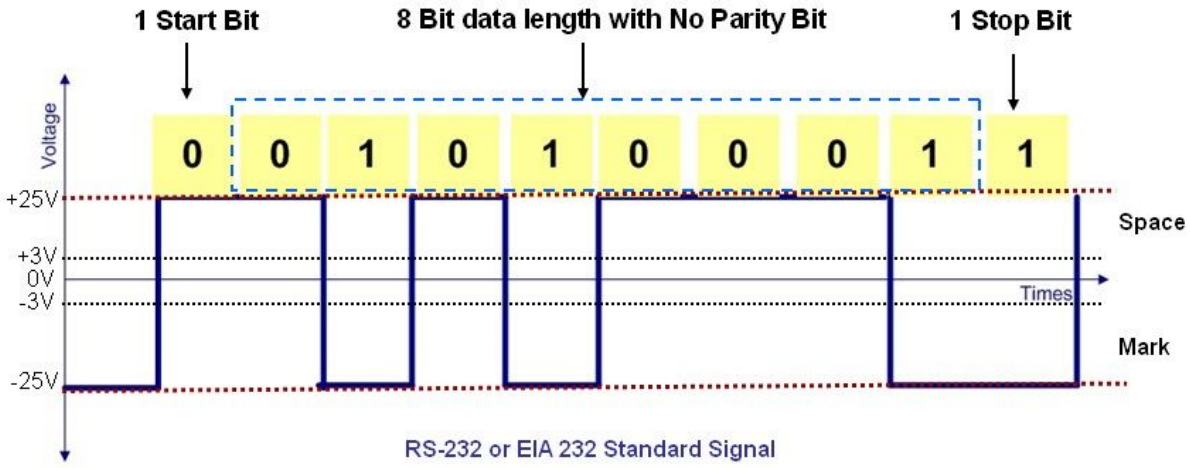
[1] High-speed, for connecting computers to mass storage devices
[2] Very high speed, broadly comparable in scope to PCI
[3] Control of electronic musical instruments
[4] Control of theatrical lighting
[5] New replacement for parallel IDE
[6] Spacecraft communication network
[7] High speed telecommunication 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].



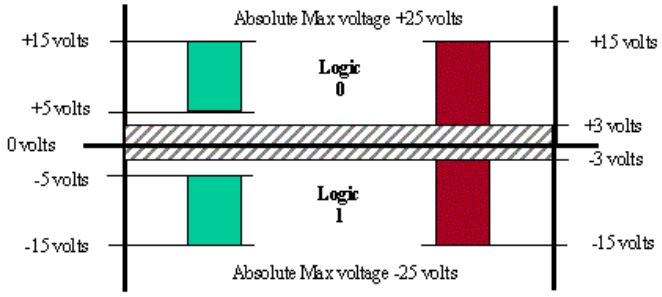
يتم إرسال كل بايت كحزمة مؤلفة من مجموعة بتات على الشكل التالي:



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:

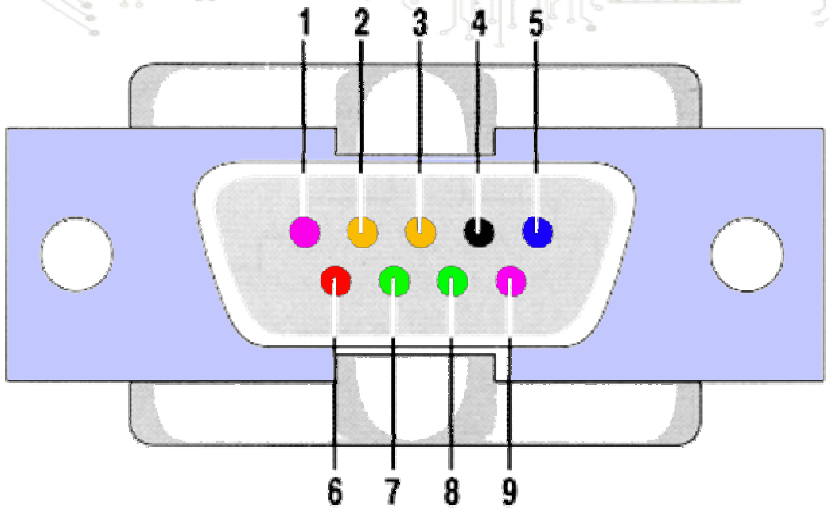
المساوئ (Disadvantages)	المحاسن (Advantages)
× مناسب فقط من أجل الربط بين System-to-System أكثر من كونه قابلاً للربط بين Chip-2-Chip أو من أجل تطبيقات Chip-2-Sensor.	ü بروتوكول اتصال شائع الاستخدام في كثير من التطبيقات ومعتمد من قبل العديد من الشركات.
× معدل نقل بيانات منخفض جداً من أجل مسافة اتصال كبيرة.	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):

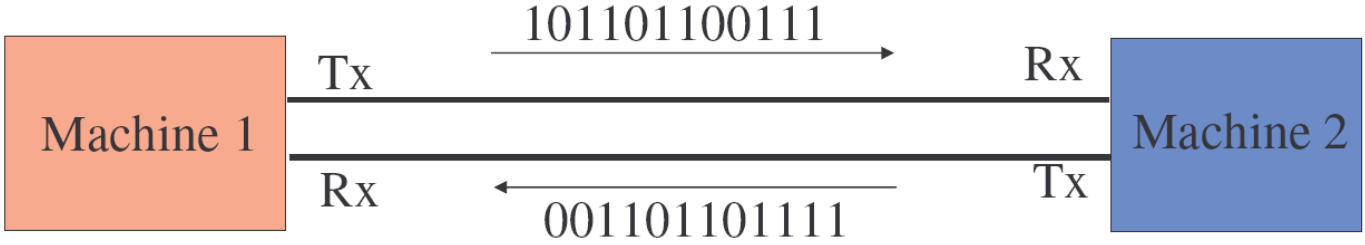
قطب تحكم يقوم بإعلام الطرفية الرئيسية (DTE) أنه استلم إشارة الإعلام بإرسال البيانات السابقة ويمكنها الآن أن تبدأ بإرسال البيانات إلى الطرفية الثانوية (DCE)، وبالتالي يمكن استخدام هذه الإشارة من أجل تفعيل دارة الاستقبال قبل إرسال أي إشارة.

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):

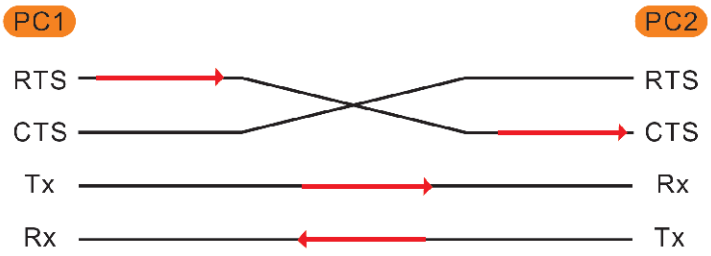
قطب تحكم يقوم بإعلام الطرفية الرئيسية (DTE) بوجود رنين من أجل فتح الخط، ويستخدم فقط في حال استخدام البوابة من أجل ربط بين حاسب وجهاز مودم.

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.



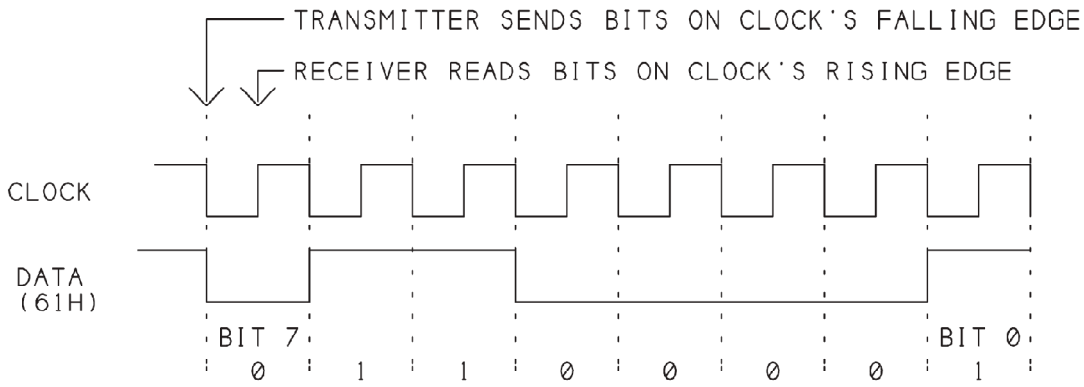
المواصفات الفنية للبروتوكول RS232:

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.

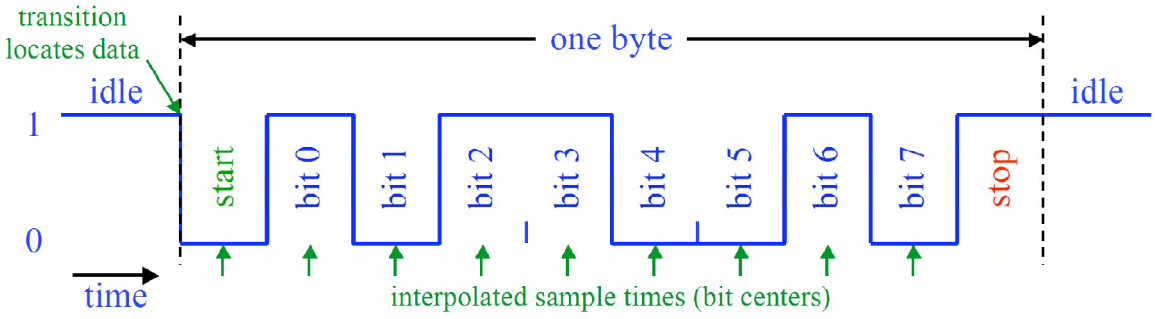
مفهوم الاتصالات التسلسلية المتزامنة (Synchronize) وغير المتزامنة (Asynchronize):

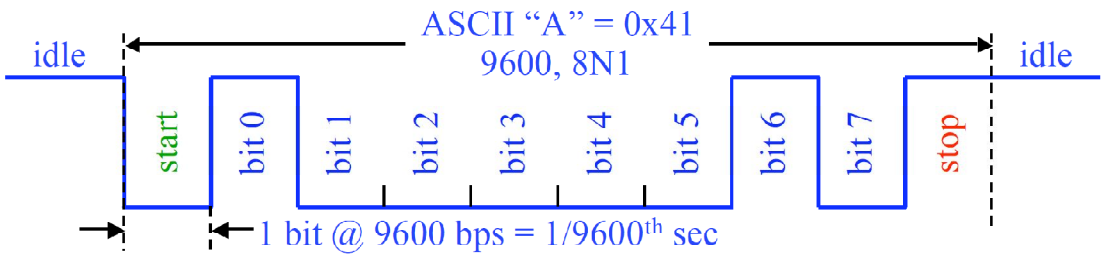
أولاً: الاتصالات المتواقتة (المتزامنة): يكون فيها بروتوكول الإرسال مؤلف من خطين على الأقل أحدهما خط التزامن (clock or strobe)، وبالتالي فإن سرعة إرسال البيانات تتحدد من خلال تردد إشارة التزامن بحث يتم إرسال كل بت من البتات تسلسلياً عند جبهة التزامن (صاعدة أو هابطة).



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.

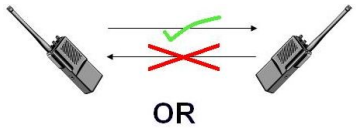




ملاحظة: بخلاف الاتصالات المتواقتة فإن ازدياد المسافة بين الطرفيتين لا يؤدي إلى فشل عملية النقل، كما أن هذه الطريقة أقل كلفة وأبسط بنية وأسهل برمجة.

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

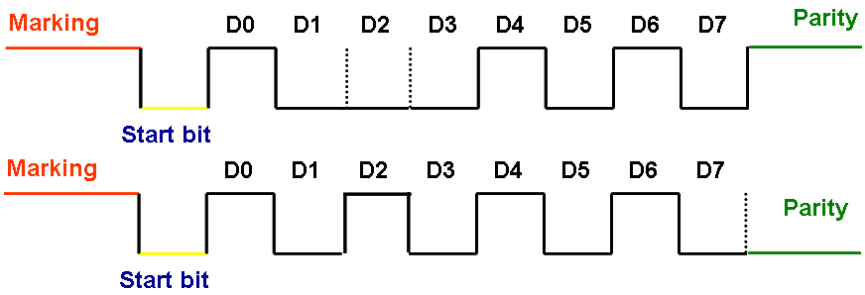
- تحديد نمط الإرسال: أحادي الاتجاه (Half-Duplex) أو ثنائي الاتجاه (Full-Duplex).
- تحديد عدد البتات لكل محرف: 6, 7 or 8 bit.
- تحديد معدل سرعة الإرسال (Baud Rate).
- تحديد استخدام أو عدم استخدام خانة فحص الإيجابية (Parity Bit)، وفي حال الاستخدام يجب تحديد نمط فحص خانة الإيجابية (Even or Odd).
- تحديد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



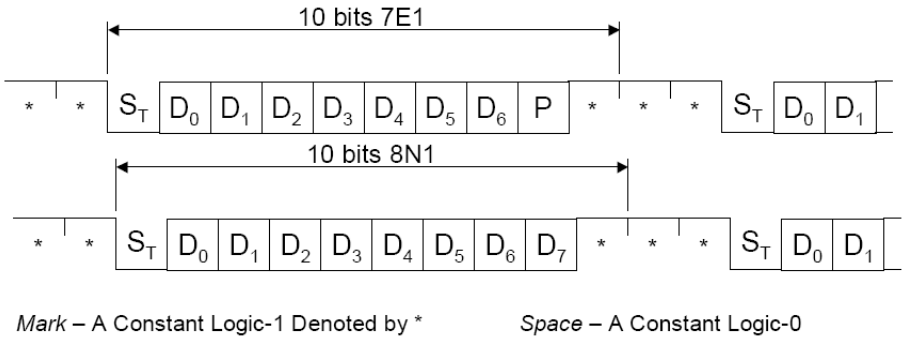
في حال كون خانة الإيجابية "Even" فإن هذه الخانة تملك القيمة "0" إذا كان عدد الواحدات في البايت المرسل زوجي وإلا فستصبح "1". الأمثلة التالية توضح ذلك.

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

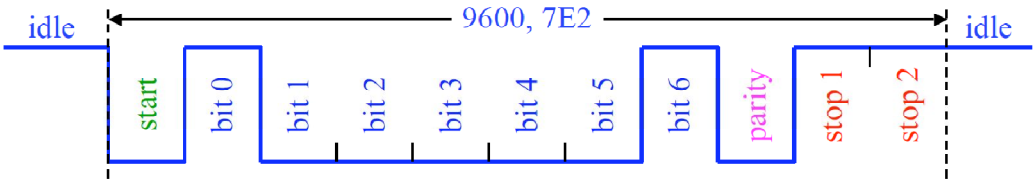
في حال كون خانة الإيجابية "Odd" فإن هذه الخانة تملك القيمة "0" إذا كان عدد الواحدات في البايت المرسل فردي وإلا فستصبح "1". الأمثلة التالية توضح ذلك.

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:

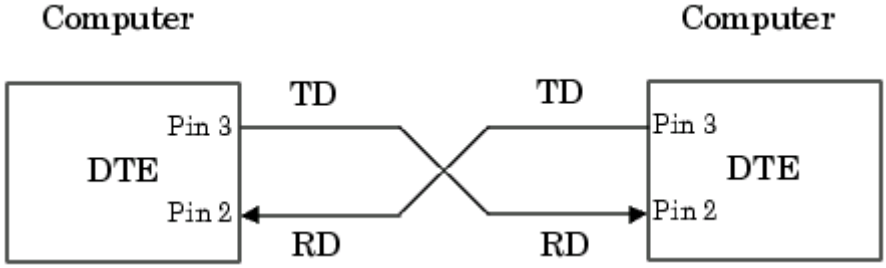
300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, etc...

إن الزمن اللازم لإرسال بت واحد يعطى بالعلاقة التالية:

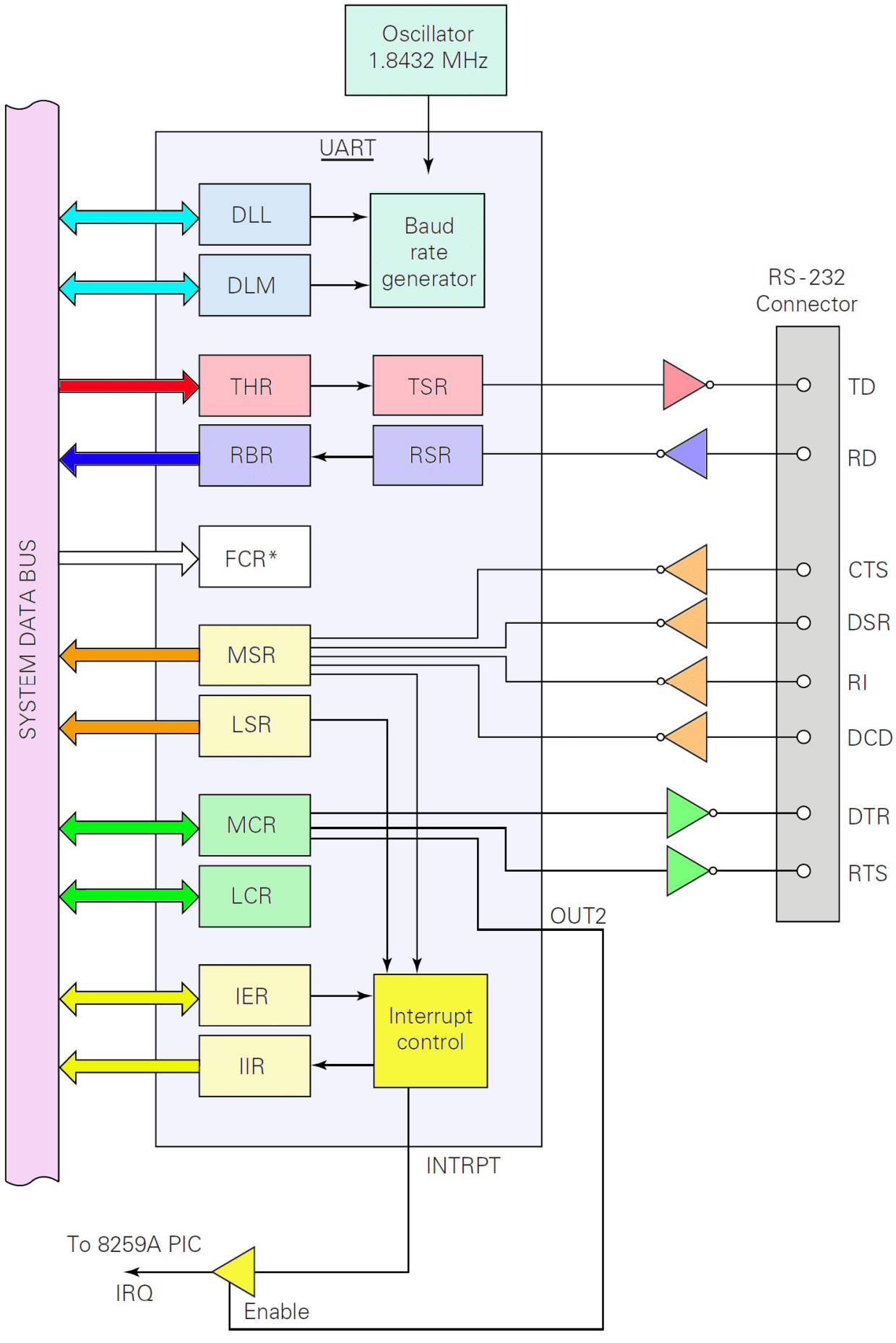
$$Bit_{Time} = \frac{1}{Baud\ Rate}$$

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:



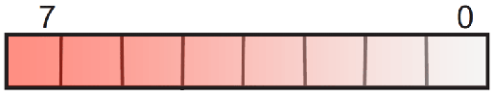
المسجلات الداخلية للنافذة التسلسلية RS232 في الحاسب:

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

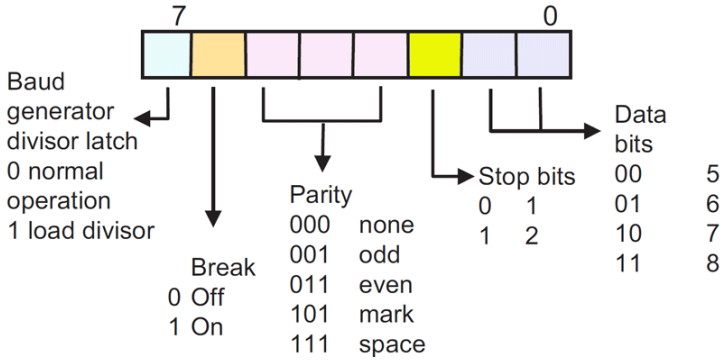
يتم منه قراءة البيانات الواردة عبر القطب RxD وإرسال البيانات الصادرة عبر القطب TxD.



مسجل التحكم بالخط (LCR, Line Control Register):

COM1: 0x3FB | COM2: 0x2FB

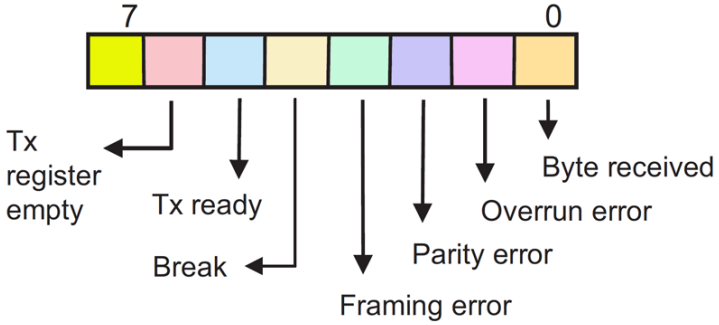
يتم فيه تعيين إعدادات (بارامترات) إطار البيانات.



مسجل حالة الخط (LSR, Line Status Register):

COM1: 0x3FD | COM2: 0x2FD

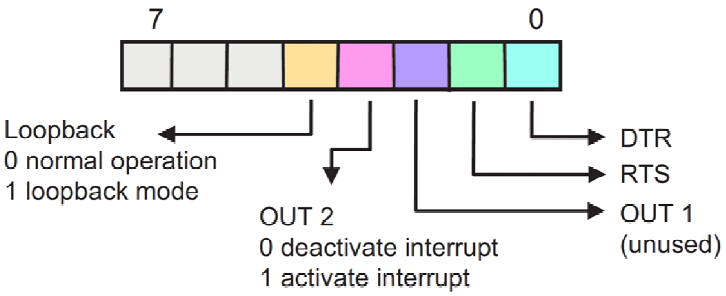
يتم منه قراءة حالة العمليات الجارية على الخط من أجل كشف الأخطاء والاستعلام عن حالة مسجل الإرسال.



مسجل التحكم بالمودم (MCR, Modem Control Register):

COM1: 0x3FC | COM2: 0x2FC

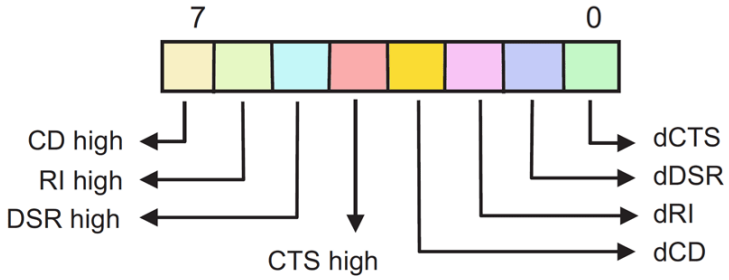
يتم فيه تعيين إعدادات (بارامترات) مصافحة التخاطب بين المرسال والمستقبل والتحكم بعمل الشريحة 8250.



مسجل حالة المودم (MSR, Modem Status Register):

COM1: 0x3FE | COM2: 0x2FE

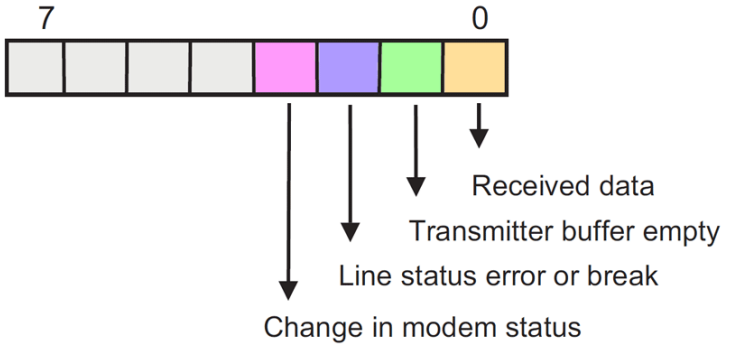
يتم منه قراءة حالة خطوط التحكم حيث أن "dxxx=1" إذا كانت حالة خطوط التحكم قد تغيرت منذ آخر عملية قراءة.



مسجل تفعيل المقاطعات (IER, Interrupt Enable Register):

COM1: 0x3F9 | COM2: 0x2F9

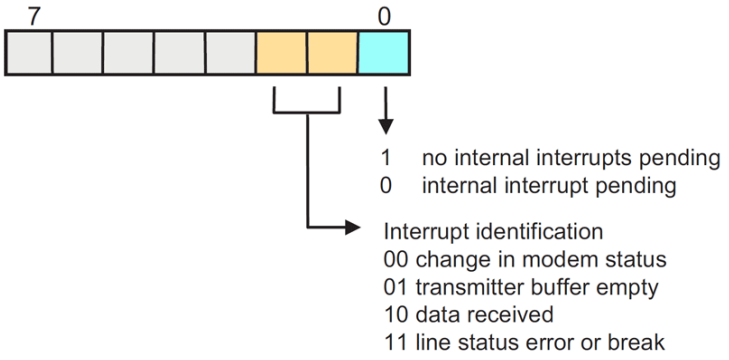
تملك النافذة التسلسلية COM أربعة مقاطعات داخلية (Active "1") موصلة إلى المعالج عن طريق أحد قطب مقاطعة المعالج، هذا القطب هو قطب المقاطعة IRQ4 للمنفذ COM1 وقطب المقاطعة IRQ3 للمنفذ COM2.



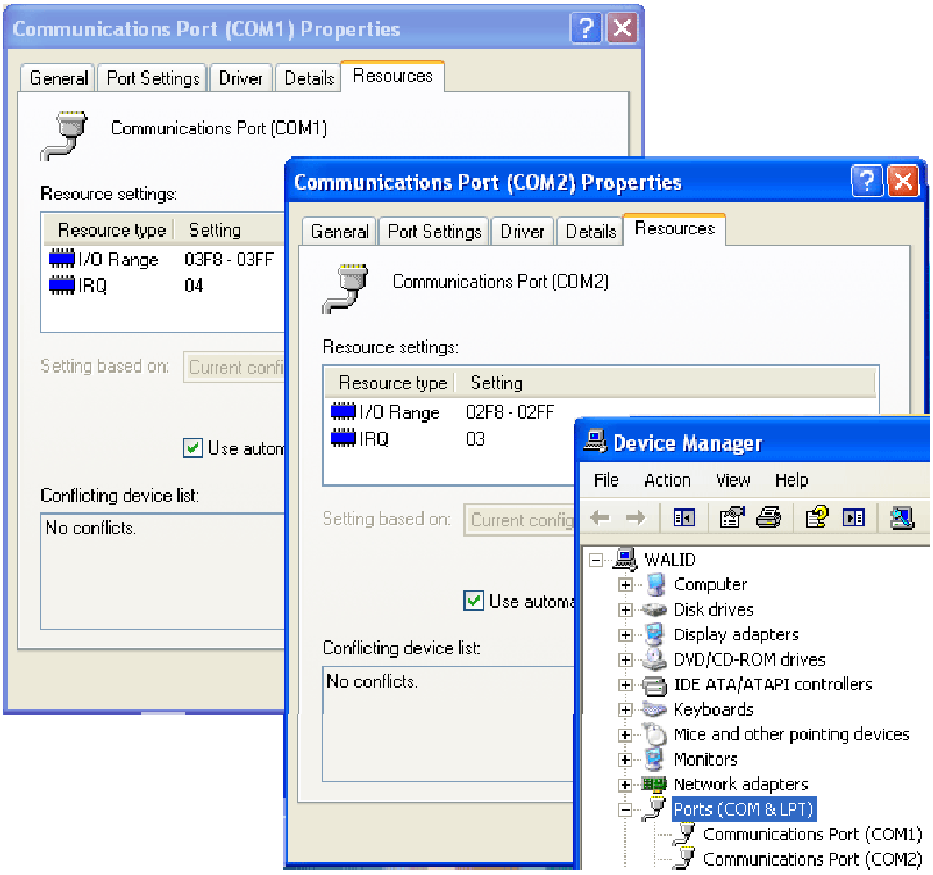
مسجل التعرف لهوية المقاطعة (IIR, Interrupt Identifcation Register):

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:

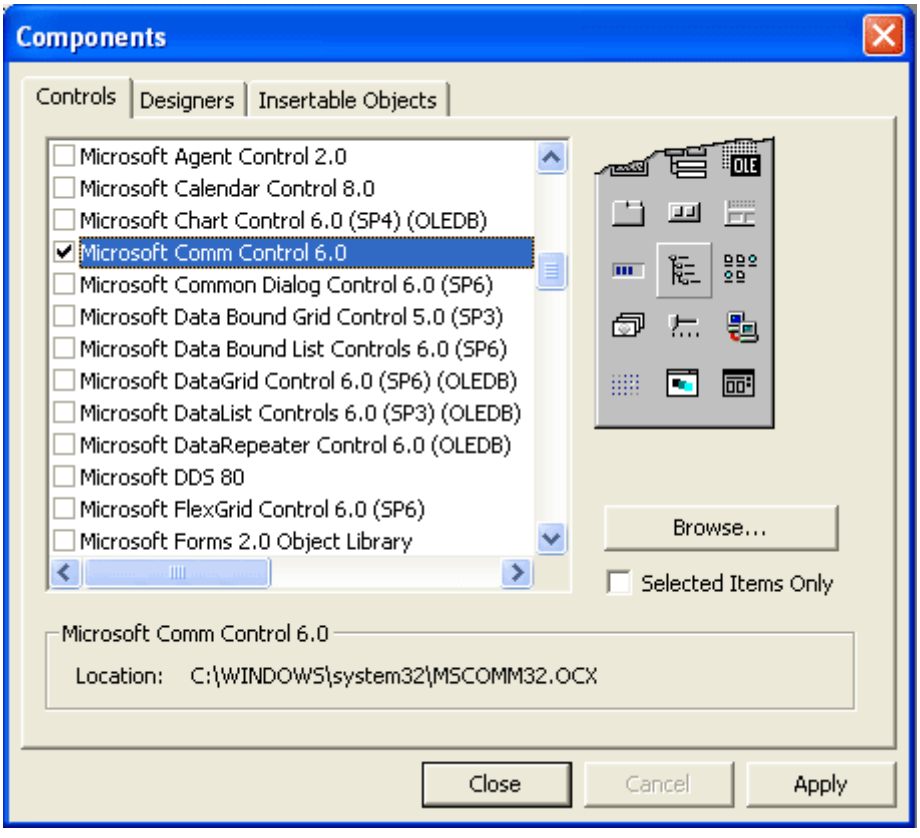
إن التعامل مع المسجلات بشكل مستقل يعتبر معقداً بعض الشيء، لذلك توفر البيئات البرمجية المرئية أدوات (ActiveX & OCX Components) تمكن المبرمج من القراءة والكتابة من مسجلات المنفذ بشكل مباشر كذلك استثمار المقاطعات والأحداث دون الحاجة إلى الوصول البرمجي المباشر لـ Bios، بالإضافة إلى إمكانية إعداد بارامترات المنفذ بشكل مبسط جداً.

إن هذه الأدوات تختلف باختلاف البيئة البرمجية المستخدمة أو الشركة المزودة.

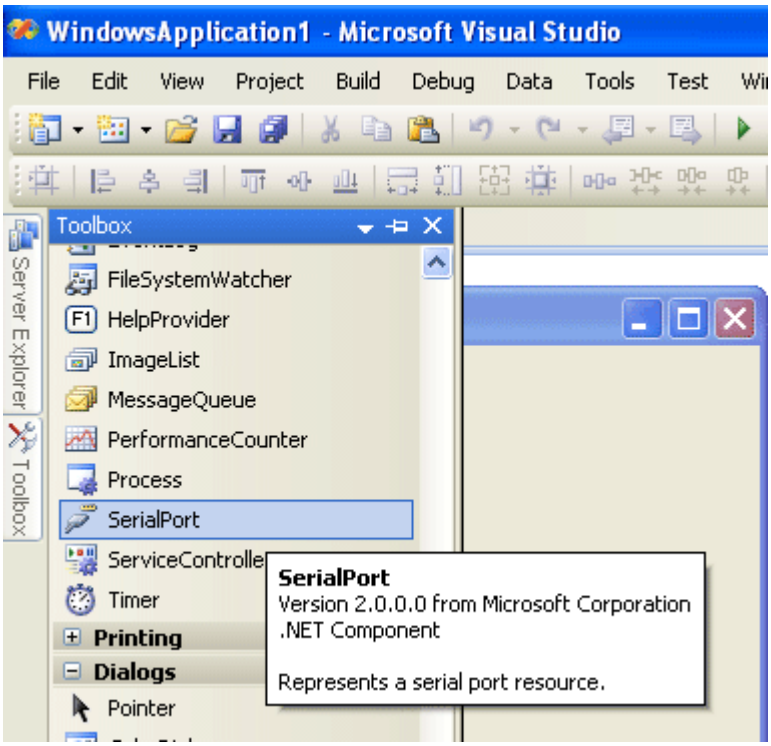
تمتلك بيئة VB6 أداة تسمى "MSComm" وهي عبارة عن "OCX" (MSCOMM32.ocx) تمكّن المستخدم من التخاطب مع منفذ الاتصالات التسلسلية COM بشكل مرن.

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).

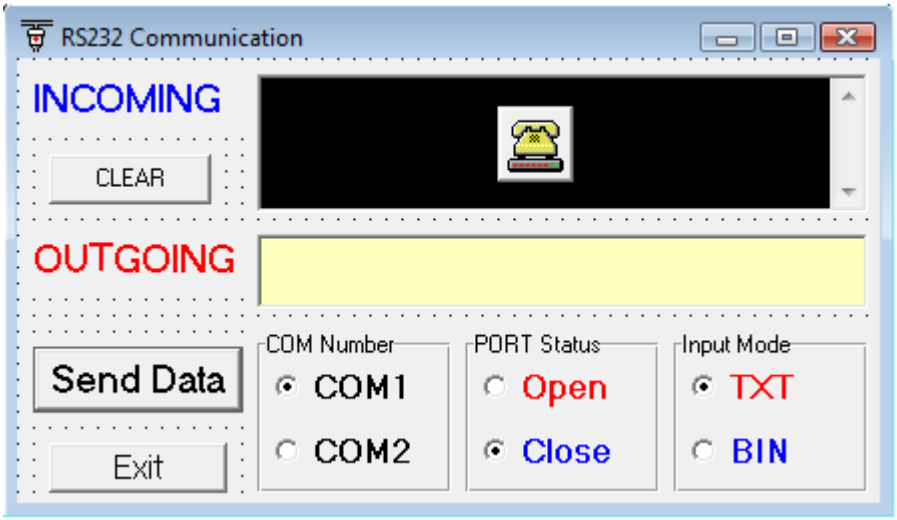


ملاحظة إن التعامل مع الموديل البرمجي للأداة SerialPort مشابه تماماً (إلا من تغييرات في شكل التعليمات) للموديل البرمجي في بيئة VB6.

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:

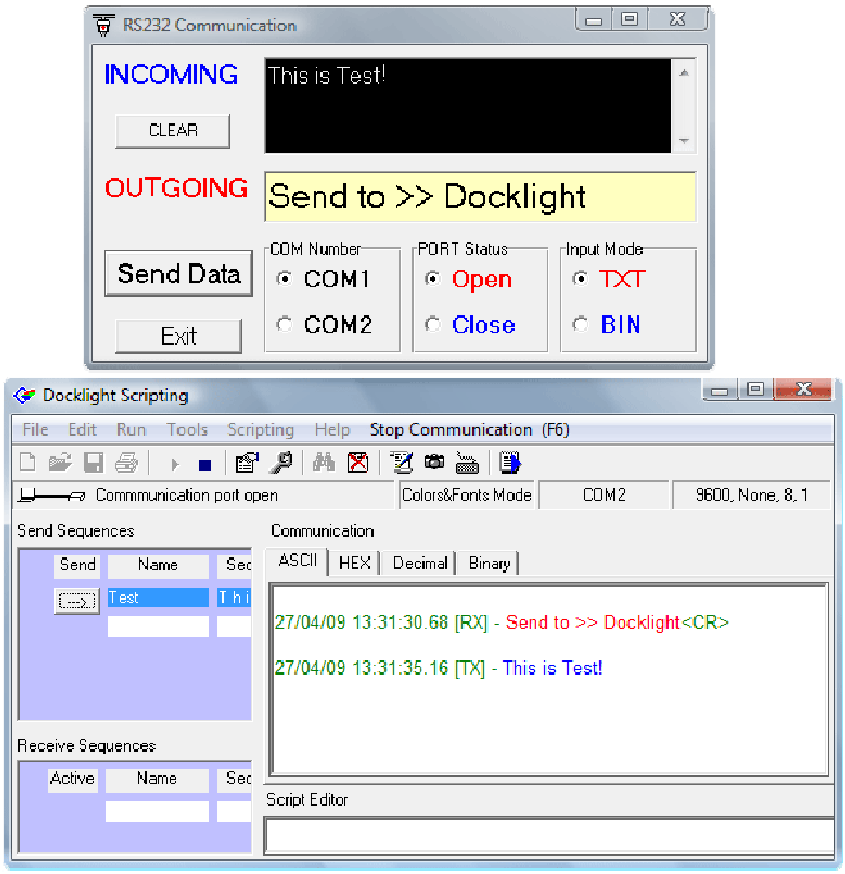
يوجد طريقتان لقراءة البيانات من مسجل الاستقبال للنافذة التسلسلية:

- 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);
```

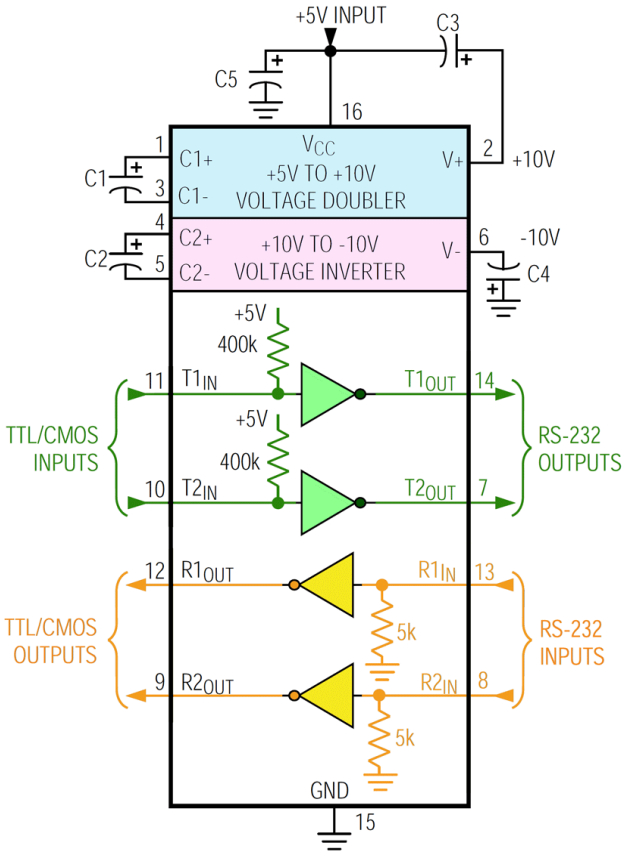
قراءة البيانات بشكل محرفي (TXT) من مسجل الاستقبال.

```
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
```

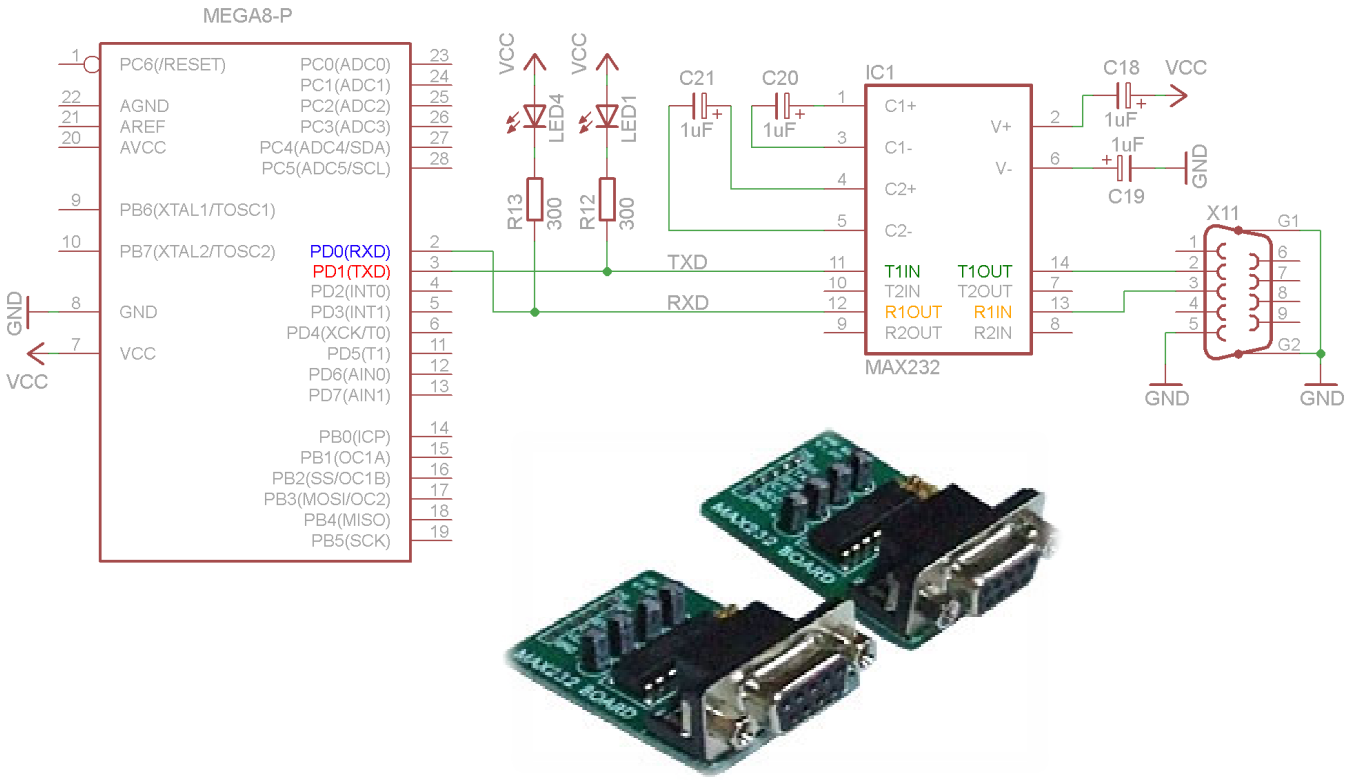


دارات الملائمة TTL <> RS232:

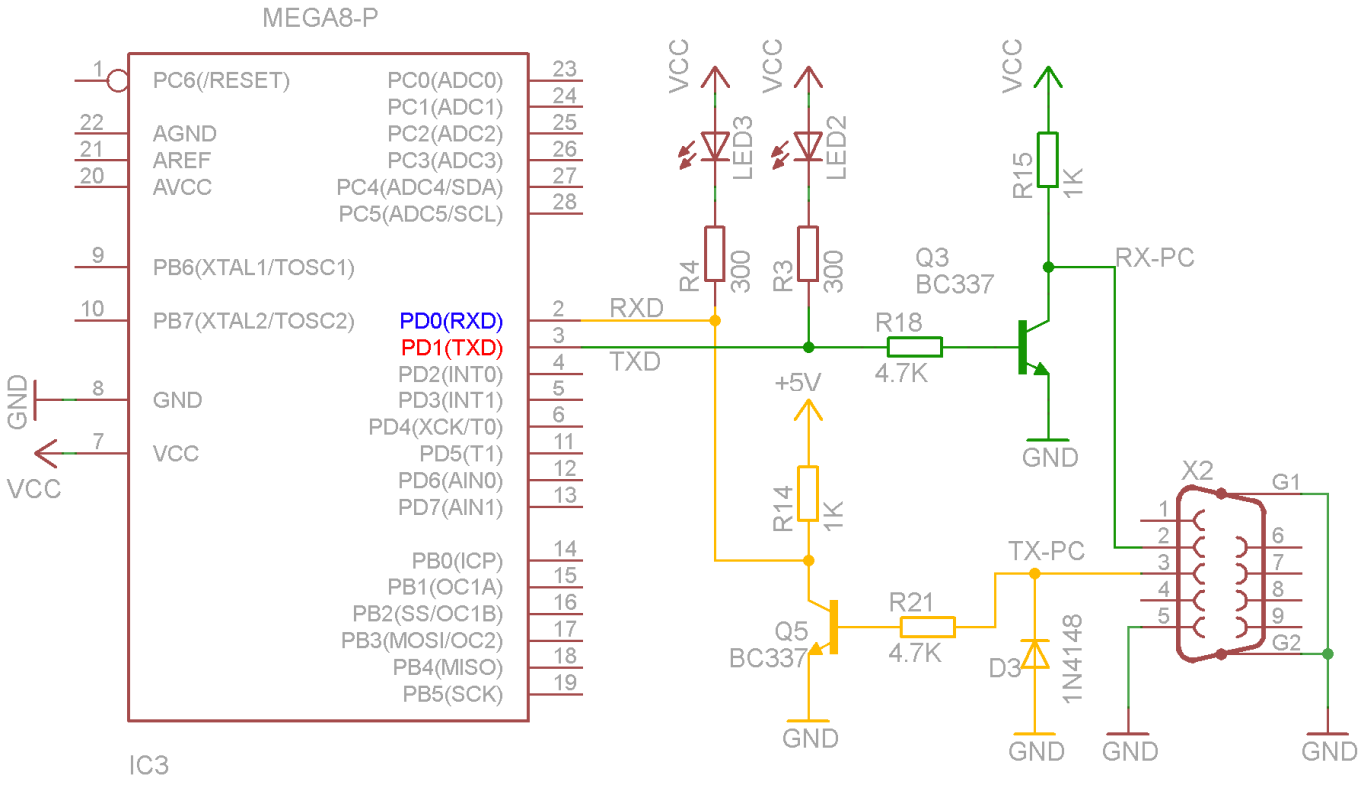
إن المستويات المنطقية للبروتوكول RS232 تختلف عن المستويات المنطقية للمتحكمات المصغرة وللدارات الرقمية الأخرى التي تعتمد المنطق TTL في عملها، وبالتالي نحتاج إلى دارة وسيطية (Adapter) من أجل الملائمة بين الطرفين.

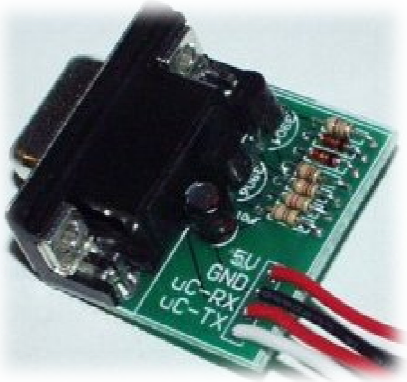
تستخدم الدارة المتكاملة Max232 كدارة تحويل وعزل TTL<>RS232.

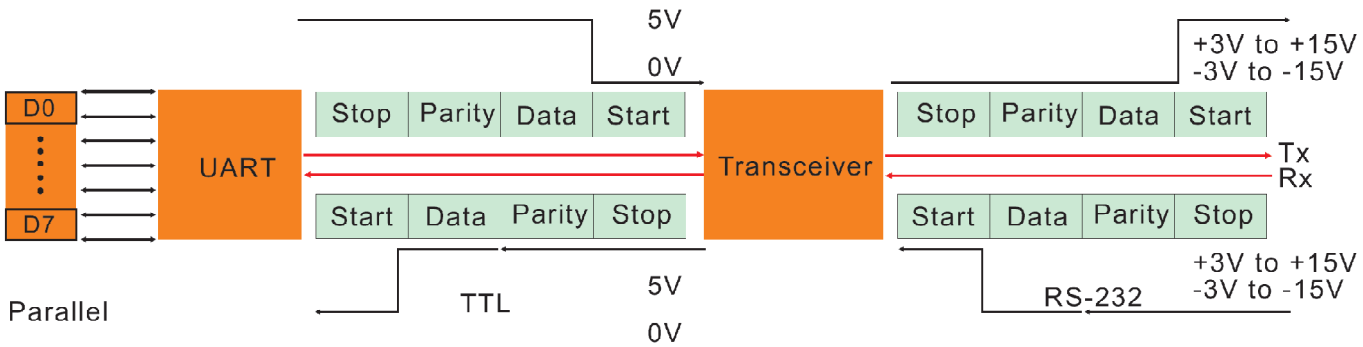
الشكل التالي يبين طريقة تحقيق دارة ملائمة TTL<>RS232 بين منفذ الحاسب التسلسلي (RS232) وبين نافذة تسلسلية (UART) لمتحكم مصغر باستخدام الدارة المتكاملة Max232.



الشكل التالي يبين طريقة تحقيق دارة ملائمة TTL<>RS232 بين منفذ الحاسب التسلسلي (RS232) وبين نافذة تسلسلية (UART) لمتحكم مصغر باستخدام وصلة مفاتيح ترانزستورية.





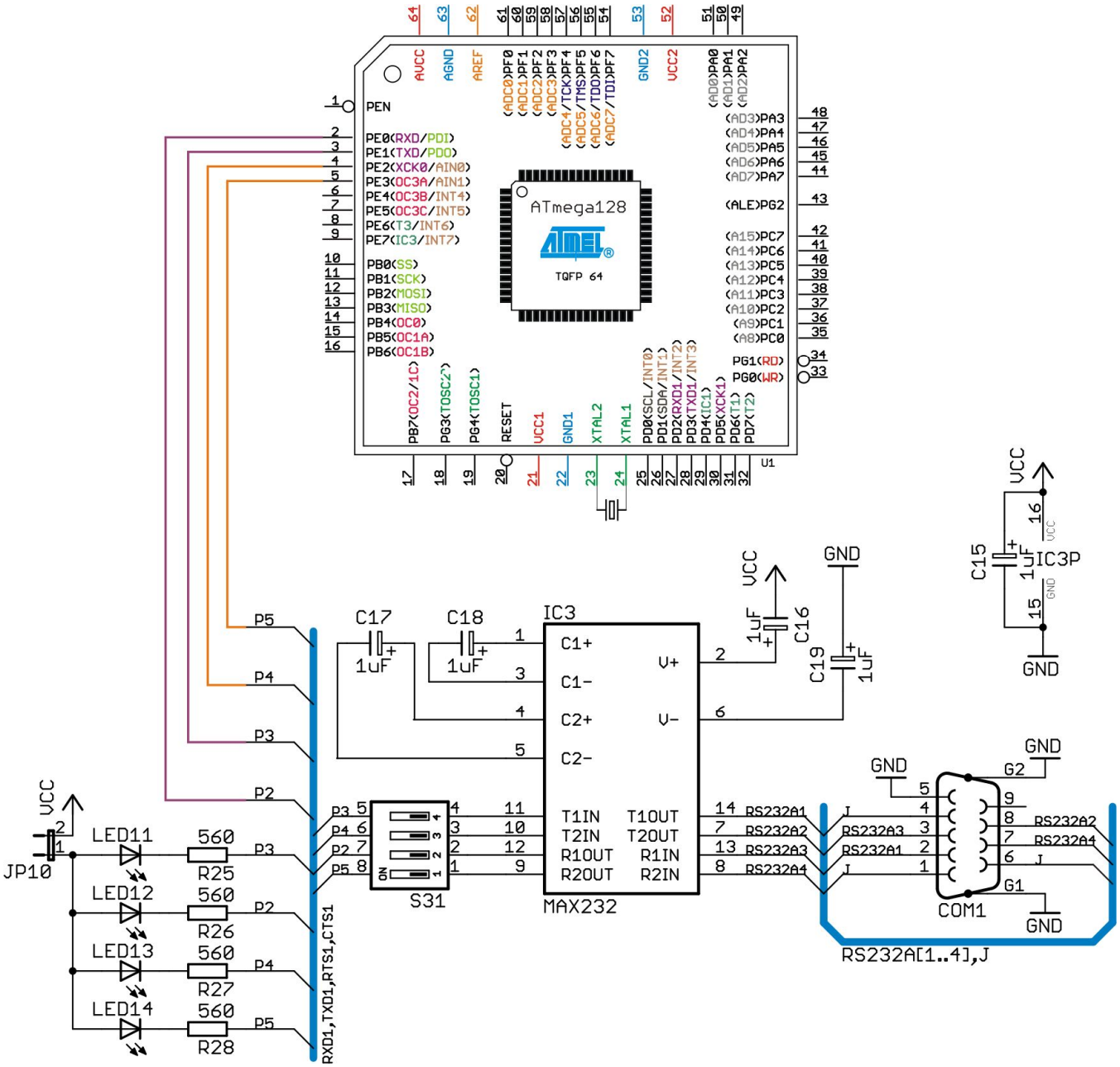


Exp.27: UART Interface التجربة السابعة والعشرون: النافذة التسلسلية UART

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.

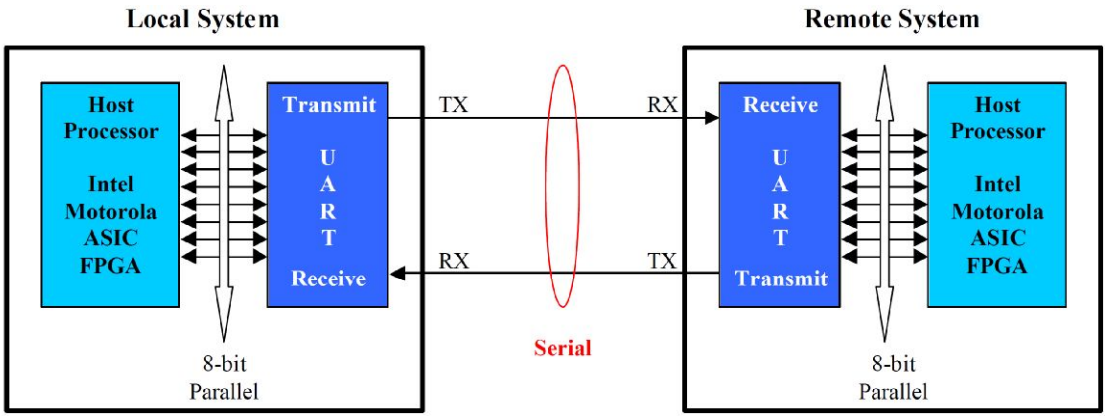
شرح عمل الدارة:

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).

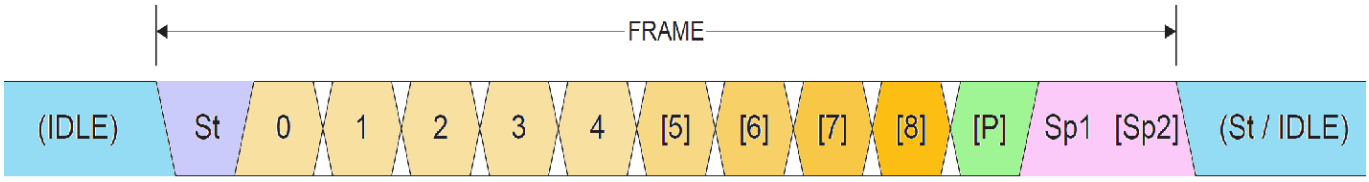


تملك النافذة التسلسلية في متحكمات العائلة AVR على ميزات عديدة وهي تعمل في نمطين مستقلين:

- 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:

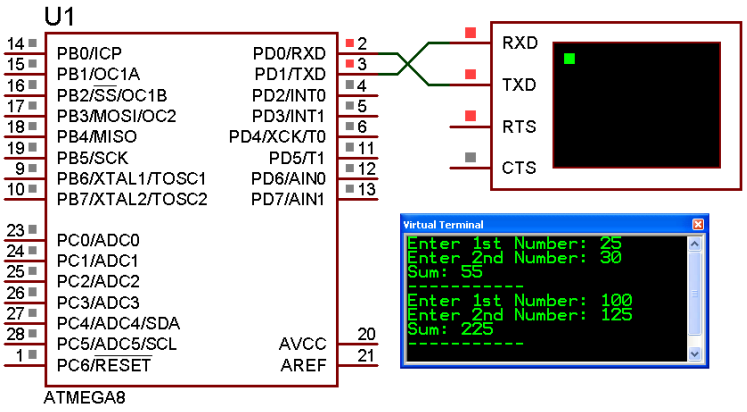
fosc	**$Timeout** = value			
	Value = 100	**Value = 1000**	**Value = 10000**	**Value = 100000**
1MHZ	1.6ms	16ms	160ms	1600ms
2MHZ	0.8ms	8ms	80ms	800ms
4MHZ	0.4ms	4ms	40ms	400ms
8MHZ	0.2ms	2ms	20ms	200ms
16MHZ	0.1ms	1ms	10ms	100ms

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

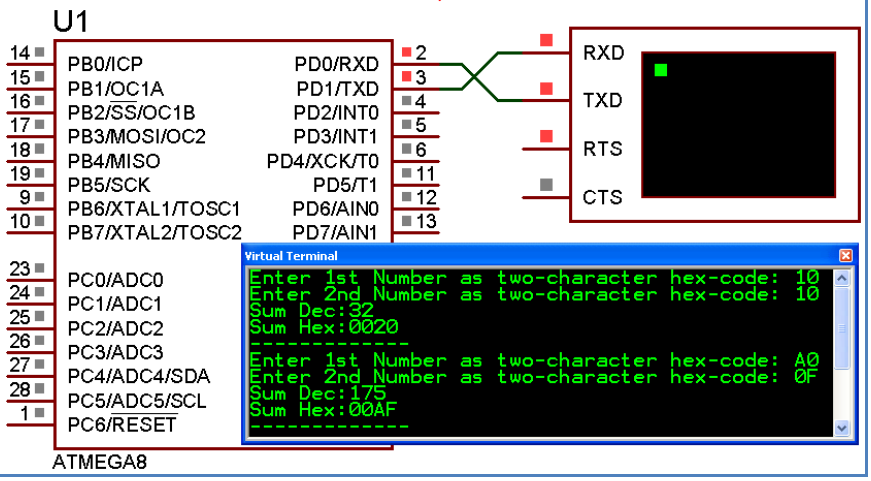
إدخال البيانات باستخدام التعليمة Input:

```
$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
```



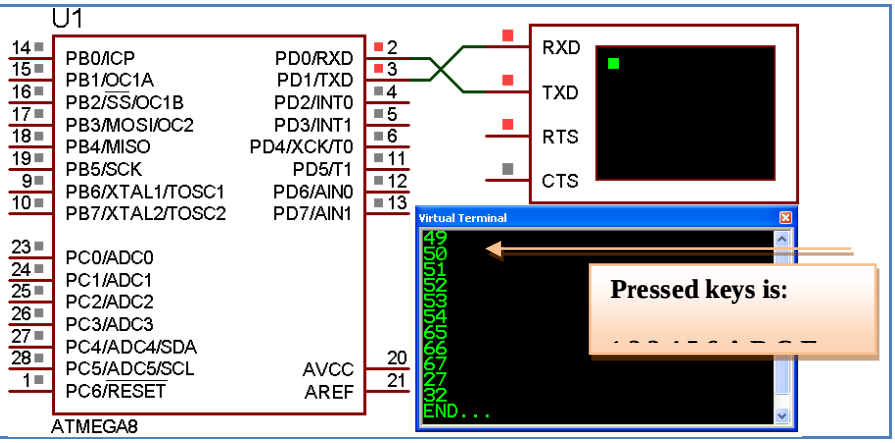
إدخال البيانات باستخدام التعليمة InputHex:

```
$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
```



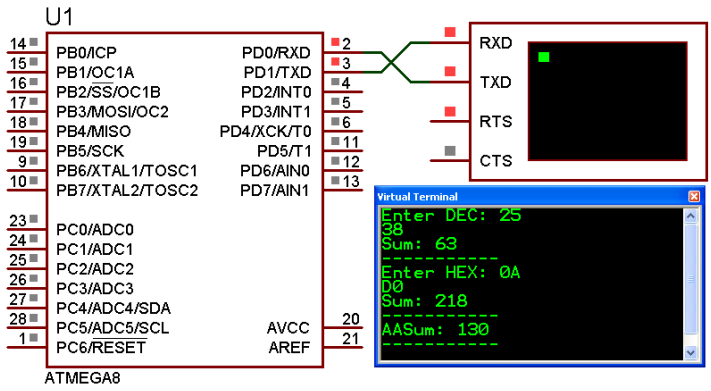
إدخال البيانات باستخدام التعليمة Waitkey():

```
$regfile = "m8def.dat"
$crystal = 4000000
$baud = 9600
'----------------------
Dim Inchar As Byte
'----------------------
Do
   Inchar = Waitkey()
   Print Inchar
Loop Until Inchar = " "
Print "END..."
End
```



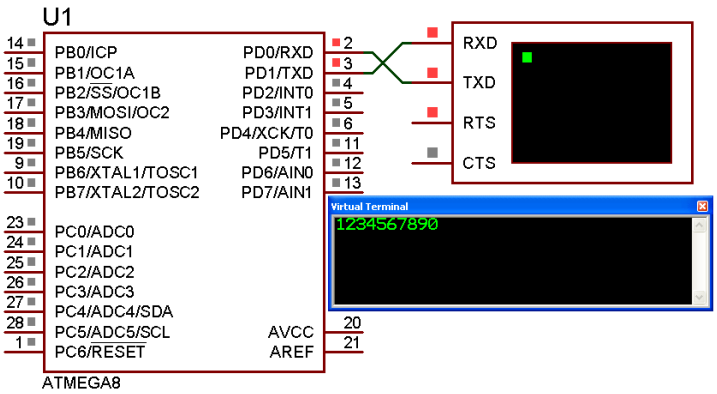
إدخال البيانات بشكل مصفوفي باستخدام التعليمة Input, Inputhex:

```
$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
```



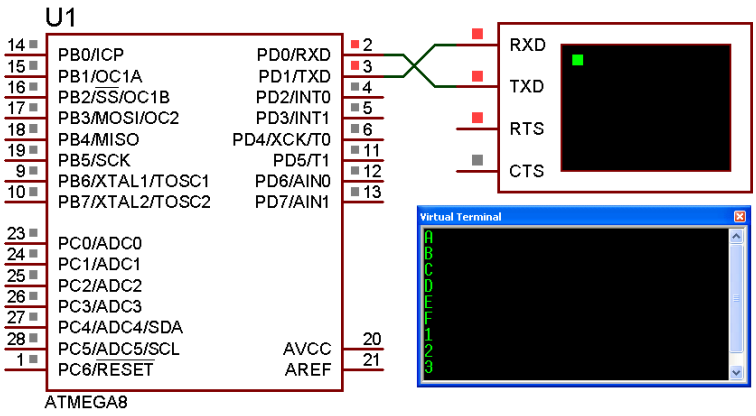
تفعيل بفر دخل وبفر خرج للنافذة التسلسلية UART0:

```
$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
```



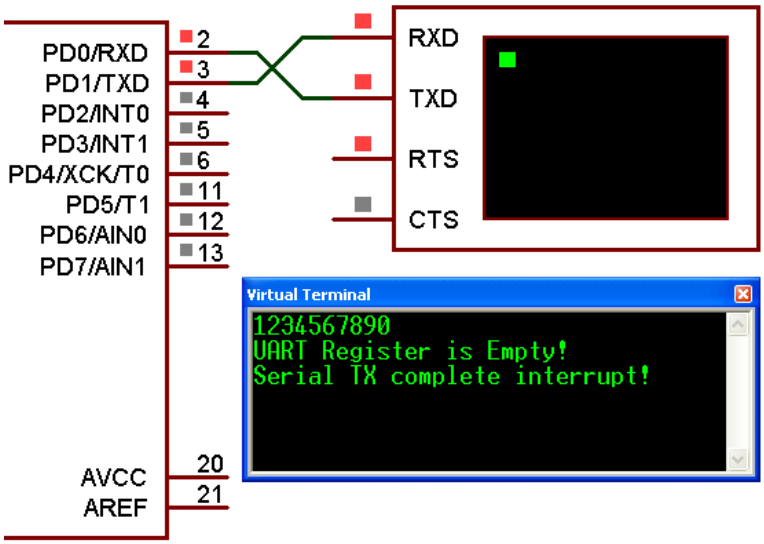
مقاطعة اكتمال استقبال البيانات:

```
$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
```



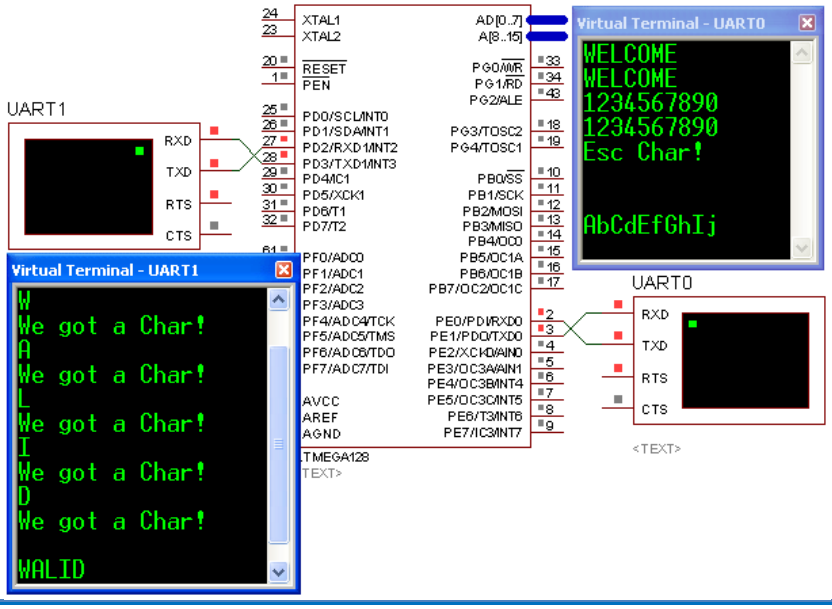
مقاطعات النافذة التسلسلية Urxc, Utxc, Udre:

```
$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
```