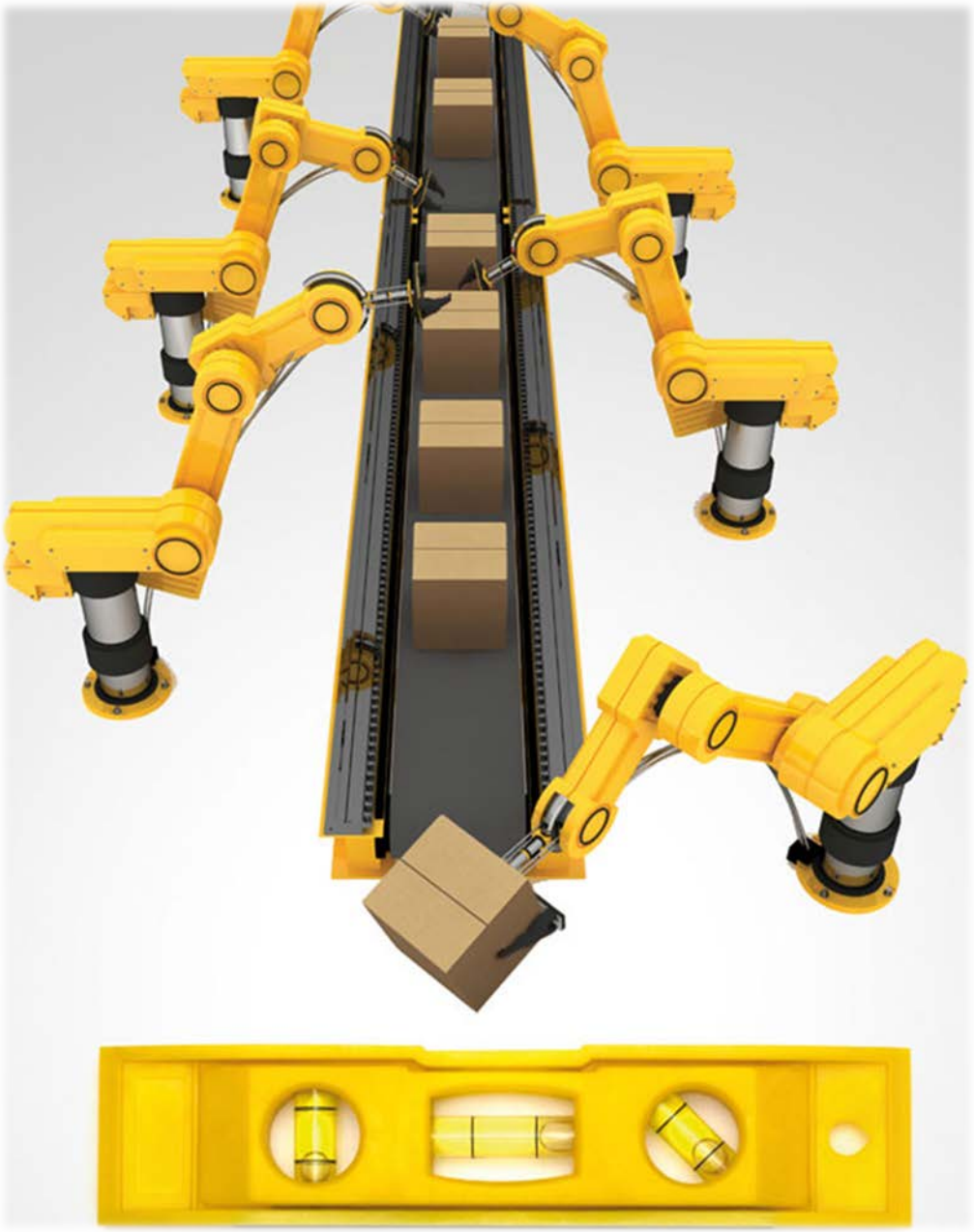
جامعة حلب
كلية الهندسة الكهربائية والإلكترونية
قسم هندسة التحكم والأتمتة

﴿ مادة أتمتة العمليات التكنولوجية ﴾

Technology Processes Automation

السنة الخامسة | قسم تحكم وأتمتة



م. وليد بليد

﴿ الاتصالات التسلسلية Serial Communications ﴾

بروتوكولات 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.

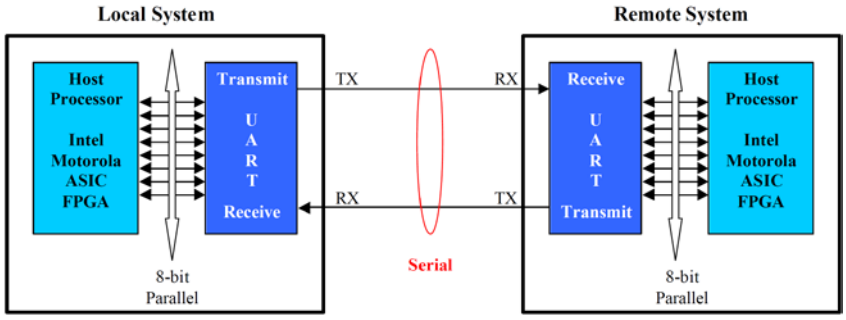
تستخدم الاتصالات التسلسلية على نطاق أوسع بكثير من الاتصالات التفرعية وتمتاز بمناعة عالية ضد الضجيج ونقل لمسافات بعيدة، كما أن حجم الناقل سيكون صغير وكلفته ضئيلة نسبياً مقارنة مع الناقل التفرعية.

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 • InfiniBand • MIDI • DMX512 • Serial ATA • SpaceWire • PCI Express • SONET and SDH • T-1, E-1	o I2C o SPI o PS2	▪ LPT ▪ ISA ▪ EISA ▪ VESA ▪ ATA ▪ SCSI ▪ PCI ▪ PCMCIA ▪ IEEE-1284 ▪ IEEE-488

﴿ النافذة التسلسلية UART ﴾

النافذة التسلسلية UART (Universal Asynchronous Receiver and Transmitter Interface):

تعتبر هذه النافذة من أكثر نوافذ الاتصال التسلسلي استخداماً في الأنظمة الرقمية ومبدأ عملها وكذلك بروتوكولها متوافق تماماً مع البروتوكول RS232 إلا أن المستويات المنطقية فيها وفق المنطق TTL، 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).

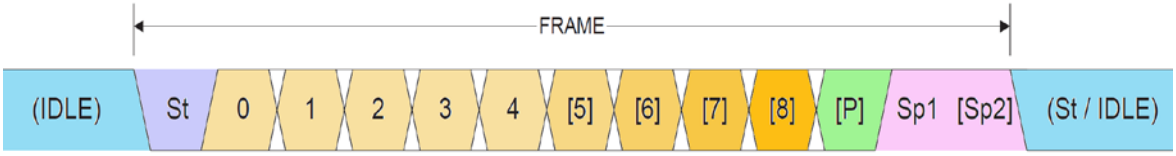


تملك النافذة التسلسلية في متحكمات العائلة AVR على ميزات عديدة وهي تعمل في نمطين مستقلين:

- ✓ 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).

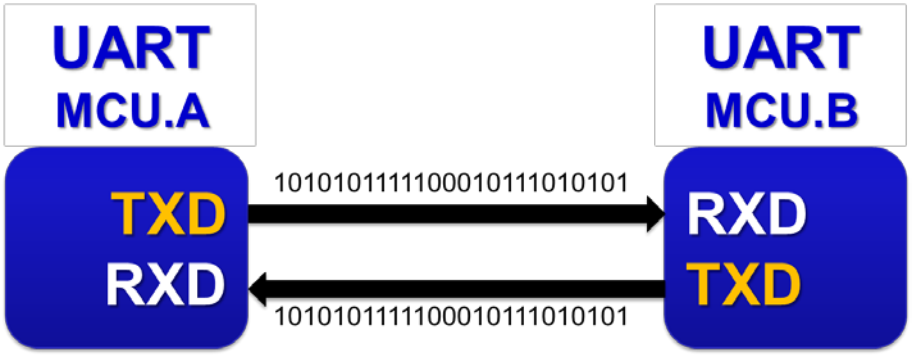
P: Parity bit (Can be odd or even)

Sp: Stop bit, always high.

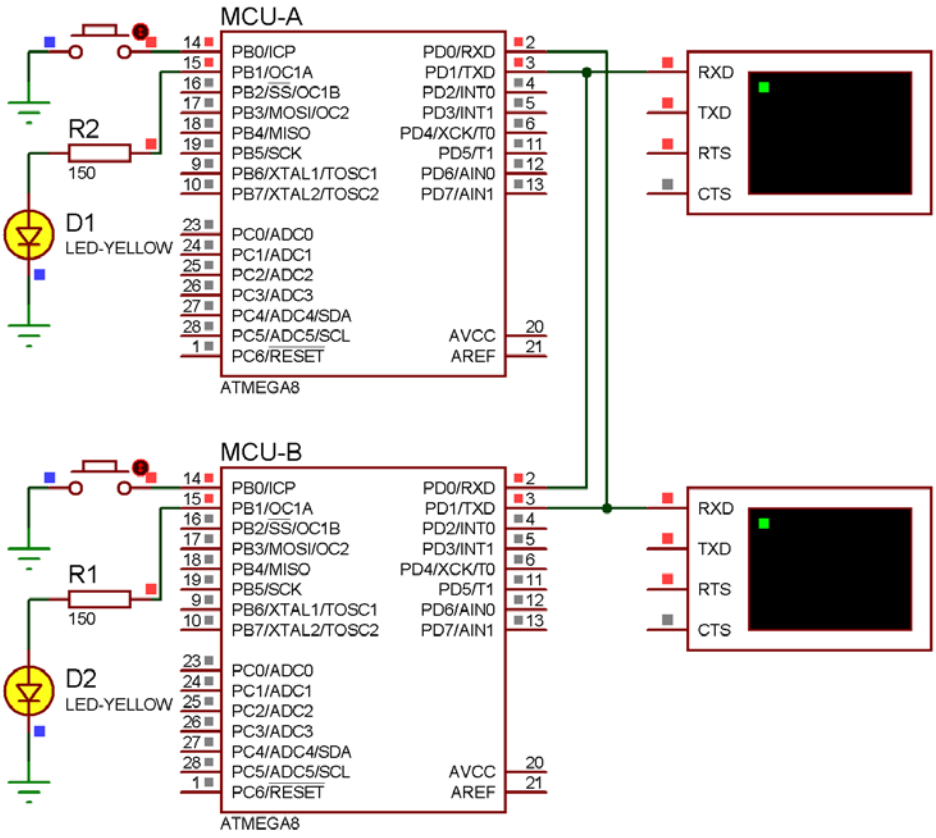
IDLE: No transfers on the communication line (RxD or TxD), IDLE line is high.

تطبيق1: ربط متحكمي AVR من خلال النافذة التسلسلية UART . . .

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



الشكل التالي يبين طريقة الوصل للنافذة التسلسلية بين المتحكمين...



برنامج المتحكم MCU-A في البيئة Bascom-AVR:

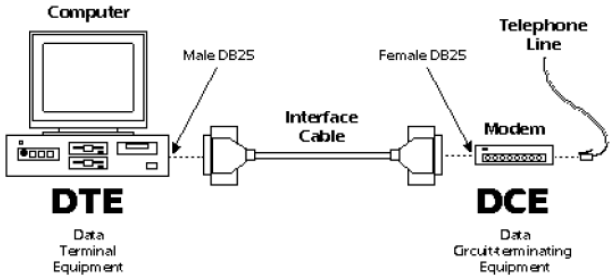
```
'-----------------------[Definitions]
$regfile = "m8def.dat"
$crystal = 8000000
$baud = 9600

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

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

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

برنامج المتحكم MCU-B في البيئة Bascom-AVR:

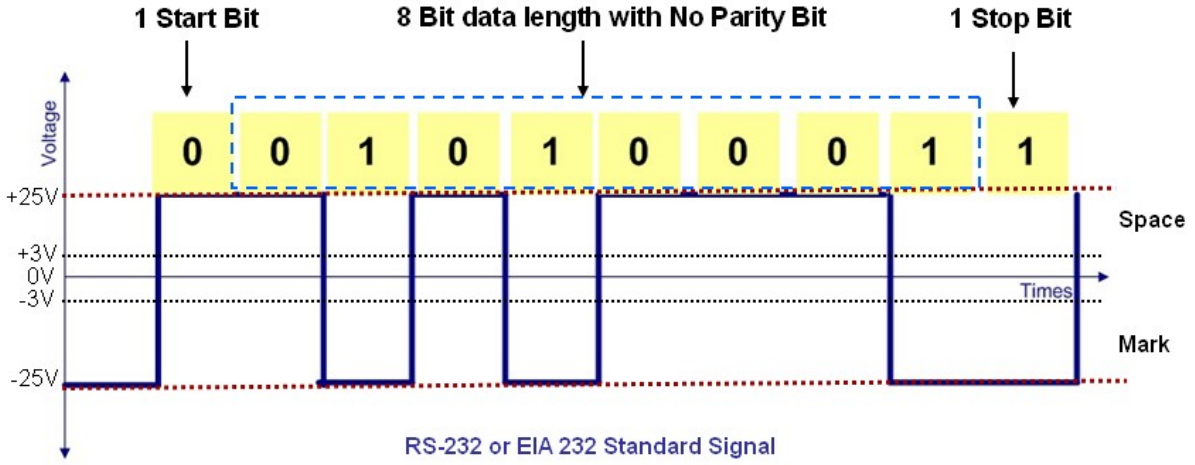
```
'-----------------------[Definitions]
$regfile = "m8def.dat"
$crystal = 8000000
$baud = 9600

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

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

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

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

﴿ بروتوكول الاتصال التسلسلي RS232 ﴾

هو عبارة عن برتوكول اتصال تسلسلي غير متواقت يستخدم من أجل الربط بين طرفيتين، تسمى الأولى DTE وتسمى الثانية DCE.



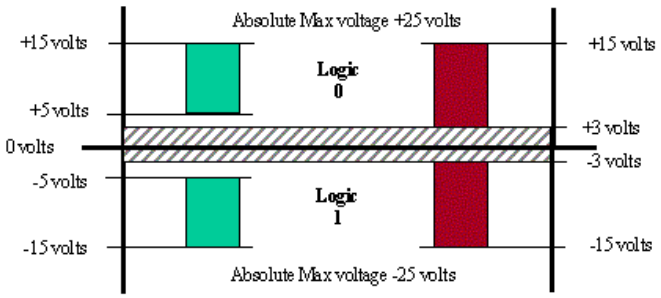
يتم إرسال كل بايت كحزمة مؤلفة من مجموعة بتات على الشكل التالي:



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**.



الميزات والمساوئ لبرتوكول الاتصال RS232:

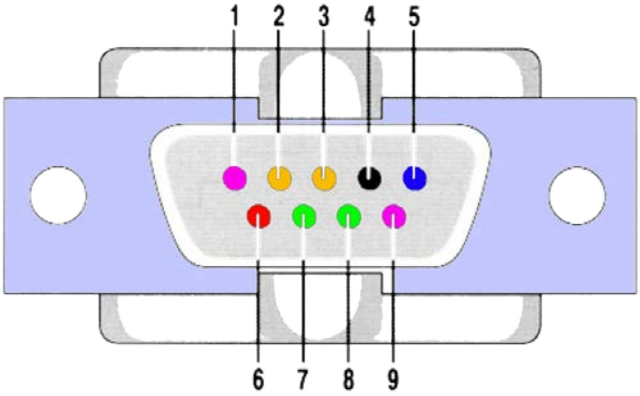
المساوئ (Disadvantages)	المحاسن (Advantages)
✗ مناسب فقط من أجل الربط بين System-to-System أكثر من كونه قابلاً للربط بين Chip-2-Chip أو من أجل تطبيقات Chip-2-Sensor.	✓ بروتوكول اتصال شائع الاستخدام في كثير من التطبيقات ومعتمد من قبل العديد من الشركات.
✗ معدل نقل بيانات منخفض جداً من أجل مسافة اتصال كبيرة.	✓ 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.	✓ مناعة ضد الضجيج بسبب الجهد المرتفع نسبياً (±25) للمستويات المنطقية (“1” , “0”).
✗ مخصص للربط بين Single Master/Single Slave.	✓ سهل البناء والبرمجة ومتوفر برمجياً وككيان 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):

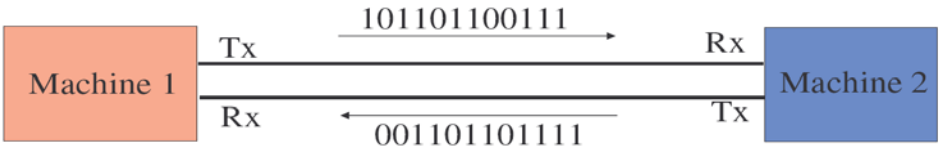
قطب تحكم يقوم بإعلام الطرفية الرئيسية (DTE) أنه استلم إشارة الإعلام بإرسال البيانات السابقة ويمكنها الآن أن تبدأ بإرسال البيانات إلى الطرفية الثانوية (DCE)، وبالتالي يمكن استخدام هذه الإشارة من أجل تفعيل دارة الاستقبال قبل إرسال أي إشارة. فعال ("0") عندما تكون الطرفية الثانوية جاهزة لاستلام البيانات، 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.

تحقيق اتصال بين طرفيتين في RS232:

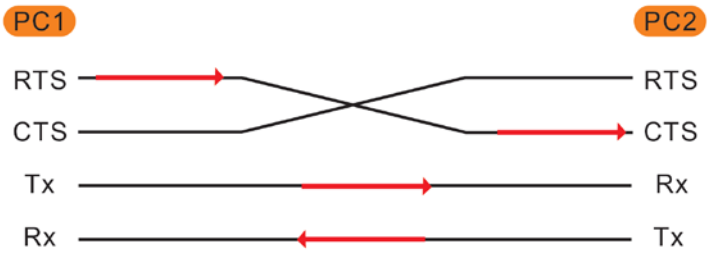
عموماً، فإنه من أجل تحقيق اتصال بين طرفيتين بدون مصافحة يكفي توصيل قطب الإرسال "**TxD**" والاستقبال "**RxD**" على التوازي المتعاكس كما في الشكل التالي:



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.



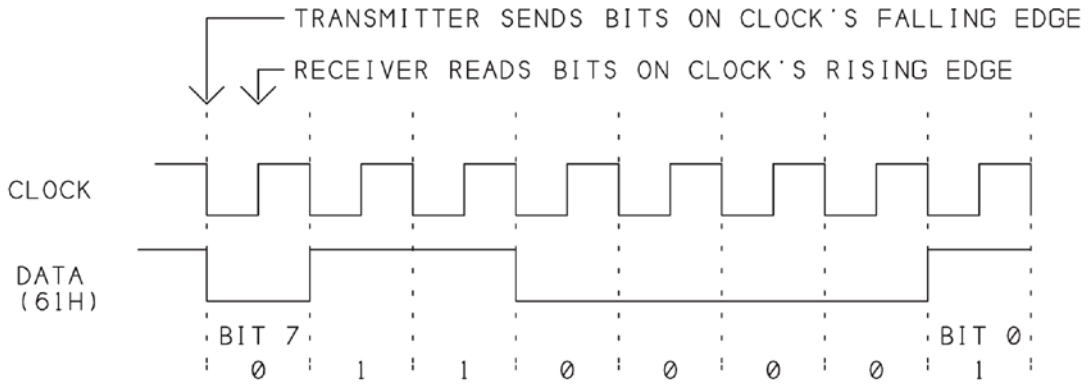
المواصفات الفنية للبروتوكول RS232:

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.

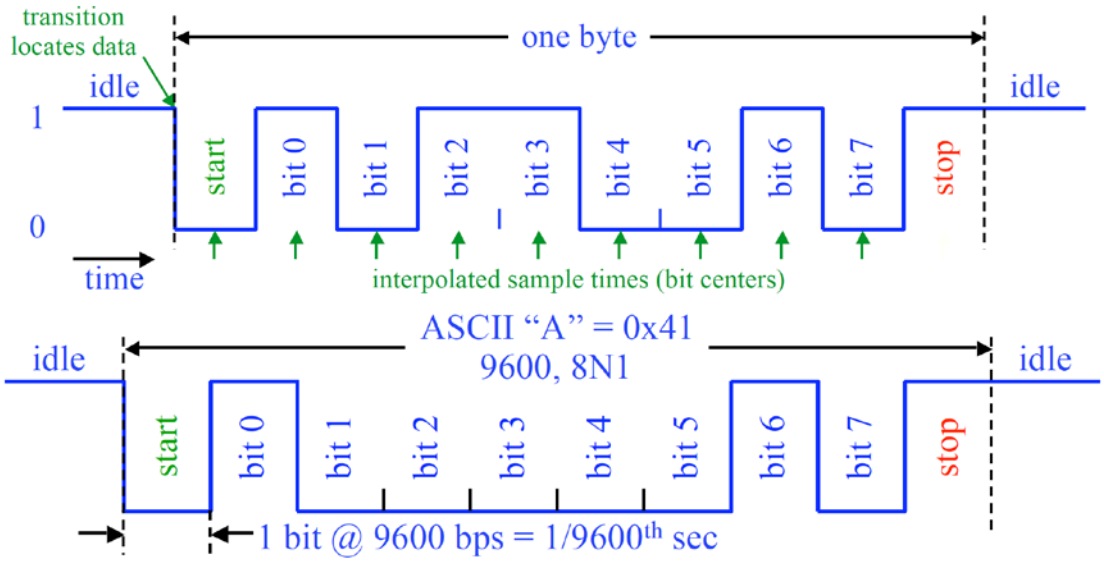
مفهوم الاتصالات التسلسلية المتزامنة (Synchronized) وغير المتزامنة (Asynchronous):

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 بحيث 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.

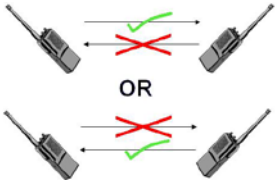
ثانياً: اتصالات غير متواقتة (غير متزامنة): لا تحوي على خط تزامن وإنما يتم بدء عملية الإرسال بإرسال بت بدء الإرسال (Start Bit) والذي بدوره يعلم المستقبل أن الذي يليه هو بايت البيانات، وبعدها يتم إرسال البايت المطلوب وتنتهي عملية إرسال البايت بإرسال بت التوقف (Stop Bit) والذي بدوره يعلم المستقبل أن عملية 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:

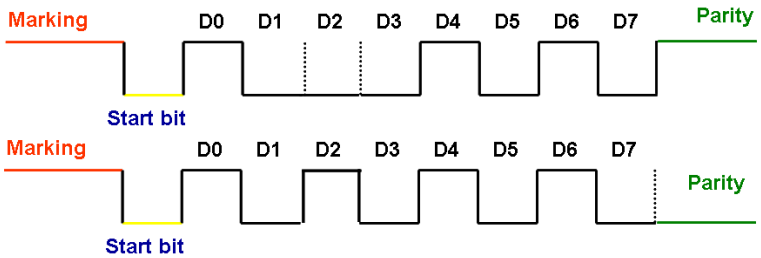
- ✓ تحديد نمط الإرسال: أحادي الاتجاه (Half-Duplex) أو ثنائي الاتجاه (Full-Duplex).
- ✓ تحديد عدد البتات لكل محرف: 6, 7 or 8 bit.
- ✓ تحديد معدل سرعة الإرسال (Baud Rate).
- ✓ تحديد استخدام أو عدم استخدام خانة فحص الإيجابية (Parity Bit)، وفي حال الاستخدام يجب تحديد نمط فحص خانة الإيجابية (Even or Odd).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



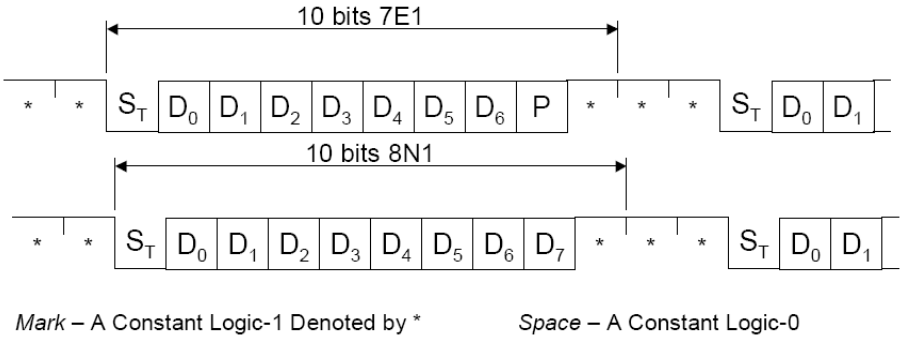
في حال كون خانة الإيجابية "Even" فإن هذه الخانة تملك القيمة "0" إذا كان عدد الواحدات في البايت المرسل زوجي وإلا فستصبح "1". الأمثلة التالية توضح ذلك.

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

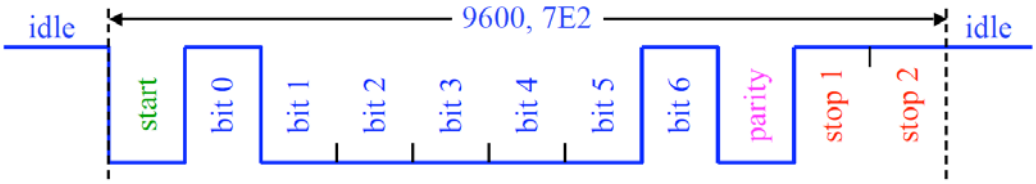
في حال كون خانة الإيجابية "Odd" فإن هذه الخانة تملك القيمة "0" إذا كان عدد الواحدات في البايت المرسل فردي وإلا فستصبح "1". الأمثلة التالية توضح ذلك.

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

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



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…

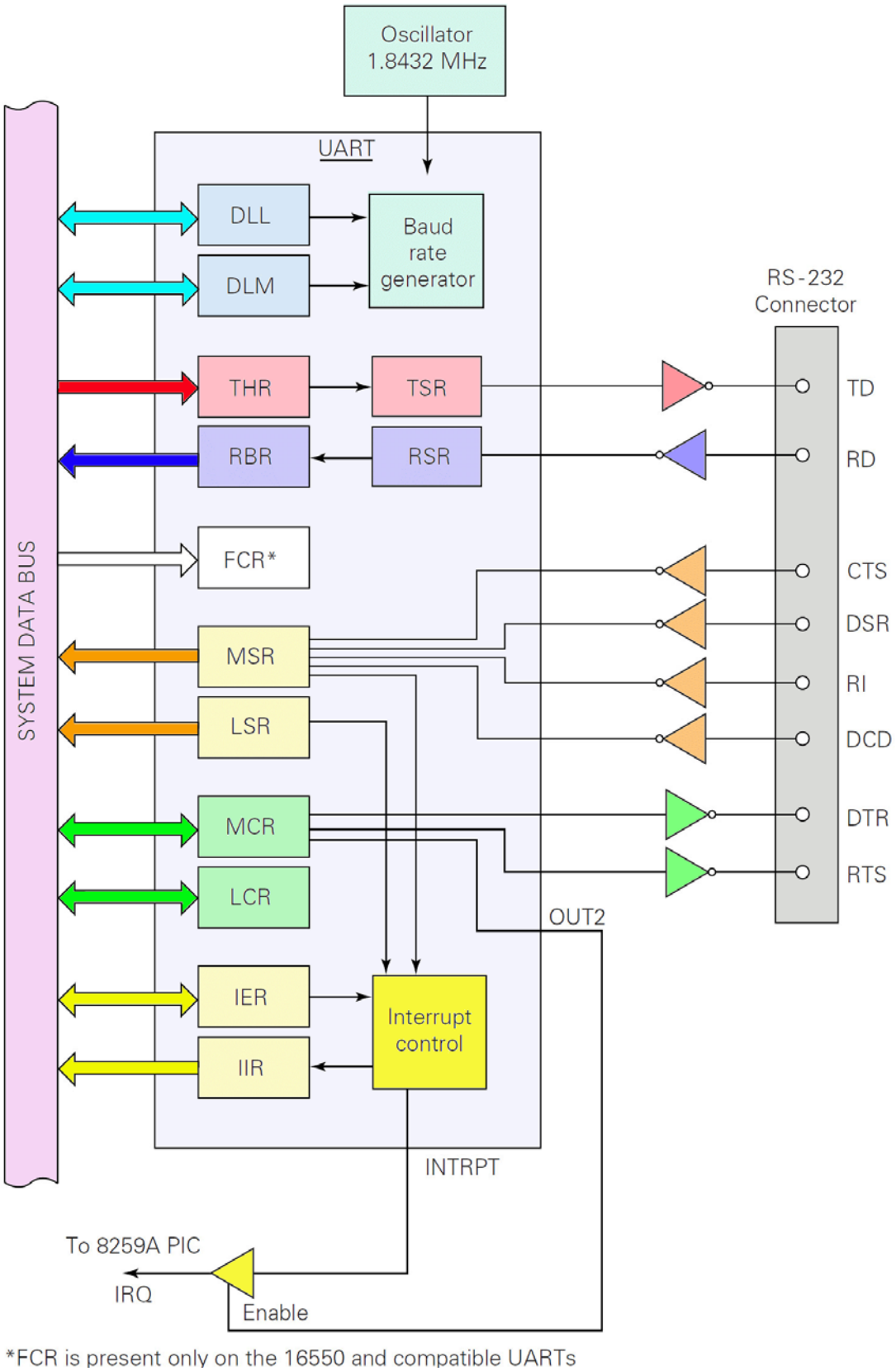
إن الزمن اللازم لإرسال بت واحد يعطى بالعلاقة التالية:

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

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:



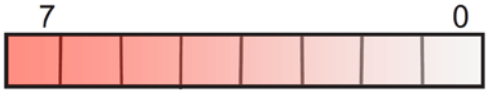
المسجلات الداخلية للنافذة التسلسلية RS232 في الحاسب:

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

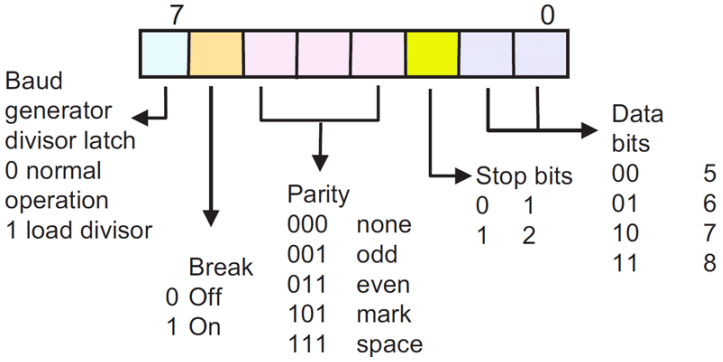
يتم منه قراءة البيانات الواردة عبر القطب RxD وإرسال البيانات الصادرة عبر القطب TxD.



مسجل التحكم بالخط (**LCR**, Line Control Register):

COM1: 0x3FB | COM2: 0x2FB

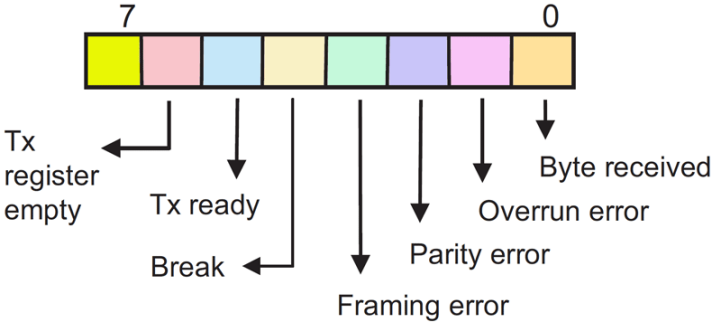
يتم فيه تعيين إعدادات (بارامترات) إطار البيانات.



مسجل حالة الخط (**LSR**, Line Status Register):

COM1: 0x3FD | COM2: 0x2FD

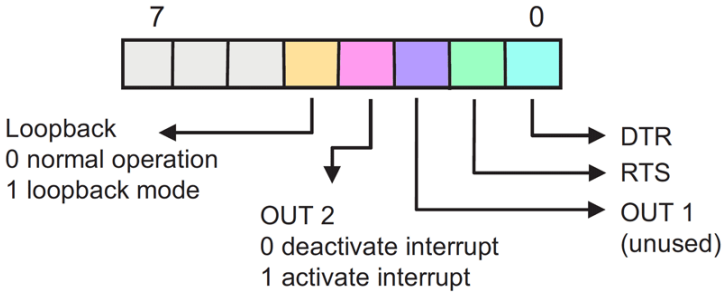
يتم منه قراءة حالة العمليات الجارية على الخط من أجل كشف الأخطاء والاستعلام عن حالة مسجل الإرسال.



مسجل التحكم بالمودم (**MCR**, Modem Control Register):

COM1: 0x3FC | COM2: 0x2FC

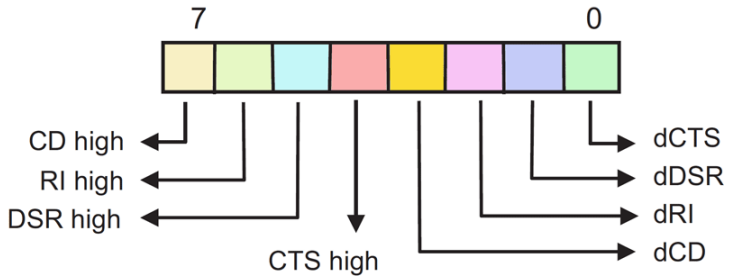
يتم فيه تعيين إعدادات (بارامترات) مصافحة التخاطب بين المرسال والمستقبل والتحكم بعمل الشريحة 8250.



مسجل حالة المودم (**MSR**, Modem Status Register):

COM1: 0x3FE | COM2: 0x2FE

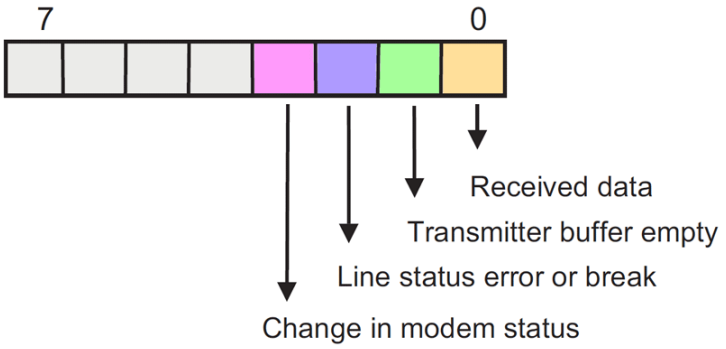
يتم منه قراءة حالة خطوط التحكم حيث أن "**dxxx=1**" إذا كانت حالة خطوط التحكم قد تغيرت منذ آخر عملية قراءة.



مسجل تفعيل المقاطعات (**IER**, Interrupt Enable Register):

COM1: 0x3F9 | COM2: 0x2F9

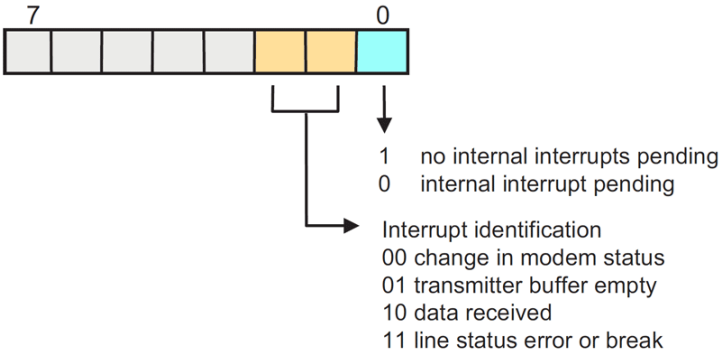
تملك النافذة التسلسلية COM أربعة مقاطعات داخلية (Active "**1**") موصلة إلى المعالج عن طريق أحد قطب مقاطعة المعالج، هذا القطب هو قطب المقاطعة IRQ4 للمنفذ COM1 وقطب المقاطعة IRQ3 للمنفذ COM2.



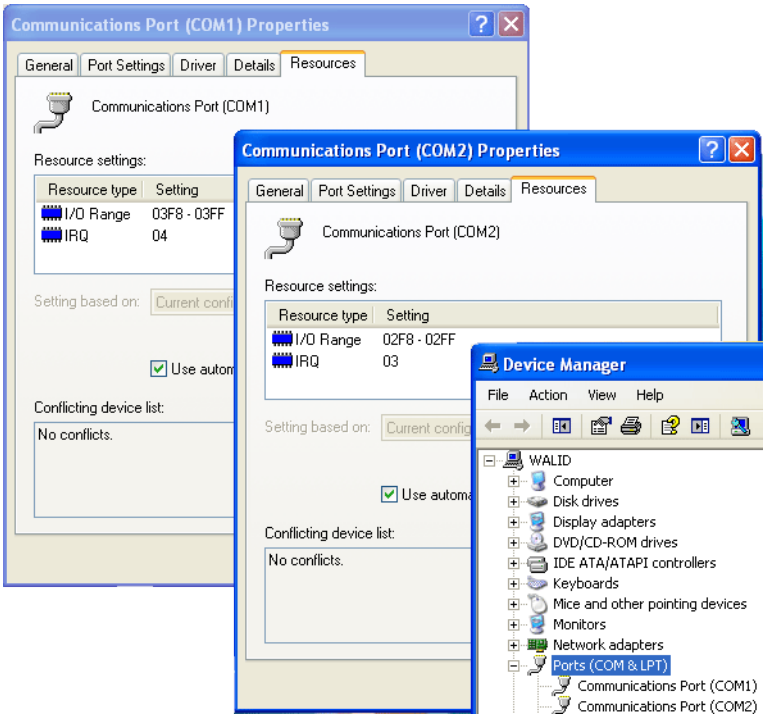
مسجل التعرف لهوية المقاطعة (**IIR**, Interrupt Identification Register):

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:

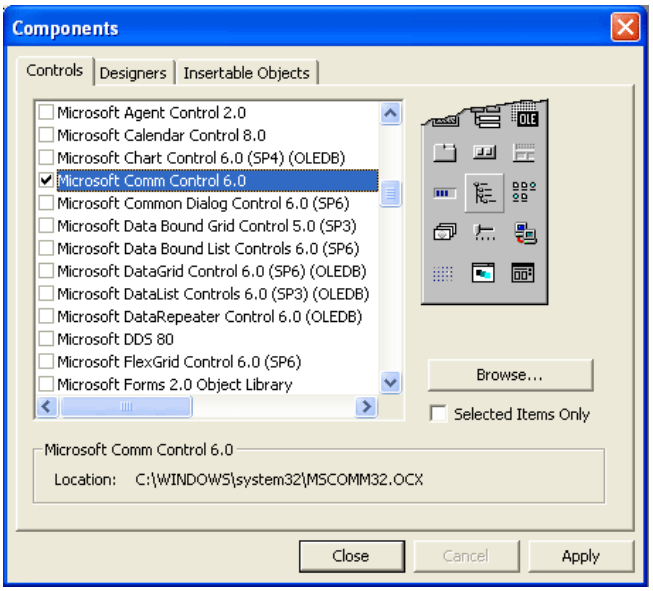
إن التعامل مع المسجلات بشكل مستقل يعتبر معقداً بعض الشيء، لذلك توفر البيئات البرمجية المرئية أدوات (ActiveX & OCX Components) تمكن المبرمج من القراءة والكتابة من مسجلات المنفذ بشكل مباشر كذلك استثمار المقاطعات والأحداث دون الحاجة إلى الوصول البرمجي المباشر لل Bios، بالإضافة إلى إمكانية إعداد بارامترات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

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

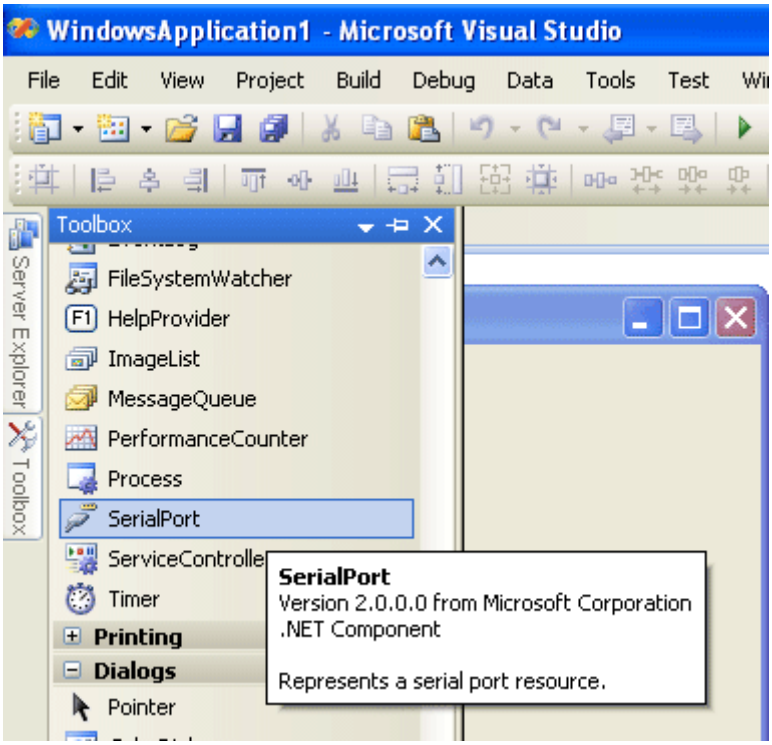
تمتلك بيئة VB6 أداة تسمى "MSComm" وهي عبارة عن "OCX" (MSCOMM32.ocx) تمكّن المستخدم من التخاطب مع منفذ الاتصالات التسلسلية COM بشكل مرن.

ملاحظة: إن هذه الأداة يجب تنصيبها 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 الأدوات كما في الشكل أدناه، أما في حال كان المشروع هو "Enterprice Edition" فسوف يتم تحميل جميع الأدوات المتقدمة والقياسية إلى شريط الأدوات.



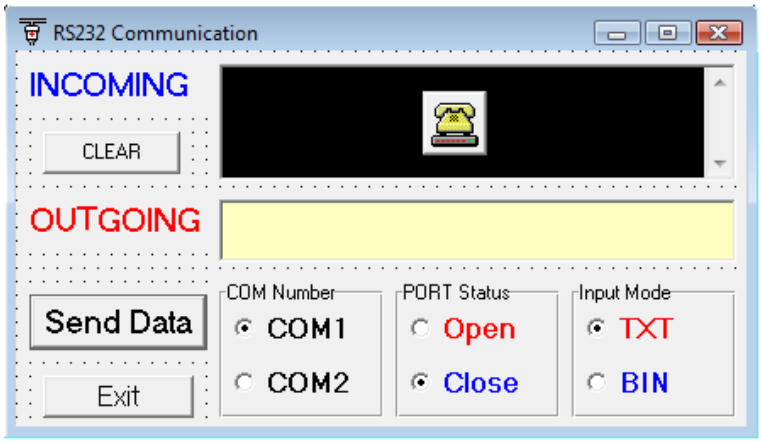
أما بالنسبة للبرمجة في بيئة "Microsoft Visual Studio 2008" فالأمر مشابه تماماً لبيئة VB6 إلا أنّ الأداة أصبحت ضمن شريط الأدوات الأساسية وتدعى "SerialPort" كما أنها يمكن أن تستخدم في أي لغة برمجية داخل بيئة .net. وذلك لأن الواجهة البرمجية والأدوات مشتركة وتختلف اللغة النصية فقط (VB.net, C#.net or C++.net).



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 حانباً 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:

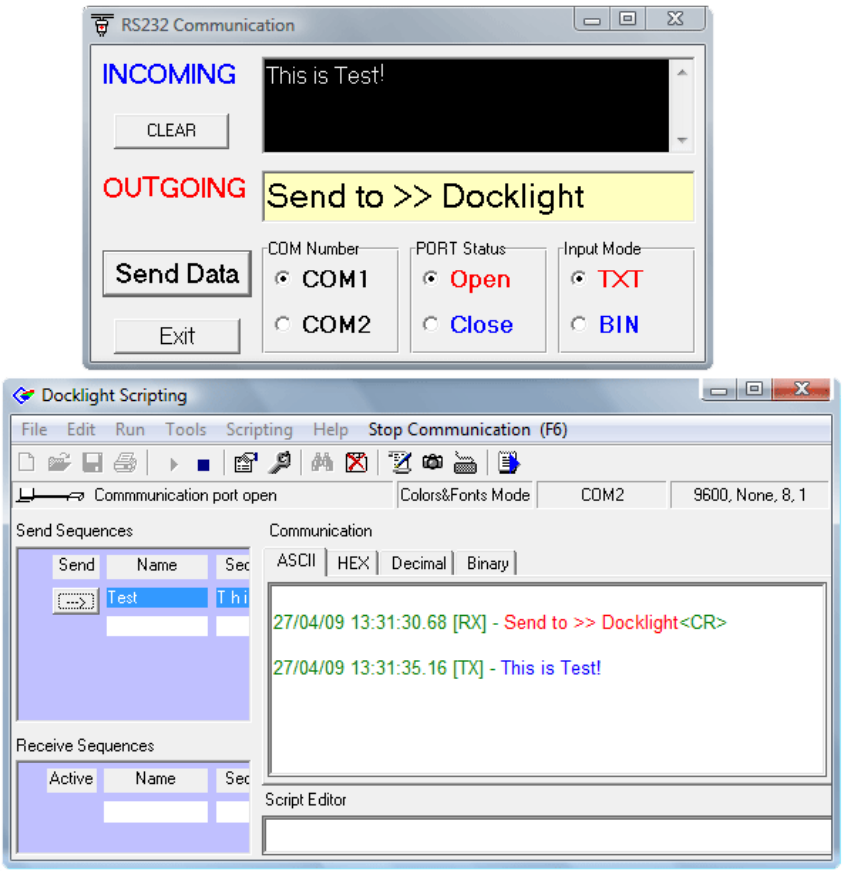
يوجد طريقتان لقراءة البيانات من مسجل الاستقبال للنافذة التسلسلية:

✓ 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);
```

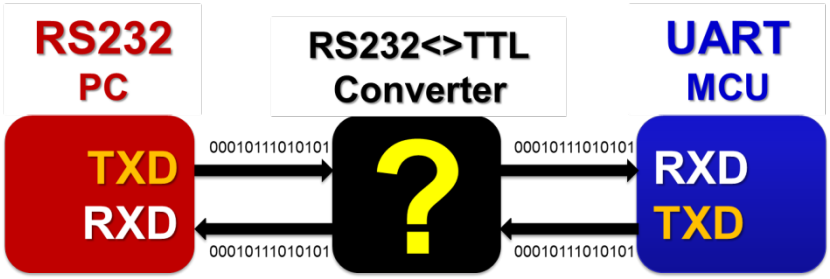
قراءة البيانات بشكل ثنائي (BIN) من مسجل الاستقبال.

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
```

تطبيق1: المطلوب ربط متحكم AVR من خلال النافذة UART (TTL) مع منفذ COM (RS232) . . . . . .

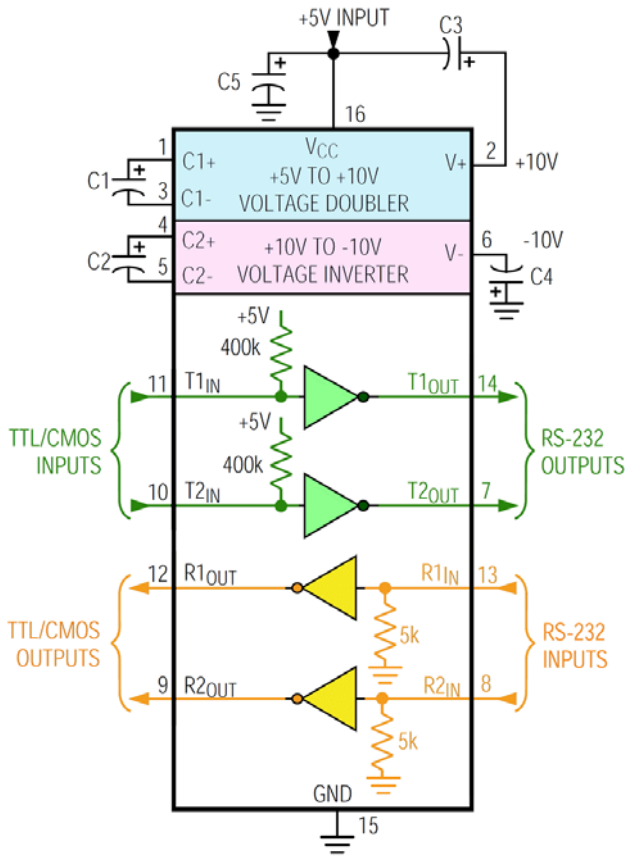
المطلوب وصل متحكم AVR من خلال النافذة التسلسلية UART مع منفذ الحاسب COM (RS232) بحيث يتم إرسال أوامر تحكم بينهما.



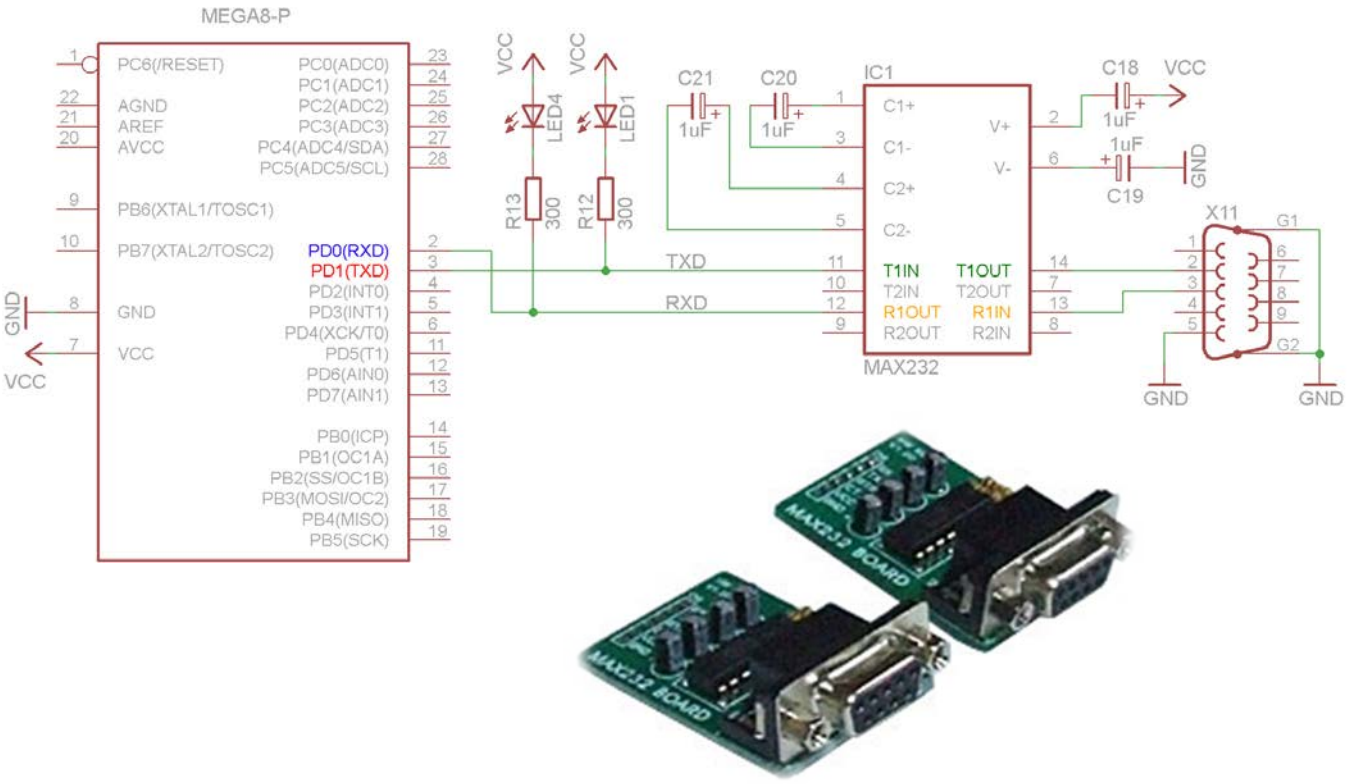
دارات الملائمة TTL <> RS232:

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.

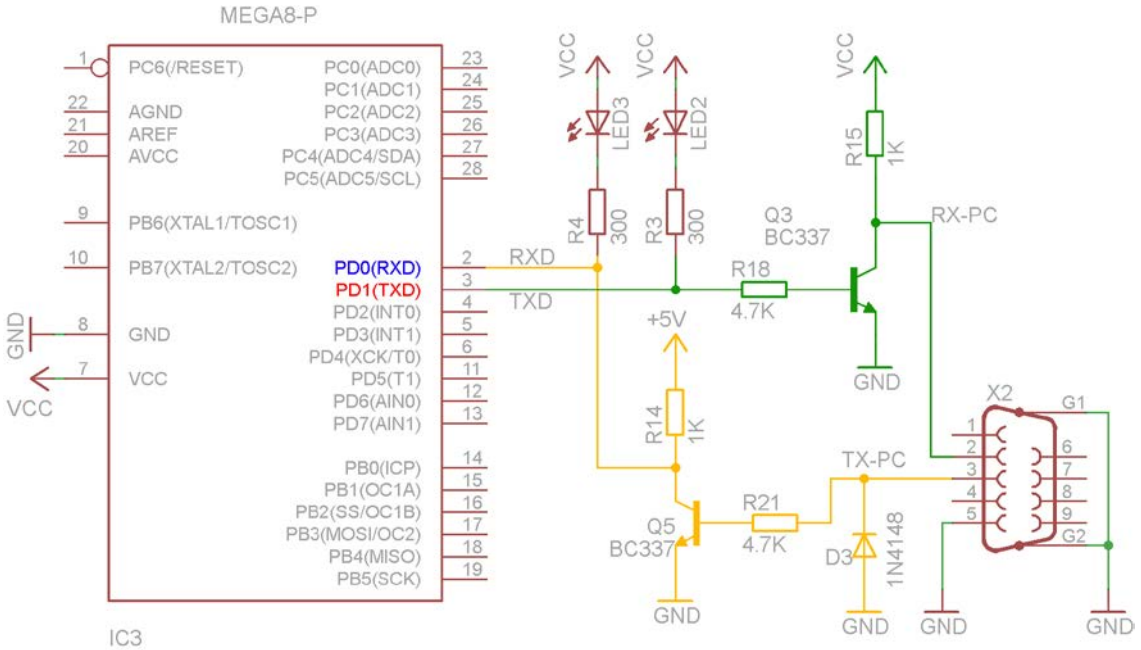
تستخدم الدارة المتكاملة Max232 كدارة تحويل وعزل TTL<>RS232.



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.

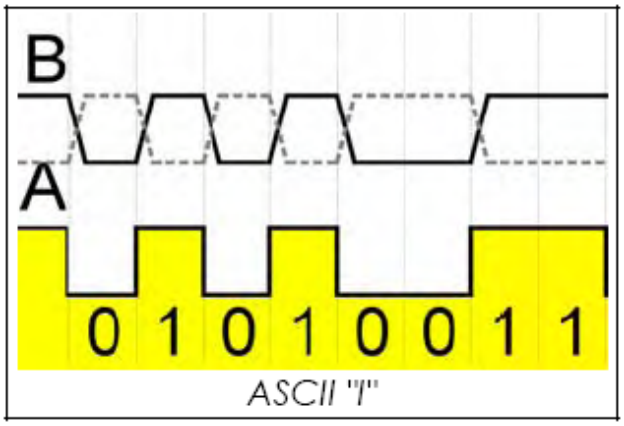


﴿ بروتوكول الاتصال التسلسلي RS485 ﴾

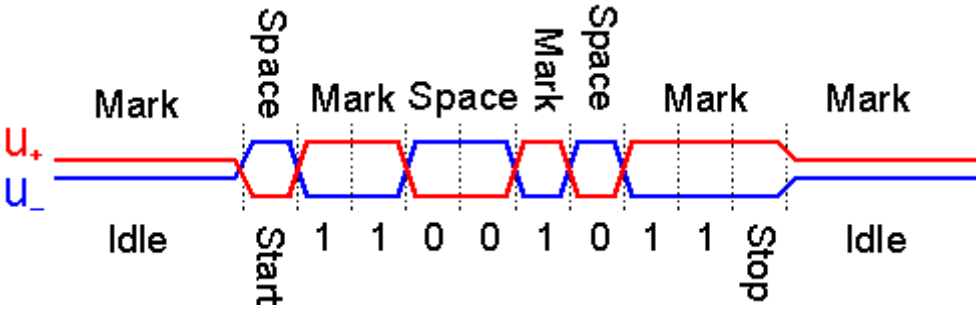
مبدأ عمل البروتوكول **RS485**:

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:

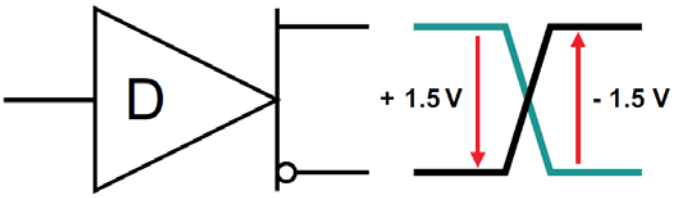
الجدول التالي يبين الشركات الأساسية المصنعة لدرايفرات البروتوكول RS485.

ST	TI	MAXIM	LTC
ST485	SN75176	MAX485	LTC485

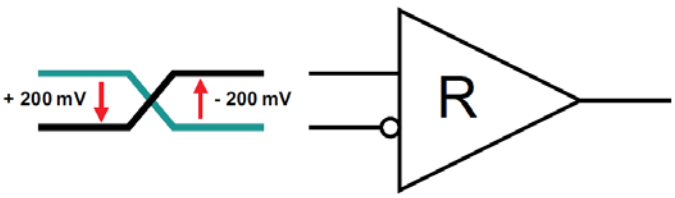
مواصفات البروتوكول RS485:

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



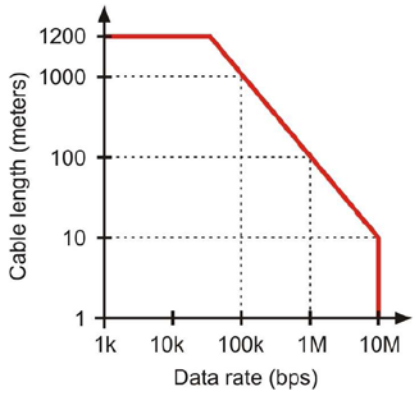
RS-485 drivers provide a differential output of a minimum 1.5V across a 54-Ω load

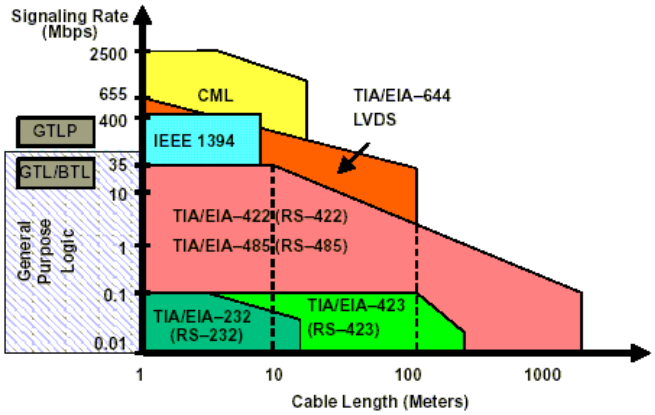


RS-485 receivers detect a differential input down to 200mV.

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.

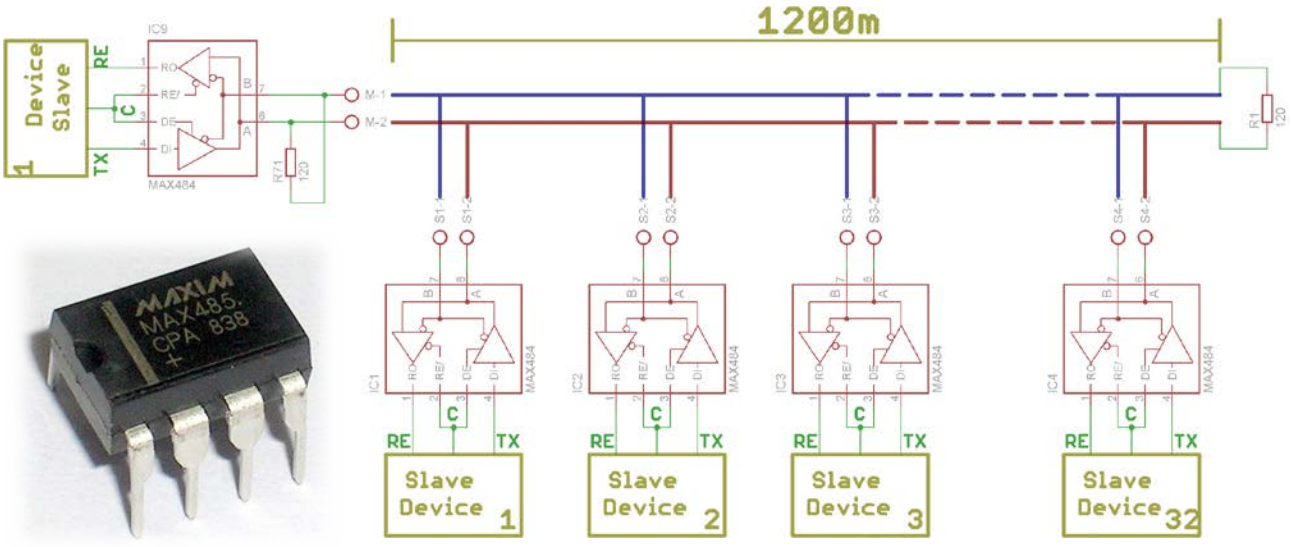




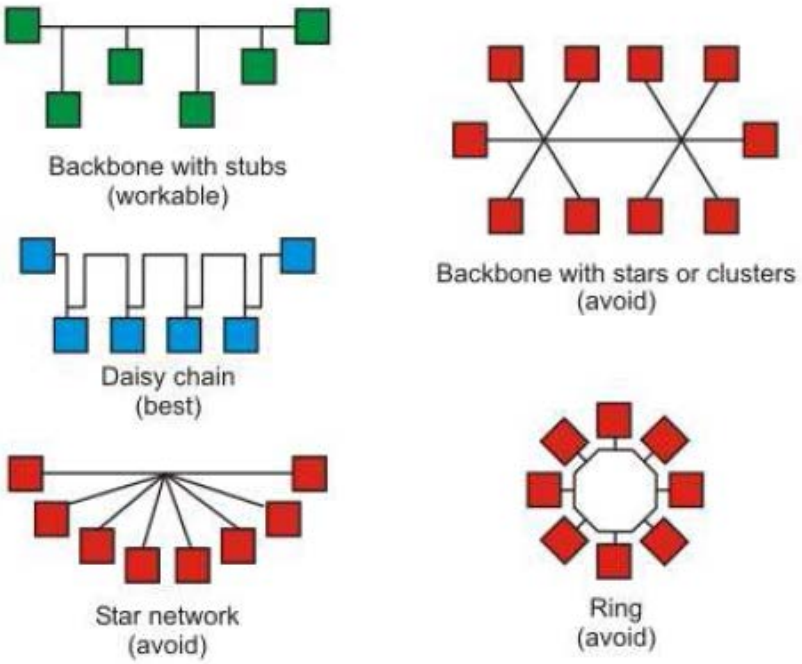
ملاحظة: بشكل عام فإنه بازدياد طول خط النقل يتم تخفيض معدل النقل لتخفيض الضجيج، ويمكن الحصول على مسافة نقل أعظمية عند معدلات نقل منخفضة.

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:

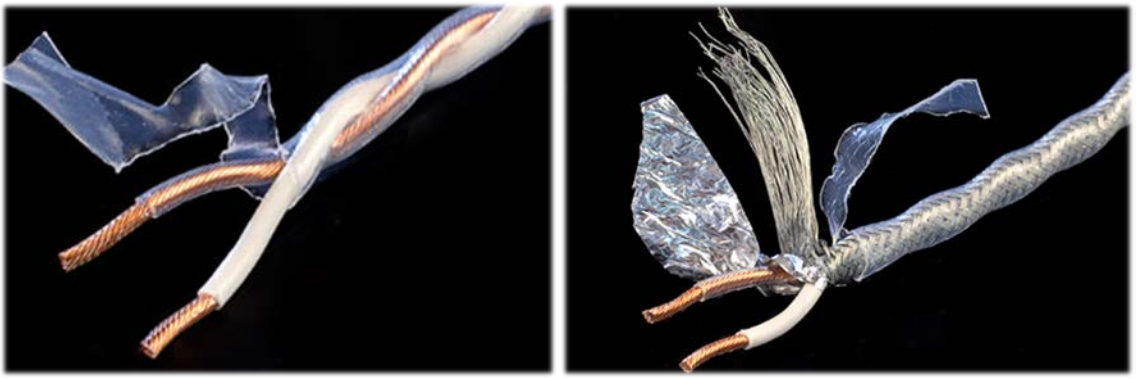


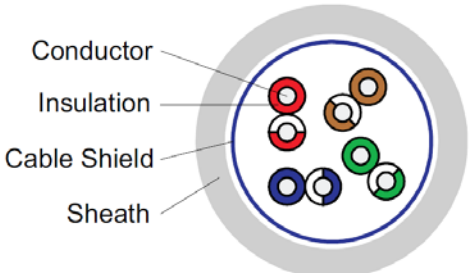
ملاحظة: إن التفريعات الصادرة عن الخط الرئيسي يجب أن لا تتجاوز 15Meter، والأفضل أن تستخدم طريقة الوصل Daisy Chain بدلاً من طريقة الوصل Backbone with Stubs.



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

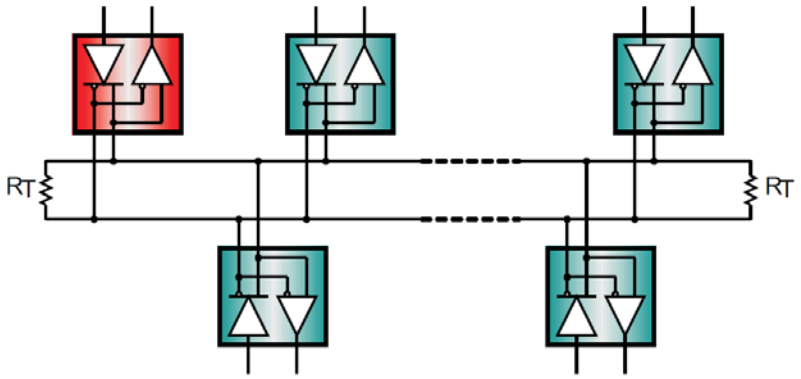




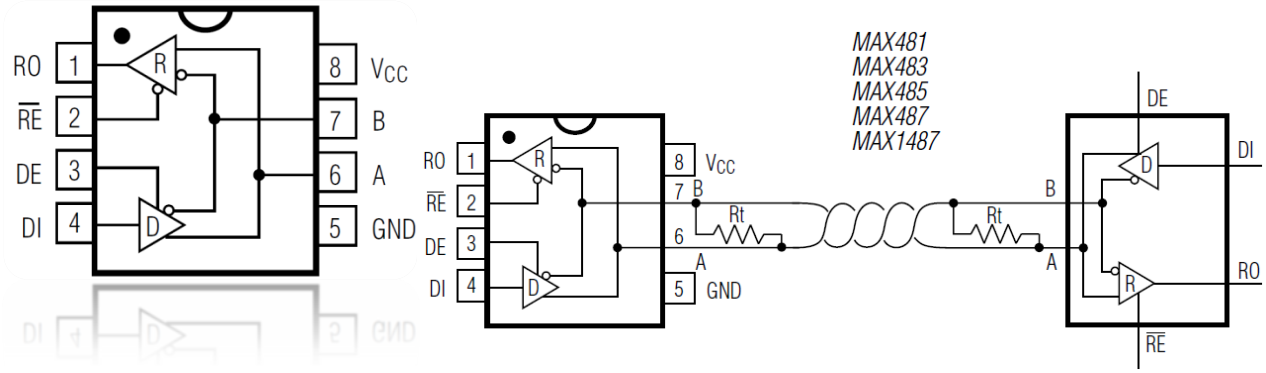
Cable : Belden 3109A
Type : 4 - pair, 22 AWG PLCT/CM
Impedance : 120 Ω
Capacitance : 11 pF/ft
Velocity : 78% (1.3 ns/ft)

توصيل شبكة RS485 أحادية الاتجاه باستخدام خطين (Half-Duplex Bus Structures in RS-485):

في هذه الحالة يتم استخدام كبل نقل بيانات بخطين فقط ويكون الناقل RS485 إما في حالة إرسال من المتحكم الرئيسي (Driver) إلى العناصر الفرعية (Receivers) على الناقل أو في حالة استقبال من العناصر الفرعية على الناقل.

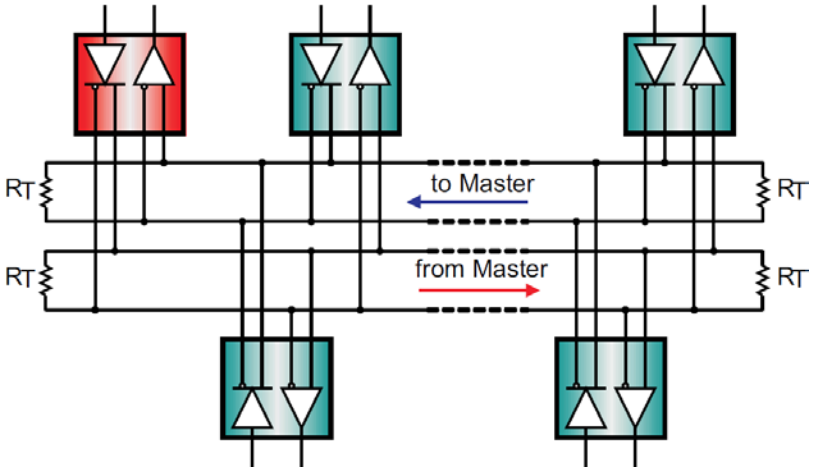


وبالتالي يمكن استخدام دارة قيادة مثل الشريحة MAX485 (تدعم حتى 32 نقطة) أو الشريحة MAX1487 (تدعم حتى 128 نقطة)

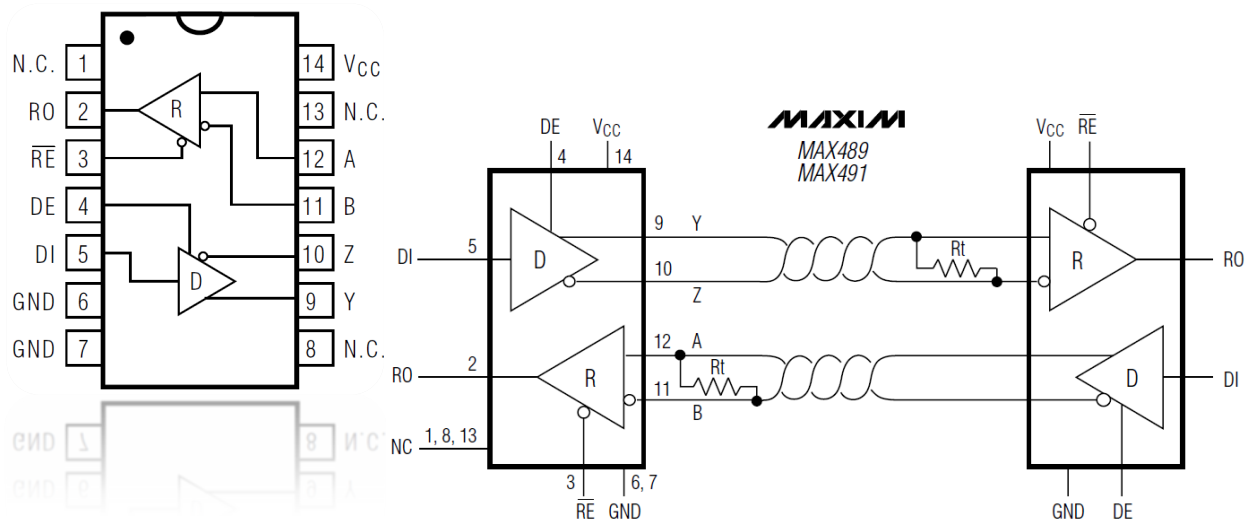


توصيل شبكة RS485 أحادية الاتجاه باستخدام خطين (Full-Duplex Bus Structures in RS-485):

في هذه الحالة يتم استخدام كبل نقل بيانات بأربع خطوط (زوجين) ويمكن ان يكون الناقل RS485 في حالة إرسال واستقبال بين المتحكم الرئيسي (Driver) والعناصر الفرعية (Receivers) على الناقل في نفس الوقت.

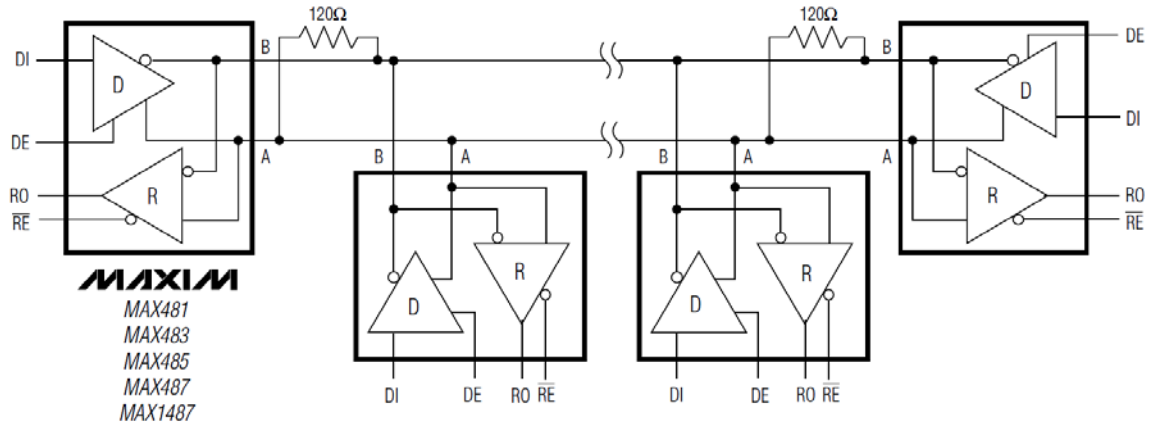


وبالتالي يمكن استخدام دارة قيادة مثل الشريحة MAX489 (تدعم حتى 32 نقطة) أو الشريحة MAX491 (تدعم حتى 128 نقطة)



مقاومة نهاية الخط في الناقل RS485 (RS-485 Bus Termination Resistor):

من المهم جداً وضع مقاومة في بداية خط ونهاية خط الناقل RS485 تساوي قيمة ممانعة الكبل (قياسياً: 120 ohm) لمنع حدوث ممانعة عالية على الناقل وبالتالي تشويش البيانات. الشكل التالي يوضع المقاومات الطرفية (على نقطتي البداية والنهاية للكبل فقط!!).



مقاومة نهاية الخط في الناقل RS485 (RS-485 Max Bus Capacitance):

عند تصميم شبكة نقل بيانات باستخدام RS485 فإن سعة التحميل على الناقل RS485 يجب أن لا تتجاوز 4000pF (سعة الكبل وسعة العناصر). بفرض أن لدينا كبل ذو سعة Cable Capacitance = 11pF / ft = 3.35pF / m والشرائح ذات قيمة سعة Driver Capacitance = 30pF، وطول الكبل 1Km، وعدد الوحدات على الخط 32 وحدة، فاحسب سعة التحميل على الناقل.

Total Cap = (3.35 x 1000) + (30 x 32) = 4310 > 4000!

وبالتالي سيحصل ضياع في البيانات والحل هو إما باستبدال الشرائح بأخرى ذات سعة أصغر (Max1487, 15pF) او استخدام كبل ذو مواصفات أفضل (CAT5, 9pF/ft).

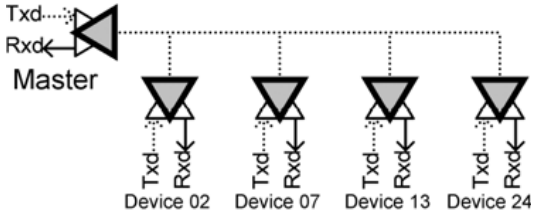
التحكم بشبكة RS485 باستخدام خطين:

يتم التحكم بهذه الشبكة وفق مبدأين:

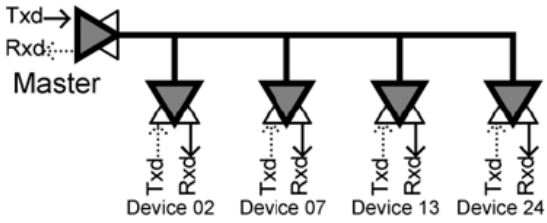
- ✓ PASSIVE DUPLEX CONTROL (*AUTOMATIC*)
- ✓ ACTIVE DUPLEX CONTROL (*RTS Pin*)

✓ **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:

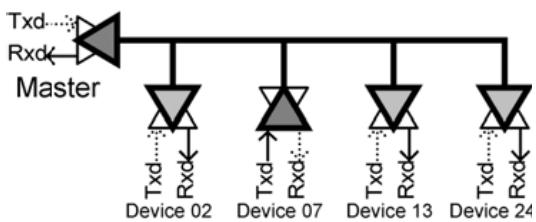
- **لا يوجد بيانات للإرسال** (No data to send): في هذه الحالة تكون الحالة المنطقية على خط النقل هي "**1**" وجميع الطرفيات الثانوية تكون في حالة انتظار استقبال بيانات من الوحدة الرئيسية للاستقبال (حالة توقف Idle).



- **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).



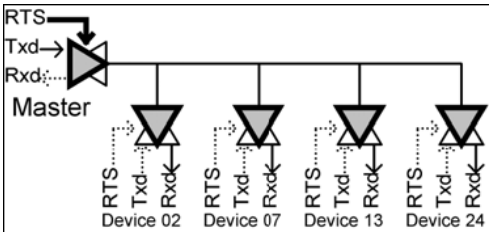
- **إرسال رد من الوحدة الثانوية** (Slave sends a response): بفرض أن عنوان الجهاز الذي تم طلبه من قبل الوحدة الرئيسية هو add = 07 وبالتالي ستقوم الوحدة الثانوية السابعة بالرد وتتحول إلى وحدة إرسال بنفس الوقت الذي تكون فيه الوحدة الرئيسية في حالة استقبال وانتظار الرد، حالما تنتهي هذه الوحدة من الإرسال تعود إلى حالتها كمستقبل ويصبح خط النقل في حالة توقف من جديد.



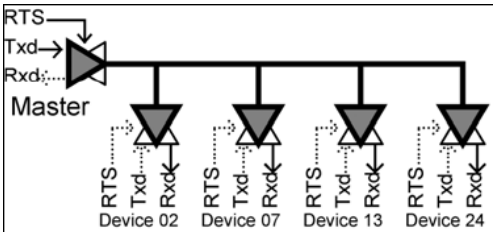
✓ **ثانياً: نمط التحكم Active Duplex**: في هذه الحالة يستخدم ثلاث خطوط للتحكم بشريحة RS485 Driver، حيث يستخدم الخط الثالث (RTS Control) كخط تحكم لتحديد اتجاه وحدة الإرسال/الاستقبال.

مبدأ العمل:

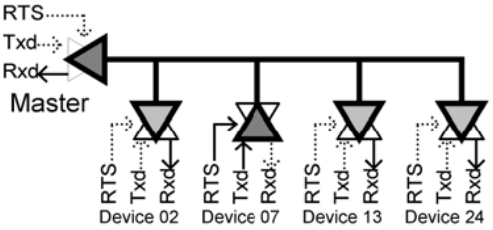
- **لا يوجد بيانات للإرسال** (No data to send): في هذه الحالة تقوم الوحدة الرئيسية بتفعيل إشارة خط التحكم بالاتجاه الخاص بها (RTS) مما يجبر خط النقل إلى التحول إلى حالة التوقف (الحالة المنطقية على خط النقل هي "1"). كل إشارات التحكم بالاتجاه الخاصة الوحدات الثانوية تكون غير مفعلة والوحدات في حالة ترقب.



- **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.



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):

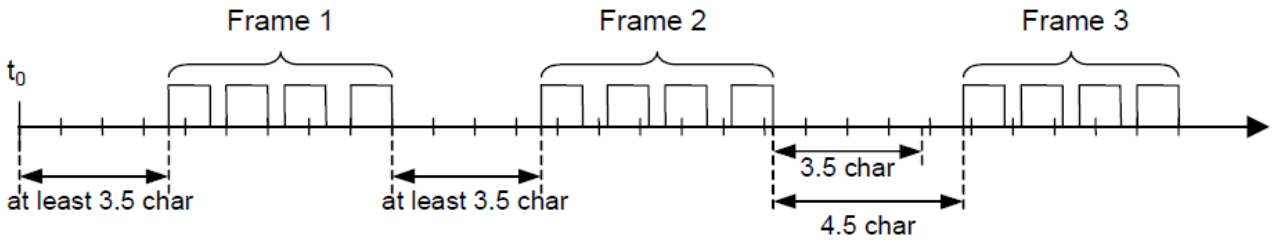
Slave Address	*Function Code*	*Byte Count*	*Start Address*	Data Stream	*CRC* *CRC Low* *CRC Hi*
1 byte	*1 byte*	*1 byte*	*2 byte*	*0 ~ 252 byte*	*1~2 byte*

- **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.

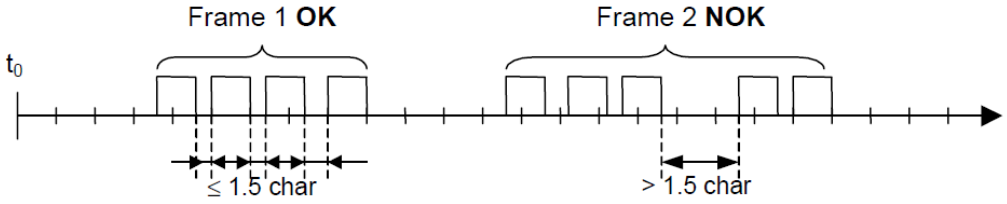
في حال أريد إرسال حزمتي بيانات متلاحقة، فإنه يجب أن يفصل بينهما بزمن يسمى Inter-frame-time وهو يساوي $t_{3.5}$ بناءً على الشكل التالي:

MODBUS message

Start	Address	Function	Data	CRC Check	End
≥ 3.5 char	8 bits	8 bits	N x 8 bits	16 bits	≥ 3.5 char



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:

$$\frac{560}{9600} \times 1000 = \mathbf{58.3\ ms} \qquad 70_{Byte} \times 8_{Bit} = 560_{Bit}$$

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.

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.

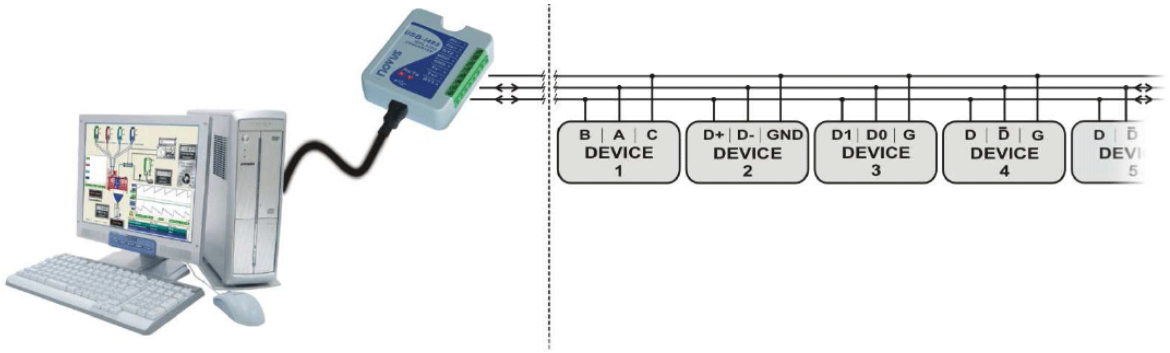
Slave Address	*Function Code*	*Byte Count*	*CRC* *CRC Low* *CRC Hi*
1 byte	*1 byte*	*1 byte*	*1~2 byte*

مثلاً: في رسالة الرد أعلاه يقوم المقاد بإرسال عنوانه (Slave) والوظيفة التي استلمها وعدد البايتات التي تلقاها ويحسب قيمة الـ CRC.

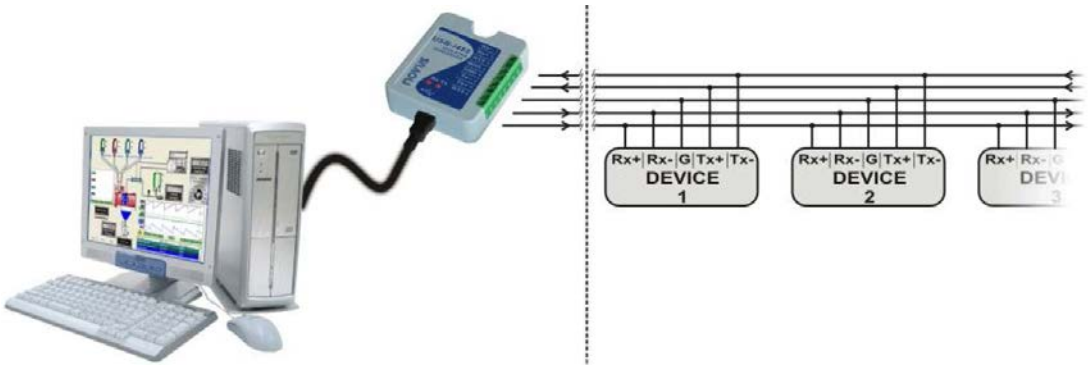
نمط الإرسال أحادي | ثنائي الاتجاه للبروتوكول RS485:

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.



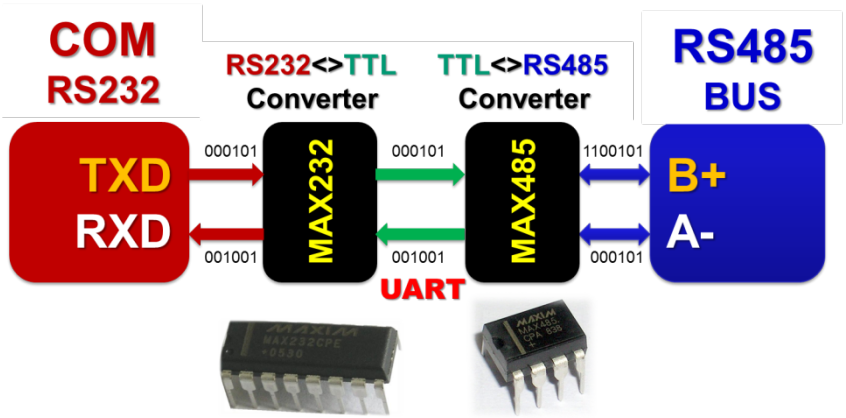
الشكل التالي توصيل شبكة ثنائية الاتجاه.

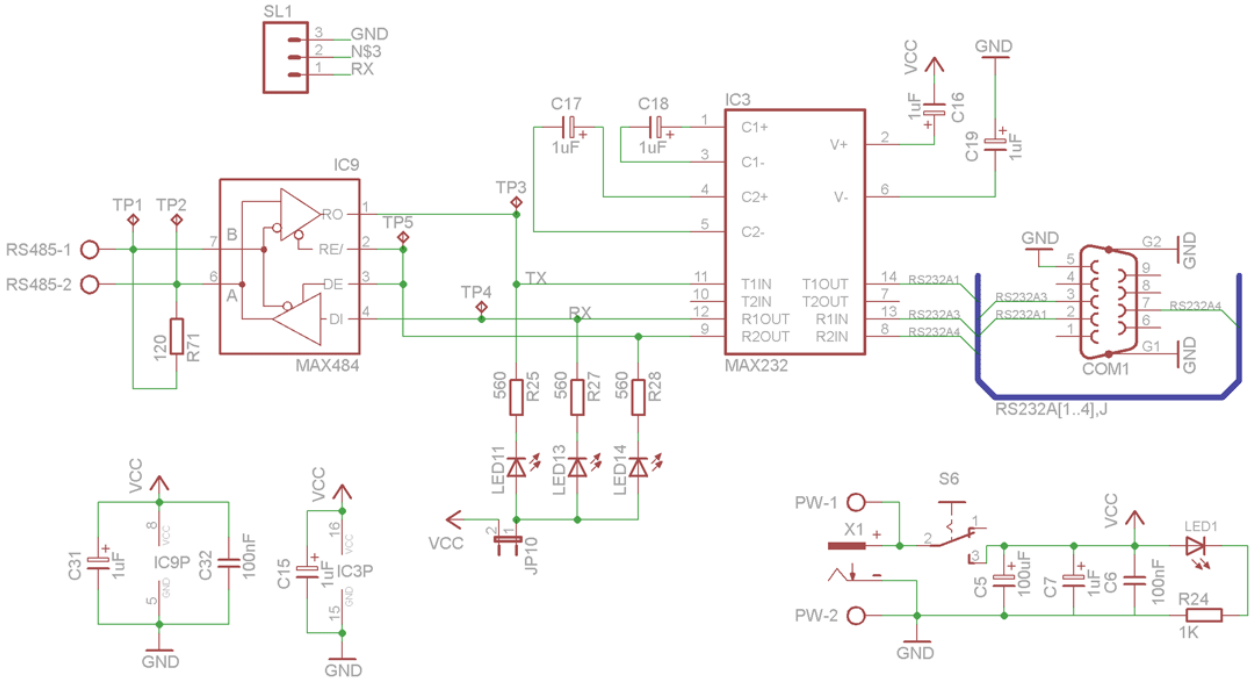


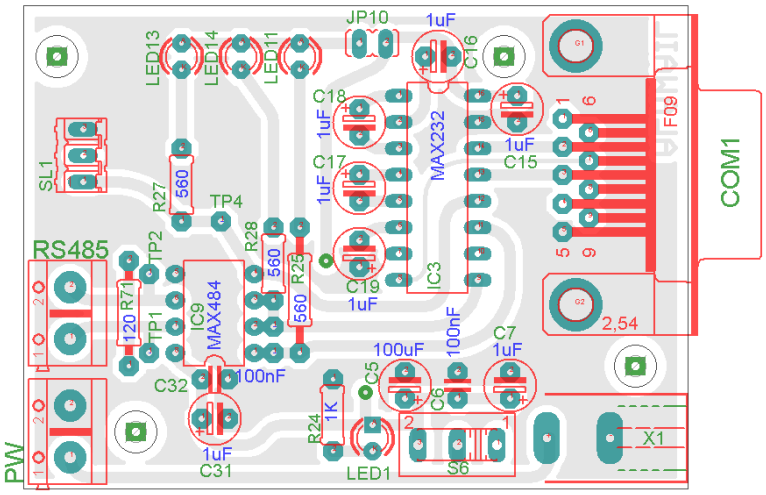
إن الشبكات RS485 أحادية الاتجاه أكثر شيوعاً، ولكن في بعض التطبيقات التي تتطلب تحكماً ومراقبةً للنظام في الزمن الحقيقي تستخدم الشبكات ثنائية الاتجاه.

مخطط دارة الاستقبال والإرسال عن طريق الحاسب (RS232<>RS458):

الشكل التالي المخطط النظري لدارة تحويل RS458<>RS232.

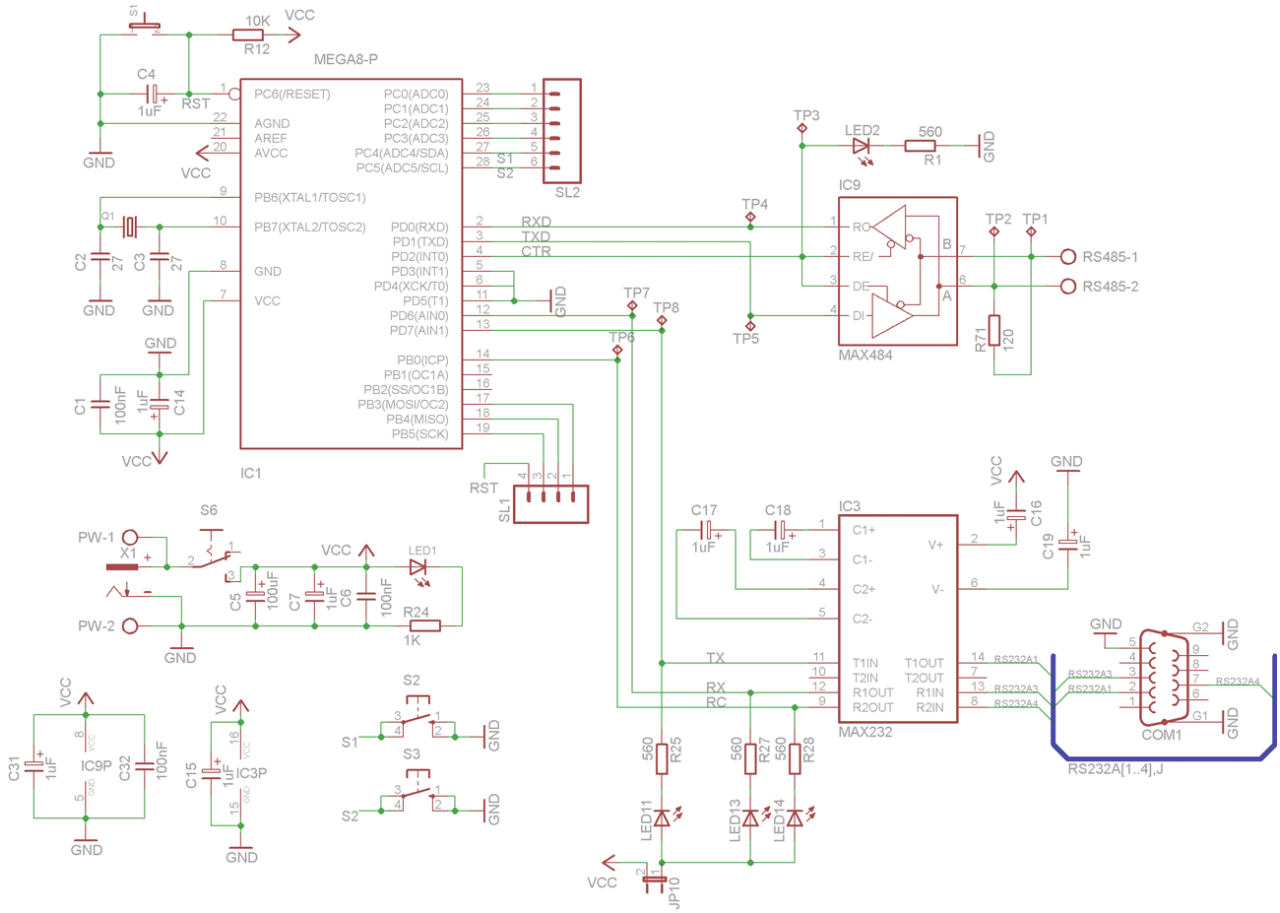




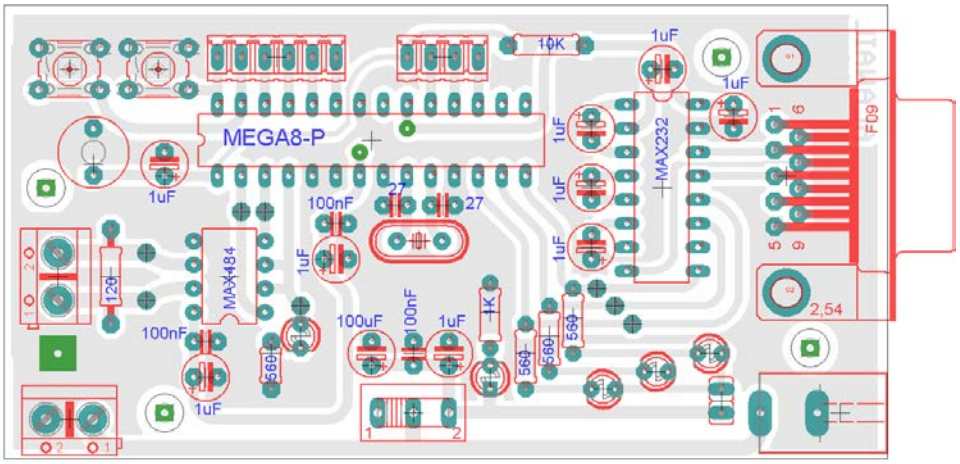


مخطط لوحة اختبار لدارة استقبال وإرسال عن طريق الحاسب ومتحكم (RS232<>UART<>RS458):

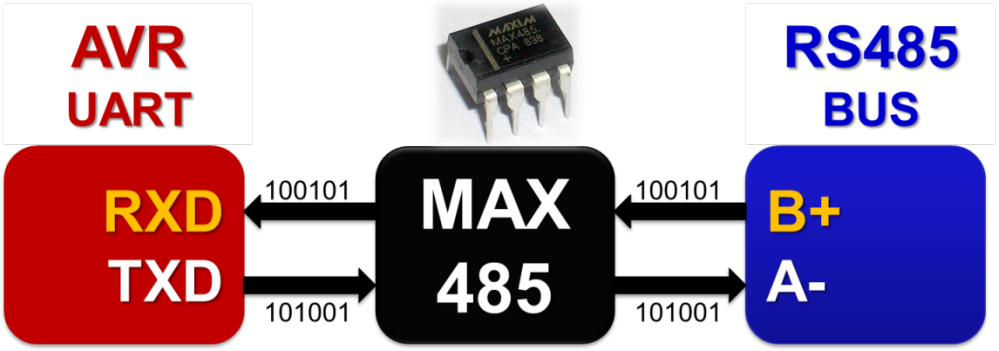
الشكل التالي المخطط النظري لدارة تحويل RS232<>UART<>RS458. في هذه الدارة يتعامل المتحكم مع الشريحة Max485 الموصولة مع النافذة UART الأساسية، ويتعامل بنفس الوقت مع الشريحة Max232 الموصولة مع نافذة UART برمجية، وبالتالي يمكن للمتحكم أن يكون وسيطاً بين البروتوكولين.

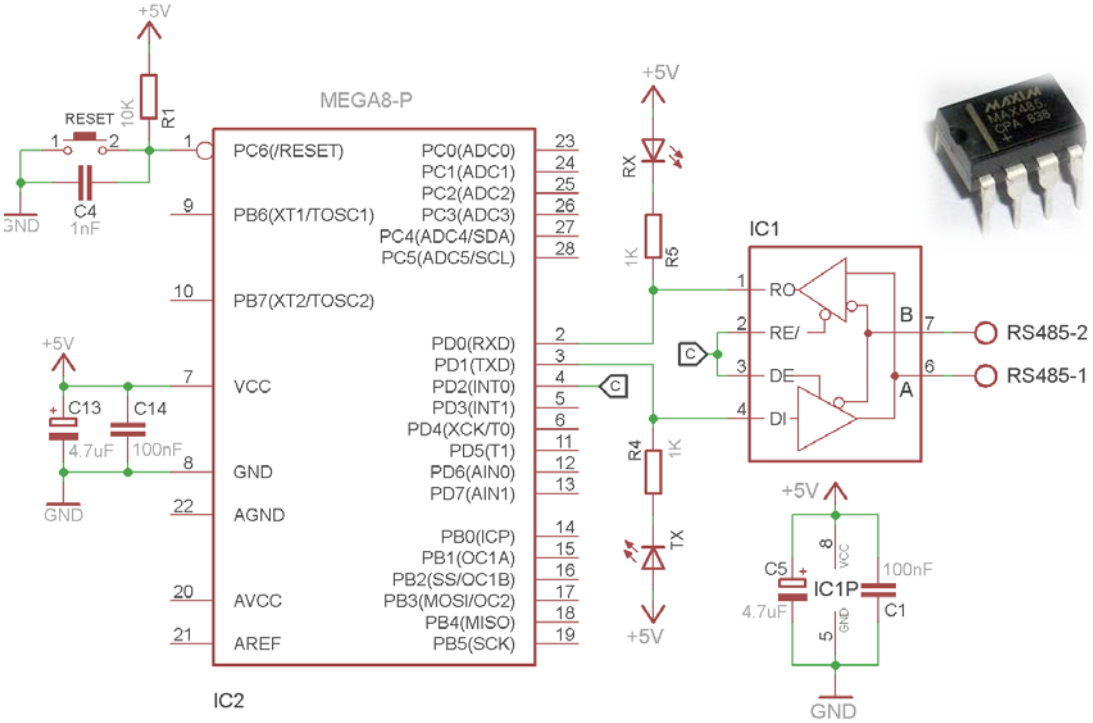


الشكل التالي مخطط الدارة المطبوعة وتوزع العناصر لدارة التحويل RS232<>UART<>RS458.



مخطط لوحة اختبار لدارة استقبال وإرسال بين متحكم وخط RS485 (RS485 Adapter <> UART)





﴿ بروتوكول الاتصال التسلسلي USB ﴾

يوجد العديد من تقنيات الربط مع الحاسب مثل منفذ الطابعة LPT والمنفذ التسلسلي COM ومنفذ USB ومنفذ Firewire إلا أن معظم الأجهزة الطرفية الحديثة أصبحت تستخدم منفذ USB كطريقة ربط مع الحاسب لما له من مميزات على الأنواع الأخرى.

مقارنة بين بروتوكولات الاتصال التسلسلي الشائعة الاستخدام:

البروتوكول	عدد الأجهزة على الممر	طول قناة الاتصال (قدم)	السرعة الأعظمية bits/sec	صيغة الاتصال	الاستخدامات
USB	127	16	1.5M, 12M, 480M	تسلسلي غير متواقت	لوحة المفاتيح، الفأرة، المودم
RS-232	2	50-100	20k -115k	تسلسلي غير متواقت	الفأرة، المودم تجهيزات أخرى
RS-485	-	4000	10M	تسلسلي غير متواقت	الشبكات الصناعية
IrDA	2	6	115k	تسلسلي غير متواقت	الطابعات، الأجهزة الكفية
Microwire	8	10	2M	تسلسلي متواقت	المتحكمات المصغرة
SPI	8	10	2.1M	تسلسلي متواقت	المتحكمات المصغرة
I^2C	40	18	3.4M	تسلسلي متواقت	المتحكمات المصغرة
FireWire	64	15	400M-3.2G	تسلسلي	الفيديو، وحدات التخزين
IEEE-488	15	60	8M	تفرعي	الشبكات الصناعية
Ethernet	1024	1600	10M/100M/1G	تسلسلي	شبكات الحاسب
MIDI	2	50	31.5k	تسلسلي	أجهزة موسيقية
LPT	2	10-30	8M	تفرعي	طابعات، ماسحات ضوئية

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:

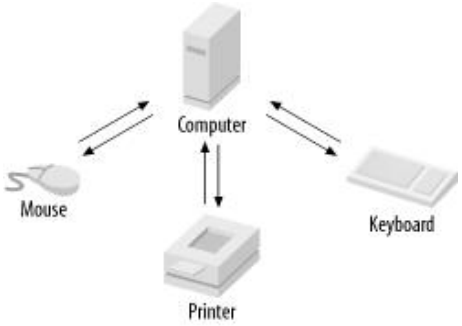
يوجد هناك عدة معايير للبروتوكول USB: USB 3.0 & USB 2.0 & USB 1.1 & USB 1.0 تدعم أربع سرعات:

- 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:

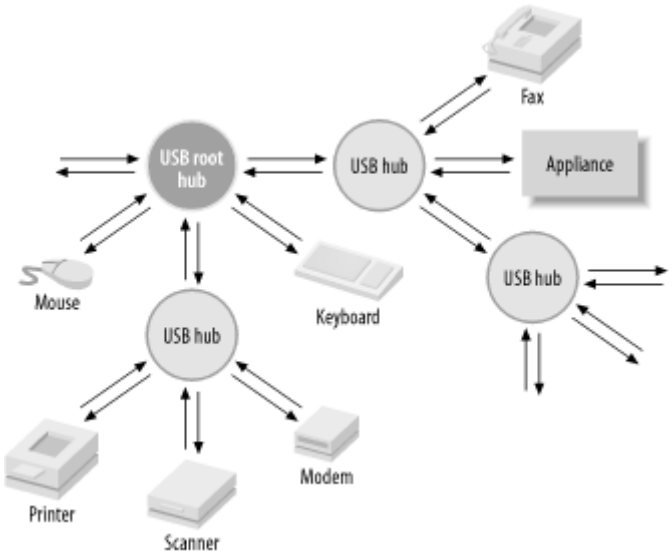
يتصف ممر USB بأنه "Host Controlled" أي يوجد هناك جهاز مضيف واحد على الممر (وهو عبارة عن جهاز يتحكم بالممر وببقية الأجهزة الموصولة معه أثناء العمل) يقع على عاتقه عدة مسؤوليات مثل تهيئة عمليات النقل وتوزيع عرض الحزمة، ولا تسمح تقنية USB بوجود أكثر من جهاز مضيف واحد على الممر، إلا أنه صدر معيار جديد يدعى"On-The-Go" 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.

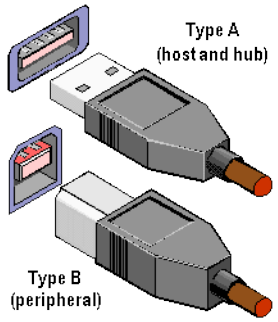
لوصل أكثر من جهاز إلى منفذ USB واحد نقوم بوصل تلك الأجهزة إلى موزع HUB والذي بدوره يوصل إلى هذا المنفذ، ويتوفر حالياً العديد من الطرفيات التي تحوي على موزعات مدمجة فيها مثل الشاشات و لوحات المفاتيح.



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.



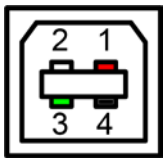
المواصفات الميكانيكية والكهربائية لمنفذ USB:

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.

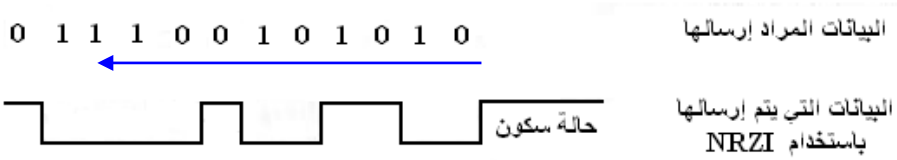
يستخدم ممر USB أربع خطوط: خطي تغذية **GND** & **+5v**، خطي معطيات **D+** & **D–**.

1 **VCC** (أحمر) 2 **D–** (أبيض) 3 **D+** (أخضر) 4 **GND** (أسود)		

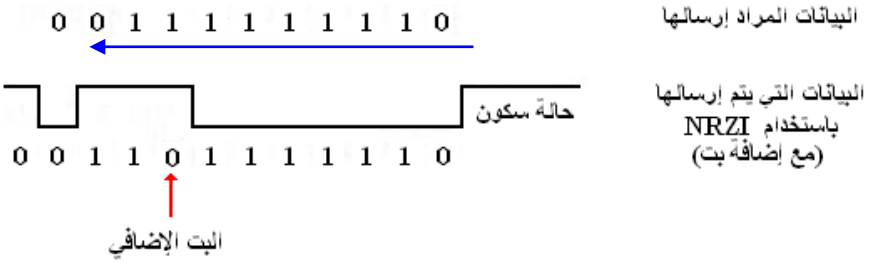
يستخدم ممر USB الخطين D+&D- لإرسال البيانات بشكل تفاضلي، فمثلاً لإرسال '1' منطقي بشكل تفاضلي على الممر يجب وضع '1' منطقي على القطب D+، '0' منطقي على القطب D- وذلك في حال السرعة الكاملة والمنخفضة للممر، وينعكس هذا التشفير في حال السرعة العالية.

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.



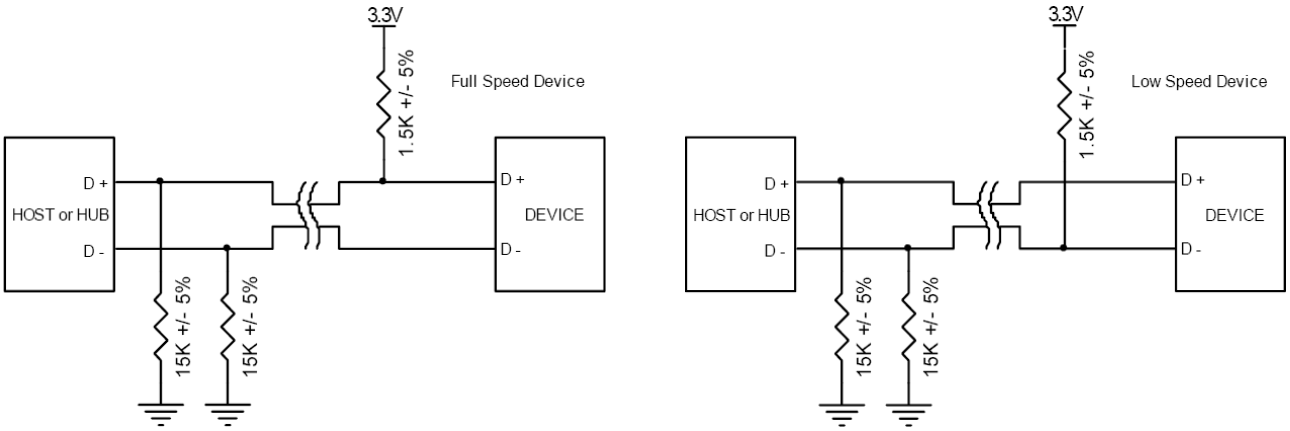
عندما يراد إرسال أكثر من ستة خانات بقيمة '1' منطقي فإنه يتم الفصل بين كل ستة من هذه الخانات بخانة إضافية تحمل القيمة صفر، و ذلك لضمان استمرار عملية التزامن بين المرسل و المستقبل.



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.

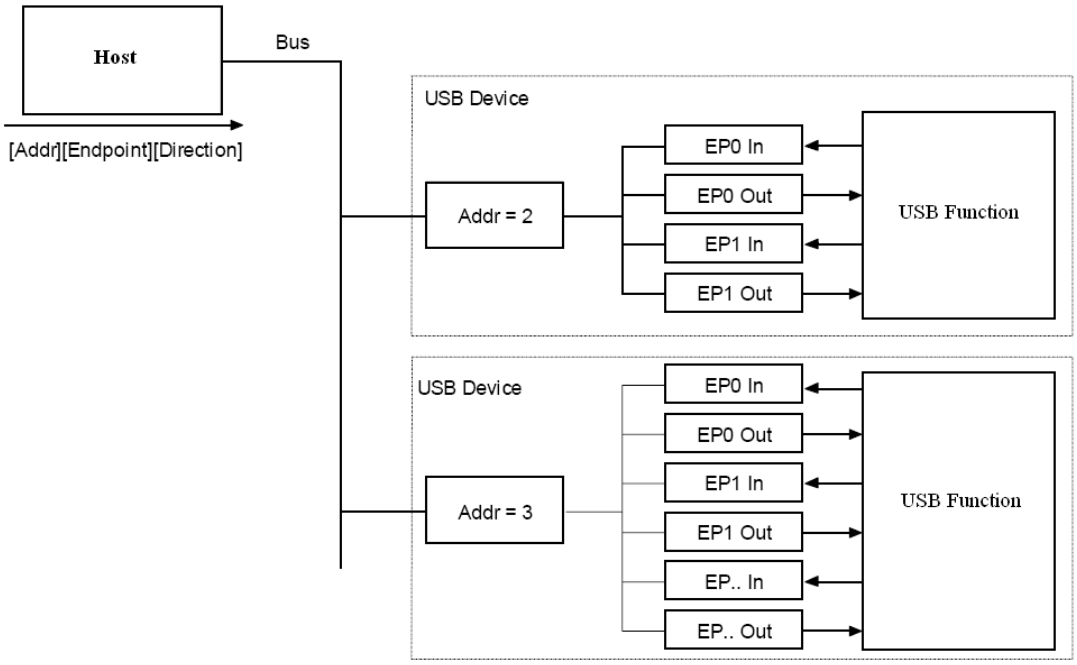
جهاز USB (USB Device): هو تعبير عام قد يشير إلى وحدة طرفية أو حاسب مضيف أو موزع أو إلى دارة متحكم USB في المضيف (Host Controller IC) ولتجنب هذه العمومية تم استخدام مصطلح (USB Function) حيث يقوم المضيف بتخصيص عنوان فريد لكل جهاز USB أثناء عملية التهيئة.

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 وتقسم 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).

Setup Token : تستخدم هذه الرزمة للدلالة على بداية عملية تهيئة Control Transfer وهي تتألف من نفس حقول رزمة In.

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.

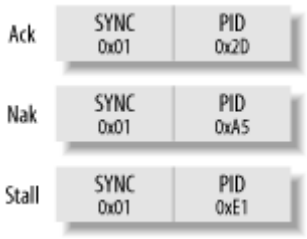
Data 0	SYNC 0x01	PID 0x3C	Data 0-1023 bits	CRC 16 bits
Data 1	SYNC 0x01	PID 0xB4	Data 0-1023 bits	CRC 16 bits

رزمة المصافحة (Handshake Packet): ولها ثلاثة أنواع:

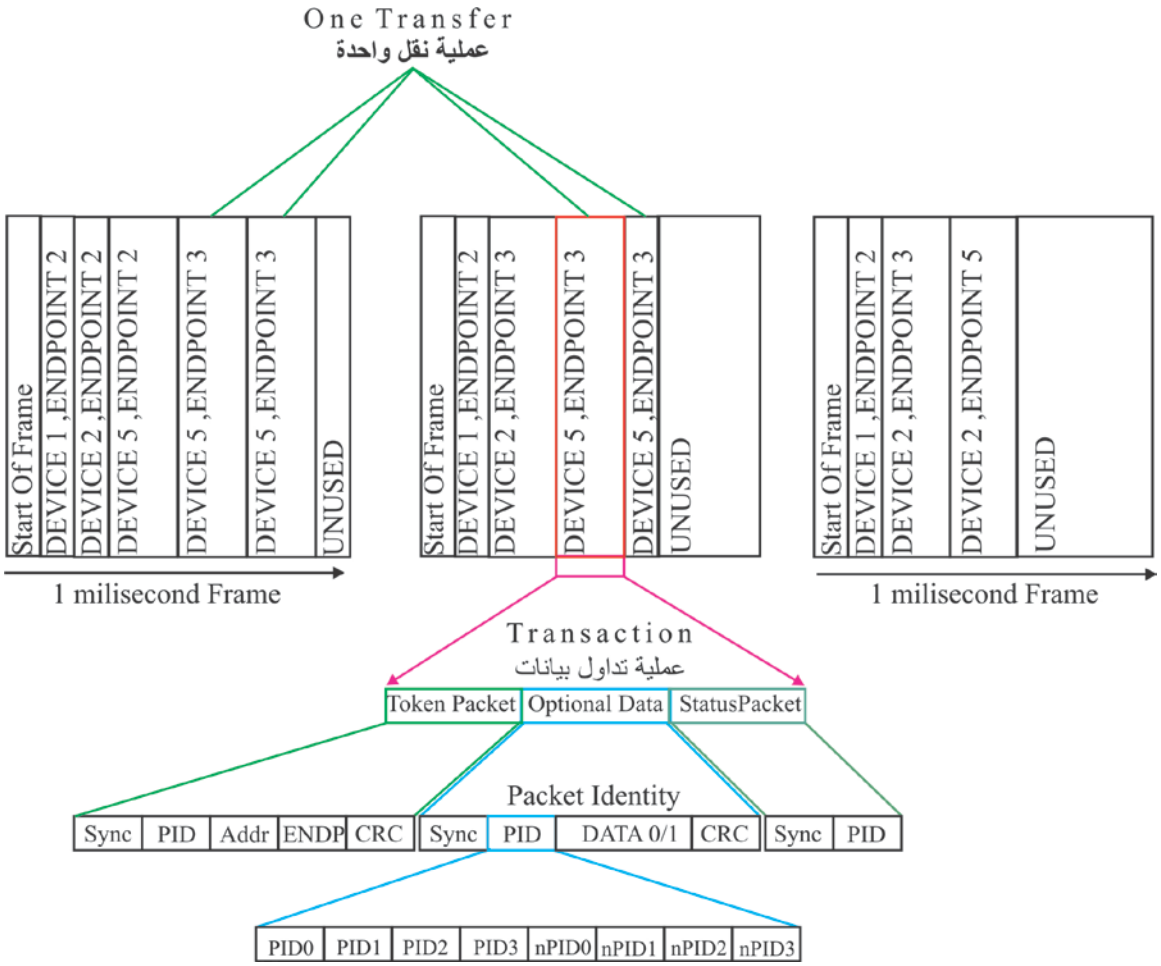
Acknowledge: وتدل على أن عملية التداول قد تمت بنجاح وأنه تم استقبال البيانات بشكل صحيح.

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.



مخطط النقل للبروتوكول USB



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.

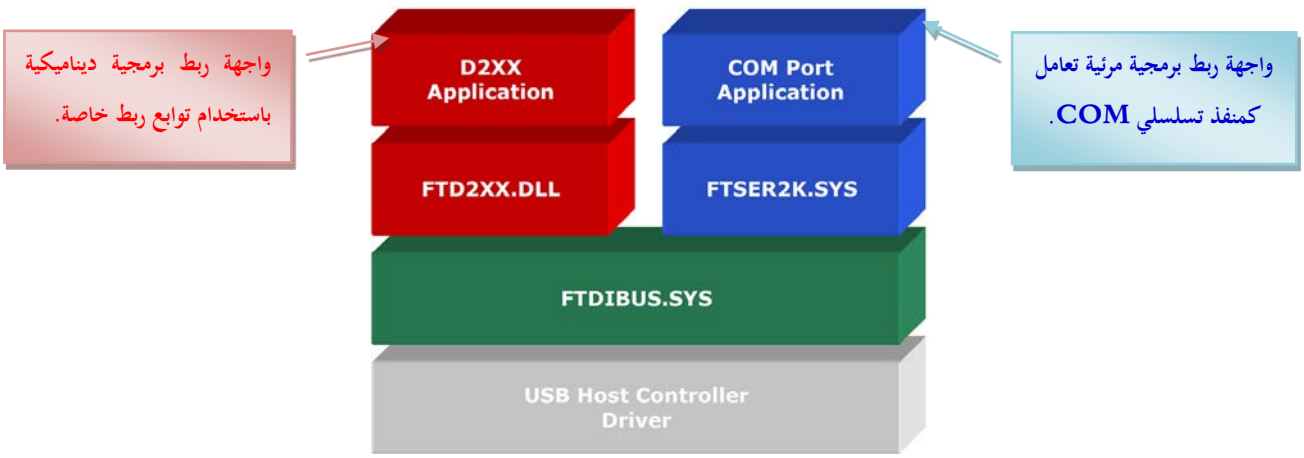
لذلك وبسبب الطلب المتزايد على هذه التقنية واقتحامها للسوق العالمية فإن هنالك الكثير من الشركات التي وفرت على المصممين عناء تصميم العتاد الالكتروني لينصّب اهتمامهم على كتابة برامج القيادة، لذلك كل ما يتوجب على المصمم هو الإطلاع على معايير USB بغرض فهم كيفية التعامل مع هذا العتاد الالكتروني.

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:



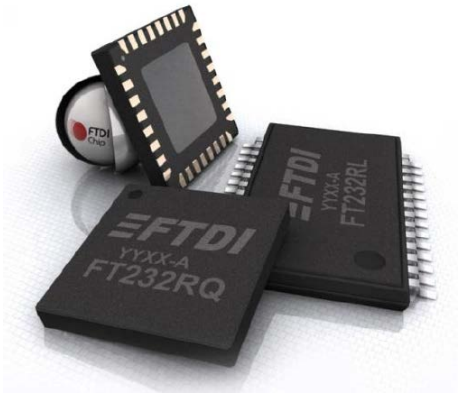
فيما يلي جدول مقارنة بين واجتهي الربط (D2XX driver & VCP driver):

	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.

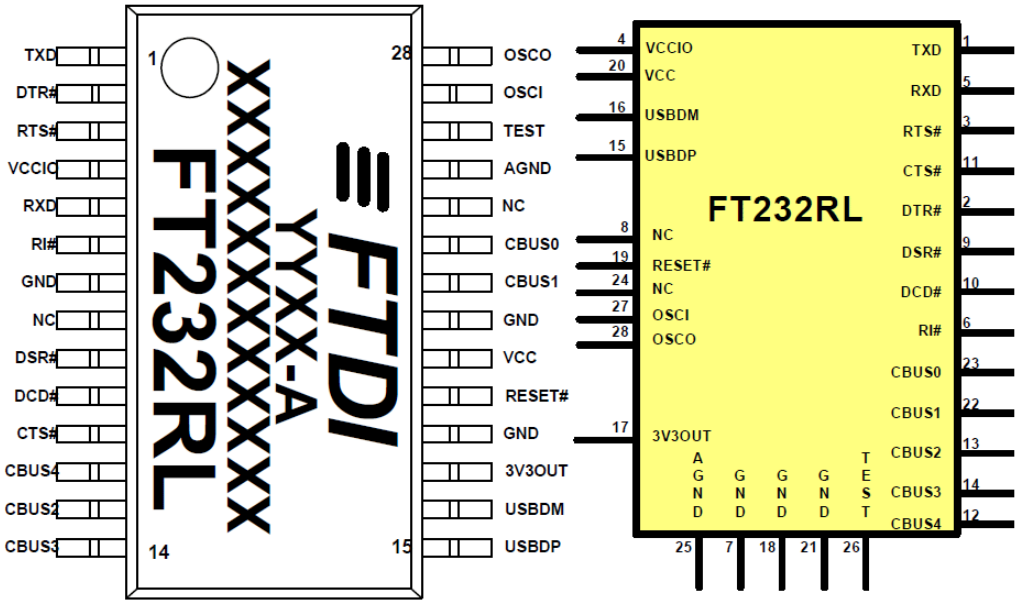
الشريحة FT232BM:

- ✓ 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~3Mb بحسب نوع برنامج القيادة.
- ✓ ذاكرة استقبال وسيطية من نوع FIFO بطول 256 بايت.
- ✓ ذاكرة إرسال وسيطية من نوع FIFO بطول 128 بايت.
- ✓ رقمي VID, PID ورقم تسلسلي للمنتج ووصف لهذا الجهاز.
- ✓ توفر العديد من المقالات التقنية من الشركة المصنعة تقدم معلومات مفصلة عن طرق استخدام هذه الشريحة.

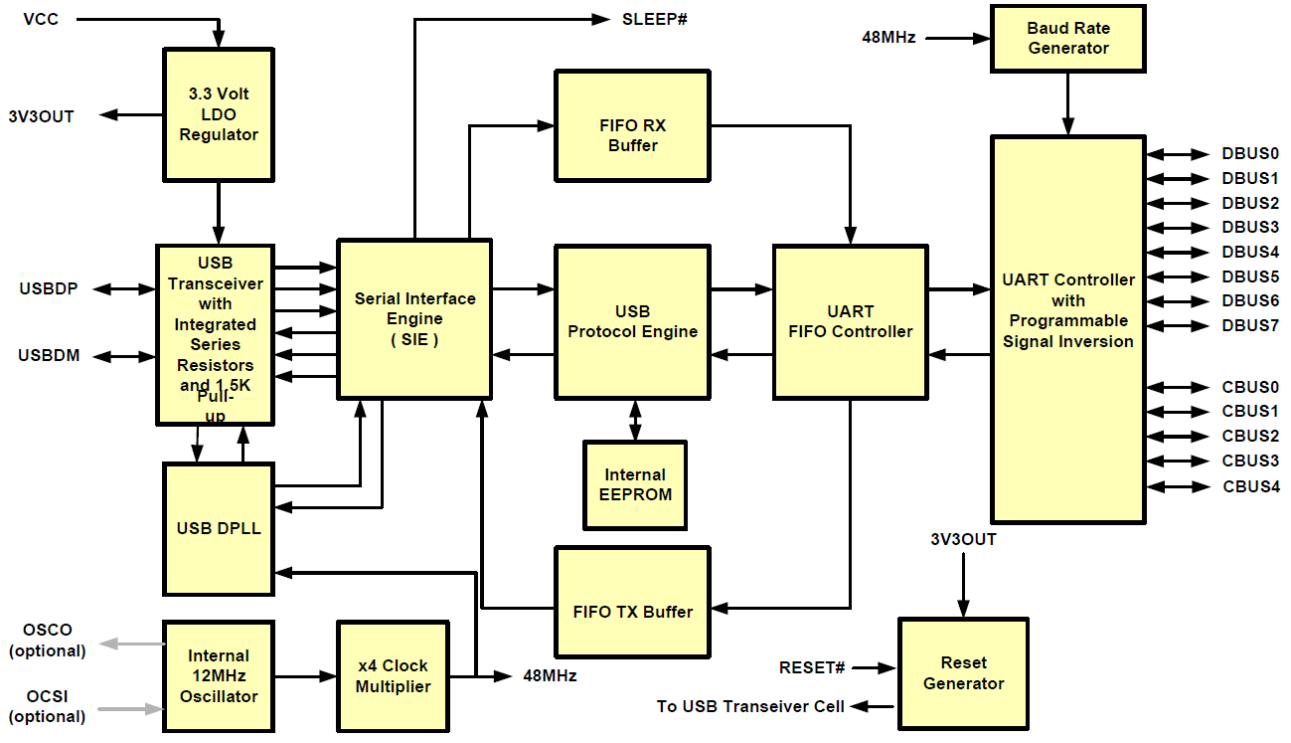


تلعب هذه الشريحة دور الملائم بين منفذ USB وبين النظام حيث تقوم باستقبال بيانات منفذ USB وتستخلص منها البيانات المطلوبة، كما تقوم بإرسال البيانات من المتحكم بشكلها التسلسلي إلى منفذ USB بعد إضافة الحقول اللازمة لتحقيق بروتوكول USB.

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.

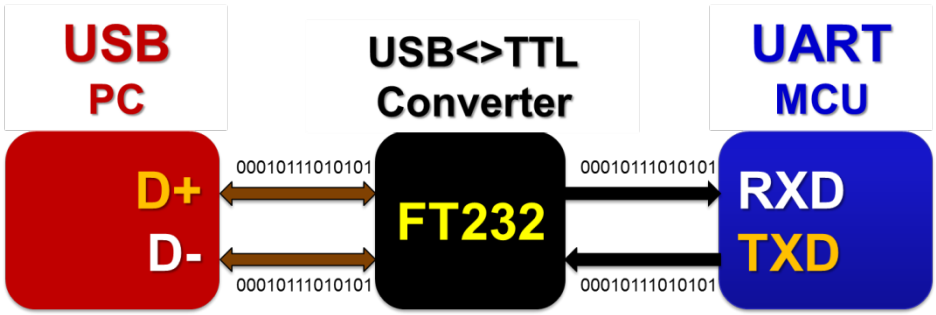
تمتلك هذه الشريحة قطب Wake up يسمح للنظام بإخراج الحاسب من الوضع الاحتياطي عند ورود جبهة صاعدة وذلك في حال كان النظام في حالة وضع احتياطي، أما إذا لم يكن كذلك فإن هذه الجبهة الصاعدة تؤدي إلى إرسال البيانات الموجودة في ذاكرة الاستقبال إلى الحاسب.

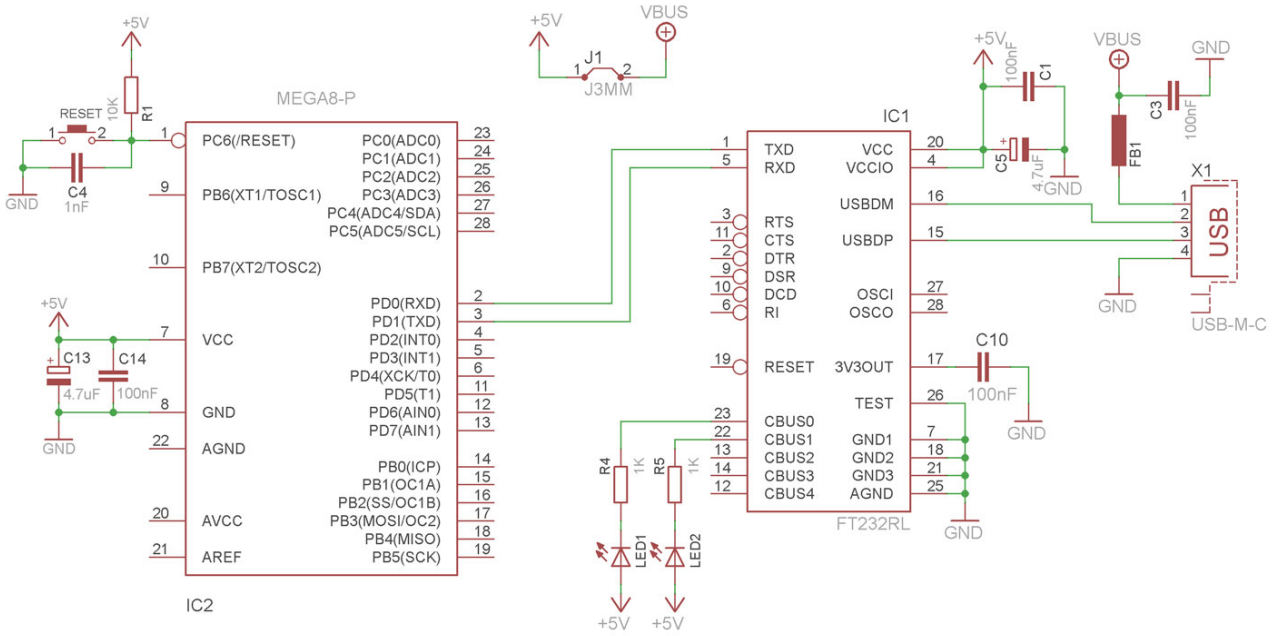
يتم تحقيق المتطلبات الخاصة لبرتوكول التخاطب مع منفذ USB في الشريحة من قبل وحدة الملائمة التسلسلية SIE (Serial Interface Engine) التي تقوم بالمهام التالية:

- 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).

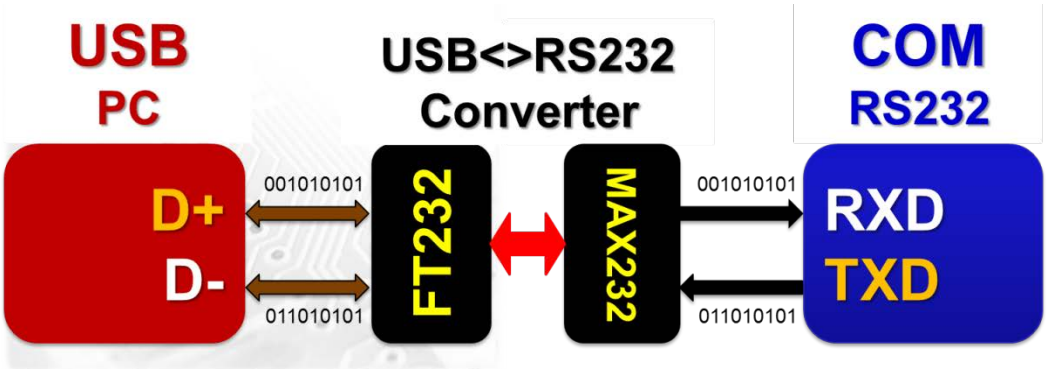
تقوم وحدة (USB Transceiver) بملائمة الشريحة مع الممر وفقاً للشروط الكهربائية المحققة للمعايير USB1.1 ,USB2.0، بما في ذلك تعريف سرعة الشريحة على الممر باستخدام مقاومة شد على القطب +D لأن سرعة الشريحة تصنف كسرعة كاملة.

ربط متحكم AVR من خلال النافذة UART (TTL) مع منفذ USB (Differential) .

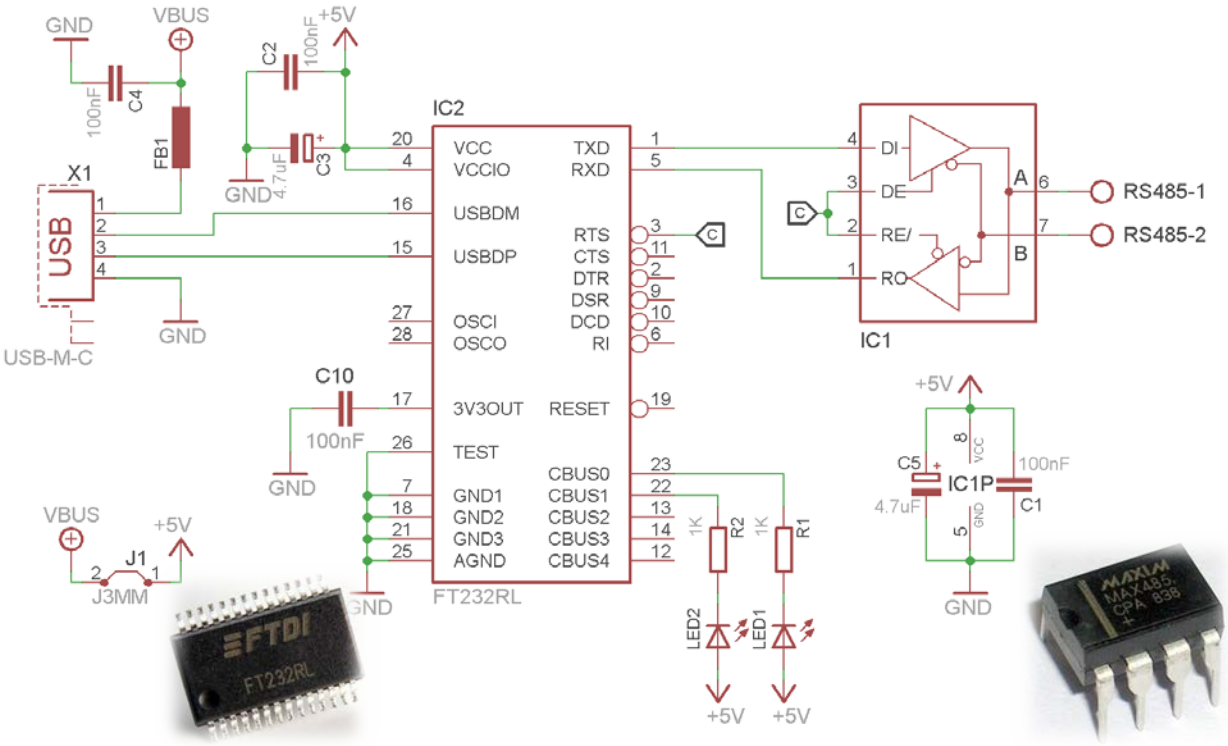




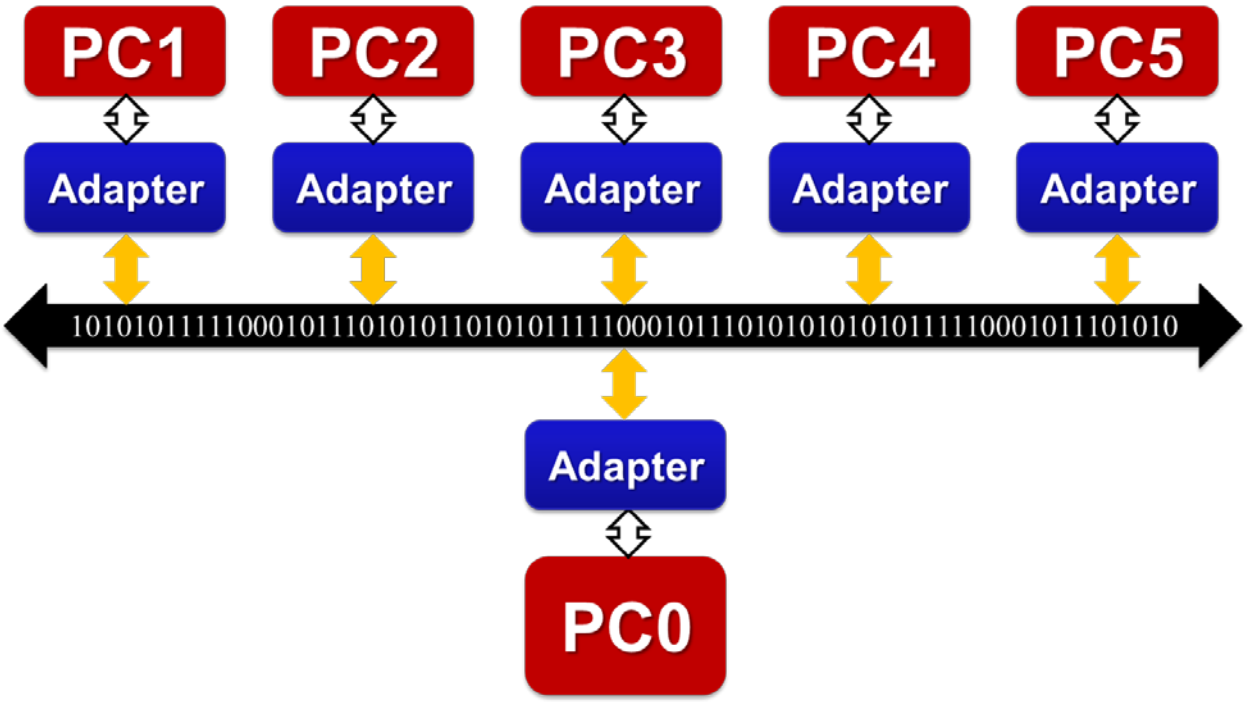
ربط منفذ COM (RS232) مع منفذ USB (Differential) . . .



ربط منفذ USB مع ناقل RS485 (RS485 Adapter <> USB)



مخطط ربط مجموعة حواسب مع ناقل RS485



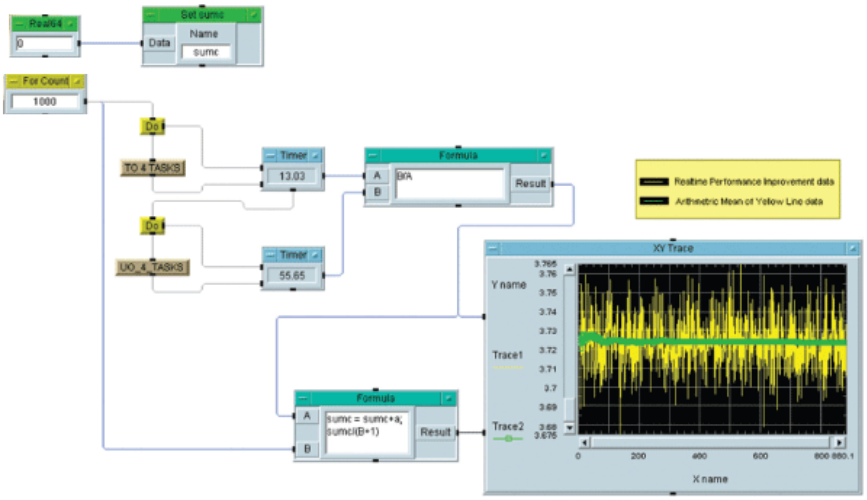
﴿مدخل إلى البرمجة الرسومية في البيئة LabVIEW﴾

لغات برمجة الكيان الصلب الرسومية (Graphical Hardware Programming Language):

على مدى سنوات عديدة تطورت لغات البرمجة الرسومية واتسعت دائرة التطبيقات التي تشملها لتغطي التطبيقات البرمجية الحاسوبية وتطبيقات الأنظمة المدمجة وغيرها. حالياً يوجد العديد من لغات البرمجة الرسومية؛ من أشهرها:

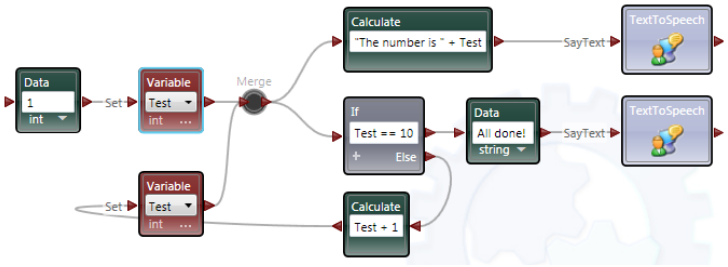
- Agilent VEE[428].
- National Instruments LabVIEW[429].
- Microsoft Visual Programming Language (MVPL)[430].

تعتبر البيئة "Agilent VEE" من لغات البرمجة الرسومية التدفقية التي تستخدم في تصميم تطبيقات القياسات والاختبارات المؤتمتة وتحليل البيانات، وهي تعتبر لغة برمجية رسومية موجهة للتطبيقات الصناعية. الشكل3-28 يبين أحد التطبيقات البرمجية باستخدام Agilent VEE Pro 9.2.



الشكل3-1 مخطط برمجي باستخدام البيئة Agilent VEE Pro 9.2

البيئة البرمجية "Microsoft Visual Programming Language" تعتبر أيضاً من لغات البرمجة الرسومية التدفقية، إلا أنها موجهة بشكل خاص لبرمجة تطبيقات الروبوت. الشكل3-29 يبين أحد تطبيقات الأوامر الصوتية في البيئة MVPL.

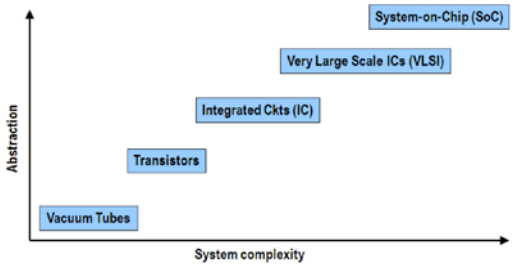


الشكل3-2 تحويل النص إلى الصوت في البيئة البرمجية MVPL

البيئة البرمجية LabVIEW تعتبر من أقوى وأكثر البيئات البرمجية الرسومية استخداماً وانتشاراً وتطبيقاً، وهي نقطة محورية في هذا البحث سنفصل فيها في ما يأتي في هذا الفصل.

أهمية لغات البرمجة الرسومية (The Importance of Graphical Programming):

لكي تكون ناجحاً في الاقتصاد العالمي اليوم، فإن مسألة وصول المنتج إلى السوق يجب أن تتم بشكل أسرع من السابق، وبالتالي فإن دورة تصميم المنتج يجب أن تكون أقصر ما يمكن. من جانب آخر فإنه في الوقت الذي تزداد فيه كثافة الترانزستورات على شريحة سيليكونية وحيدة – وفقاً لقانون Moor، فإن كلفة الترانزستورات على المستوى السيليكوني بانحدار، وبالتالي فإن العناصر المتكاملة المعقدة البنية (FPGAs, Multi-core MPUs, SoCs) أصبحت أكثر استخداماً وشيوعاً في التطبيقات، وهذا بدوره أدى إلى حجم تعقيد برمجي أكبر بكثير ودورة تصميم أطول بكثير. الشكل3-30 يبين منحني تطور العناصر المتكاملة ودرجة تعقيد النظام.

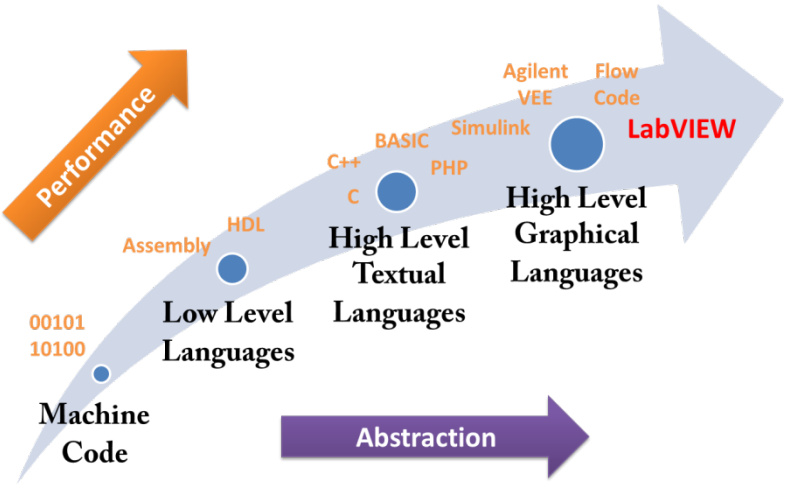


الشكل3-3 درجة التعقيد للنظام ومستوى التجريد لتطور تقنيات الدارات المتكاملة

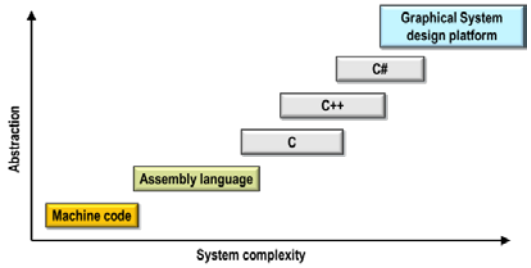
عند استخدام لغات البرمجة التقليدية النصية وبعد الانتهاء من كتابة البرنامج، فإن على المهندسين المصممين أو الدارسين (الطلاب) المرور بالعديد من المراحل المرهقة قبل توليد الملف البرمجي النهائي الذي يتم برمجته على شريحة الFPAG، فيجب إعادة كتابة أو تعديل البرنامج بحيث يكون قابلاً للترجمة (Synthesizable) من قبل المترجم المحدد. أضف إلى ذلك أن لكل مترجم متطلبات خاصة تختلف عن غيره من المترجمات الأخرى تبعاً للشركة المطورة للمترجم، لهذا السبب فإن المصممين أو الباحثين يصرفون وقتاً كبيراً في دراسة المتطلبات الخاصة للأدوات البرمجية التي سيعملون عليها بدلاً من صرف الوقت على متطلبات التصميم نفسه.

تؤكد الأبحاث على ضرورة تطوير وتبني بيئات برمجية جديدة على مستوى جديد، وذلك بعيداً عن اللغات النصية لأن حجم البرنامج يزداد طولاً وتعقيداً – مثل: البيئات الرسومية – إضافةً إلى البيئة الأساسية بلغة الC بحيث يمكن البرمجة بكلا المنحيين بنفس الوقت وضمن بيئة برمجية واحدة، بما في ذلك مراحل التحليل والفحص والتنفيذ[431,432].

البحث[433] يشير إلى أنه من أجل برمجة الأنظمة المدمجة عموماً، وتقنية الFPGA على نحو خاص، فإنه من الضروري جداً وجود تحول أو انتقال جذري في المنهجية البرمجية المتبعة. كما تؤكد الأبحاث[434-436] على أن لغات البرمجية الرسومية مناسبة بشكل كبير لتصميم وبرمجة الأنظمة المدمجة؛ نظراً لارتكازها على منهجية تدفيق البيانات (Dataflow). الشكل3-31 يبين منحنى تطور الأنظمة البرمجية على المستوى البنيوي. الشكل3-32 يبين مخطط تطور اللغات البرمجية المخصصة لبرمجة الأنظمة المدمجة.



الشكل3-4 مخطط تطور البرمجة الحاسوبية الموافق لتطور الكيان الصلب



الشكل3-5 مخطط تطور اللغات البرمجية الموافق لدرجة تعقيد الكيان الصلب

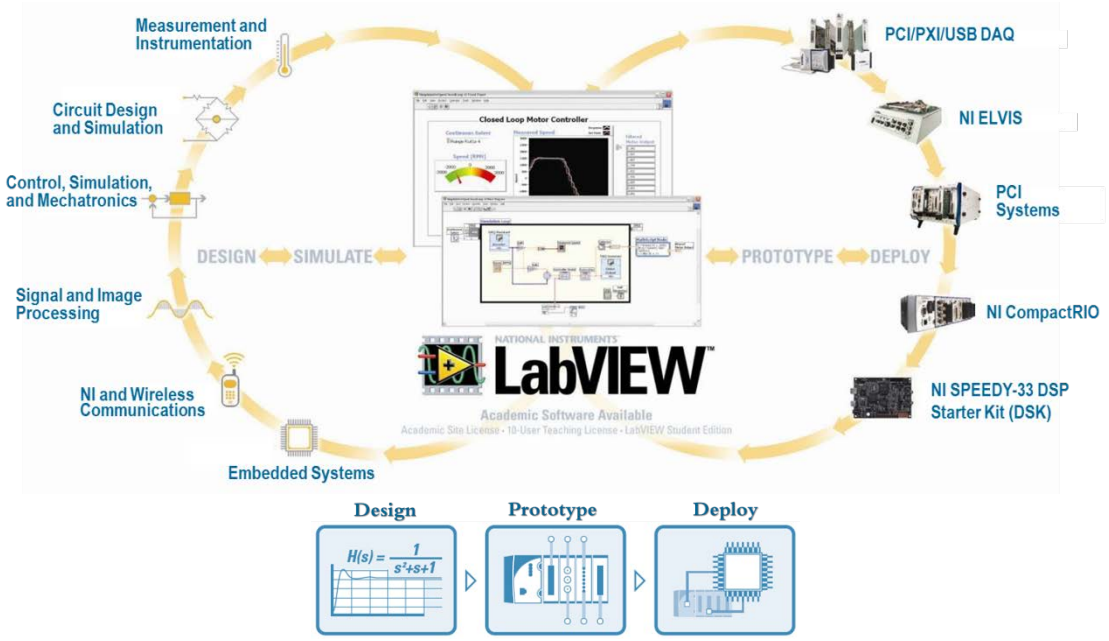
لقد أثبتت لغات البرمجة الرسومية فعاليتها على لغات البرمجة النصية، كما أنها أسرع بخمس مرات من اللغات النصية في تطوير التطبيقات[437]. علاوةً على ذلك فإن لغات البرمجة الرسومية تعزز الإنتاجية لدى الباحثين ومطوري التطبيقات بغض النظر عن مستوى خبرتهم البرمجية[438]، وذلك لأن اللغات الرسومية تعطي تنظيماً بدهياً، وتجعل المعلومات واضحة ومرئية[439]، الأمر الذي يجعل عملية كتابة أو تحويل الخوارزمية البرمجية من مخطط تدفقي (Flowchart) إلى برنامج أمراً بدهياً.

البيئة البرمجية LabVIEW (LabVIEW Programming Environment):

البيئة LabVIEW – اختصاراً لـ"Laboratory Virtual Instrumentation Engineering Workbench" – عبارة عن لغة برمجية رسومية تم تطويرها من قبل شركة National Instruments[428] في بدايات الـ1980s بهدف إيجاد أداة برمجية فعالة وتفاعلية لتطوير البرامج الخاصة بأنظمة استحصال البيانات وتجهيزات القياسات[422].

تمكن هذه البيئة البرمجية الطلاب والمهندسين والباحثين في مختلف الفروع الهندسية والمختصين في فروع العلوم، من التصميم التفاعلي (Design) وبناء النماذج الأولية (Prototype) والتطبيق العملي (Deploy) للأنظمة المدمجة بمختلف تقنياتها وتطبيقاتها (MCUs, PLC, Vision,) والأنظمة الصناعية والقياسات وتطبيقاتها (MPUs, Multi-core, FPGAs, DSPs,… Communications, Control, Measurements, Mechatronics) خلال زمن قصير، وذلك باستخدام مكتبات رسومية

نموذجية إضافةً إلى تضمين مكتبات أو برامج خارجية جاهزة (C, HDL, .m file) لبناء تطبيق موثوق يتم برمجته على الكيان الصلب مباشرةً دون أي مراحل تصميم كيان صلب مسبقة. الشكل3-33 يبين بعض التطبيقات الأساسية للبيئة البرمجية LabVIEW.

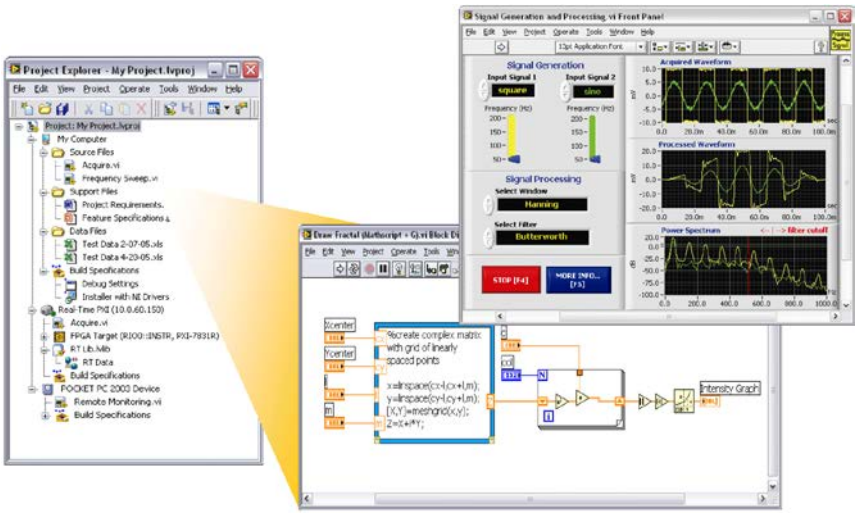


الشكل3-6 بعض التطبيقات الأساسية للبيئة البرمجية LabVIEW

عناصر البيئة البرمجية LabVIEW (The LabVIEW Environment Parts):

تتألف البيئة البرمجية الأساسية للبرنامج من:

1- واجهة المستخدم الرسومية (Front Panel)
2- واجهة البرمجة الرسومية (Block Diagram)
3- مدير ومستعرض المشروع (Project Explorer)

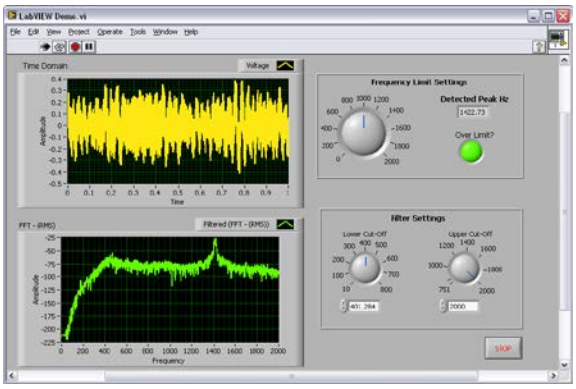


الشكل 3-7 الوجهات الأساسية والمستعرض للبيئة البرمجية LabVIEW

الشكل3-8 مجموعة منتقاة من أكبر الشركات العالمية التي تستخدم البيئة LabVIEW

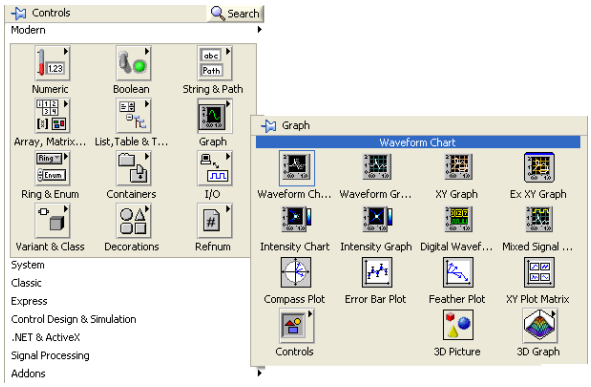
واجهة المستخدم الرسومية (Front Panel)

وهي واجهة تحكم تفاعلية مرئية للمستخدم (GUI)، تضم مجموعة عناصر تحكم وإظهار وظيفية تدعى بـ"Controls" (عناصر دخل وخرج وإظهار مرئية) يتم إضافتها من لوحة عناصر التحكم (Controls Palette). الشكل3-36 يبين مثالاً لواجهة المستخدم للتطبيقات البرمجية في بيئة LabVIEW.



الشكل3-9 واجهة المستخدم للتطبيقات البرمجية في البيئة LabVIEW

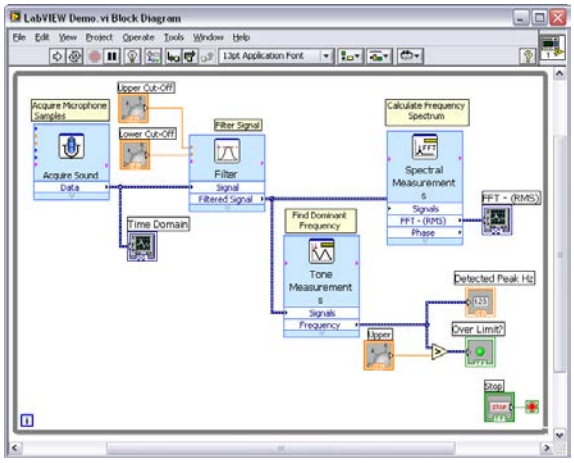
إن عناصر التحكم في واجهة المستخدم مرتبطة بشكل مباشر بالعناصر الوظيفية في الواجهة البرمجية، حيث أنه بإضافة أي عنصر في واجهة التحكم، سيتم إضافة العنصر الوظيفي له في الواجهة البرمجية آنياً، وبالتالي يمكن بناء واجهة المستخدم بالكامل ثم توصيل عناصر التحكم الوظيفية في الواجهة البرمجية. الشكل3-37 يبين لوحة "Controls Palette".



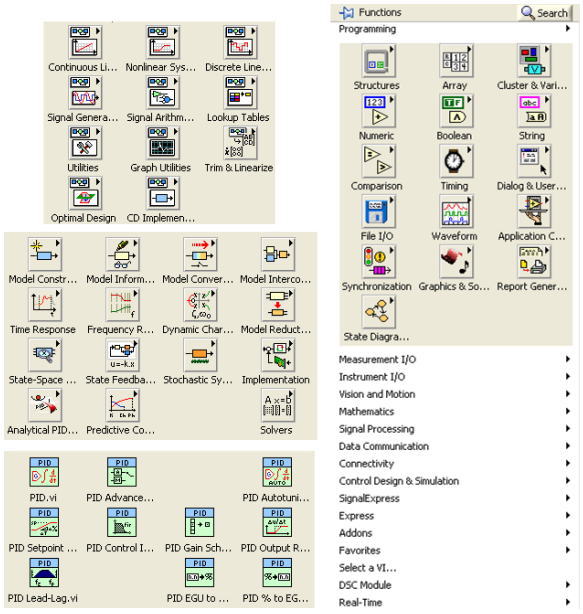
الشكل3-10 لوحة عناصر التحكم "Controls Palette" في واجهة المستخدم في البيئة LabVIEW

واجهة البرمجة الرسومية (Block Diagram):

وهي الواجهة البرمجية الرسومية (مشابهة للمحرر البرمجي النصي في لغات البرمجية التقليدية C, C++, Jave)، تضم العناصر والمكتبات البرمجية الوظيفية التي يتم إضافتها من لوحة العناصر الوظيفية (Functions Palette)، هذه العناصر والمكتبات تم بناؤها باستخدام العناصر الرسومية ولكن عند مستوى برمجي أخفض. الشكل3-38 يبين مثالاً لواجهة البرنامج للتطبيقات البرمجية في بيئة LabVIEW. الشكل3-39 يبين لوحة العناصر الوظيفية "Functions Palette".



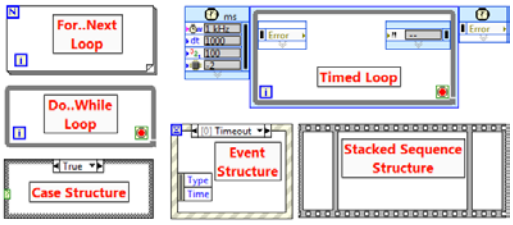
الشكل3-11 واجهة البرنامج للتطبيقات البرمجية في البيئة LabVIEW



الشكل3-12 لوحة العناصر الوظيفية "Functions Palette" في الواجهة البرمجية في البيئة LabVIEW

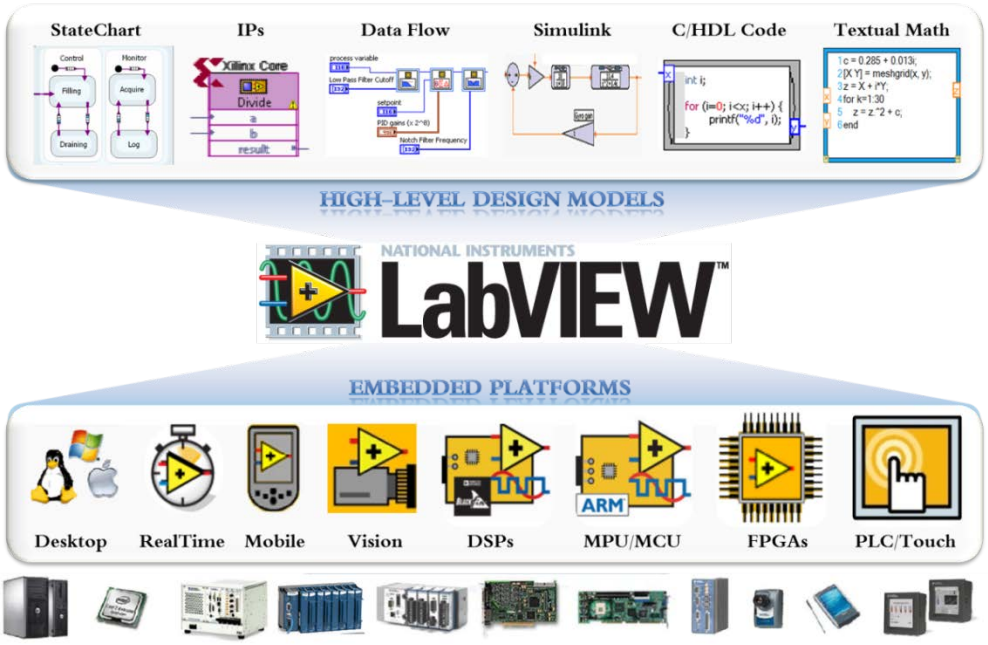
بشكل مشابه للغات النصية فإن البيئة LabVIEW تستخدم الحلقات (مثل: For..Next, Do..While) بشكل رسومي للتحكم بالعمليات التكرارية، كما تستخدم التوابع الشرطية (مثل: If...Then) لمقارنة الشروط، إضافةً إلى العديد من الحلقات المتزامنة وعناصر تنفيذ متسلسل والعديد من العناصر الأخرى.

الشكل3-40 يبين أكثر العناصر الوظيفية الشرطية والحلقات استخداماً في البيئة LabVIEW.



الشكل 3-13 العناصر الوظيفية للحلقات الشرطية في البيئة LabVIEW

إن التصميم والبرمجة في البيئة LabVIEW لا يقتصر فقط على استخدام لغة الG الرسومية، وإنما يوجد العديد من الطرق عالية المستوى لبناء التصميم (High-level Design Models)، حيث يمكن بناء التطبيق باستخدام Simulation-Module المشابه لبيئة البرنامج Matlab-Simulink، كما يمكن استخدام StateChart Module، أو تضمين ملفات برمجية نصية خارجية (HDL, .m file, C/C++) باستخدام العقد المخصصة لذلك. الشكل3-41 يبين طيفاً واسعاً من الطرق البرمجية التي يمكن استخدامها في تصميم وبرمجة حلول الكيان الصلب المبينة على الشكل.



الشكل3-14 الحلول البرمجية وحلول الكيان الصلب في البيئة LabVIEW

LabVIEW والبرمجة الرسومية التدفقية (LabVIEW, "G" Dataflow Programming):

تختلف بيئة LabVIEW عن معظم لغات البرمجة الأخرى في كونها تستخدم لغة برمجة رسومية تدعى بـ"G" (Graphical) تقوم على مبدأ توصيل أيقونات رسومية على شكل مخطط، الذي يُترجم مباشرةً إلى لغة الآلة حتى تستطيع المعالجات الموجودة في الحاسب تنفيذه، وعلى الرغم من كونها تُمثَّل بشكل رسومي عوضاً عن الشكل النصي؛ فإنَّ لغة "G" تمتلك نفس المبادئ البرمجية المتَّبعة في معظم لغات

البرمجة التقليدية. على سبيل المثال، تمتلك لغة البرمجة "G" جميع البُنى النظامية التي تحتويها لغات البرمجة مثل: أنماط البيانات، الحلقات، المتحولات، العُودية (recursion)، إدارة الأحداث (event handling)، والبرمجة غرضية التوجُّه (object-oriented programming).

الصفة المهمة الأخرى التي تميِّز البيئة البرمجية LabVIEW عن غيرها من لغات البرمجة التقليدية، هي كون لغة الـG المطوَّرة فيها تُنفَّذ وفقاً لقواعد تدفُّق المعطيات (Dataflow) عوضاً عن الطريقة التقليدية الإجرائية التي تعتمد على تنفيذ عددٍ من الأوامر (الإجراءات) المتسلسلة كما في معظم لغات البرمجة النصية – كلغة C & C++.

تعتمد لغة البرمجة G على منهجية البرمجة التدفقية (Dataflow) التي فيها يكون خرج كل عقدة برمجية حسابية محسوب عندما تكون جميع القيم محددة على مداخل العقدة؛ حيث أن تدفُّق البيانات بين عُقد البرنامج –وليس أسطر التعليمات المتسلسلة– هو ما يحدِّد أولوية التنفيذ، كما أن العمليات الحسابية يمكن أن تكون مزامنة للعقد التي لا تكون مداخلها متعلقة بمخارج عقد أخرى؛ ربما تبدو هذه الصفة ضئيلة الأثر للوهلة الأولى، ولكنَّها في الحقيقة ذات تأثير استثنائي؛ لأنها تجعل من المسارات التي تسلكها البيانات بين أجزاء البرنامج المختلفة موضع الاهتمام الأول للمبرمج.

تمتلك العقد (التوابع، البُنى كالحلقات، البرامج الفرعية، وغيرها) في بيئة LabVIEW مداخلاً لقراءة البيانات، وحالما تحتوي جميع مداخل عقدةٍ ما على بياناتٍ مناسبة، تقوم هذه العقدة بتنفيذ العمليات المنطقية المناطة بها، ثمَّ تولِّد البيانات المناسبة على مخارجها، وتمرِّر هذه البيانات إلى العقدة التالية في مسار تدفُّق البيانات؛ إنَّ العقدة التي تستقبل بيانات ما من عقدة أخرى، تستطيع تنفيذ تعليماتها فقط بعد أن تُنهي تلك العقدة تنفيذ تعليماتها بشكلٍ كامل.

فوائد لغة البرمجة الرسومية "G" (Benefits of G Programming):

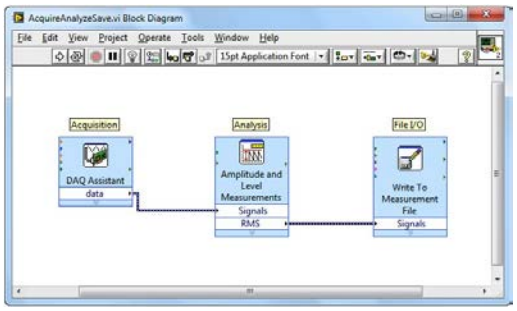
فيما يلي نلخص بإيجاز الميزات والفوائد الهامة للغات البرمجية الرسومية.

البرمجة الرسومية حدسية بديهية (Intuitive Graphical Programming):

كما هو الحال لدى معظم الناس، يتعلم المهندسون والعلماء عن طريق مشاهدة ومعالجة الصور بدون الحاجة إلى تأمل دقيق للموضوع؛ يمكننا أيضاً وصف العديد من المهندسين والعلماء "كمفكرين بصريين" (Visual Thinkers)، مما يعني أنهم ماهرون بشكلٍ خاص في استخدام الطرق الرسومية لتوصيف وتنظيم المعلومات. إن هذه الميزة يتم تنميتها غالباً في المدارس والجامعات، حيث يتم تشجيع الطلاب على إيجاد حلول لمسائل مختلفة بشكل مخططات وظيفية. بالرغم من ذلك، فإن معظم لغات البرمجة عامة الاستخدام تتطلب من المبرمج جهداً إضافياً ليتعلم التعليمات النصية المحددة الخاصة بهذه اللغة، ومن ثمَّ عليه إسقاط بنية هذه اللغة على المسألة المدروسة.

إن البرمجة الرسومية باستخدام لغة G تتيح للمبرمج تجربة أكثر بديهية وأكثر انسجاماً مع تفكيره الفطري، ذلك لأنها تعتبر أكثر سهولةً للفهم والاستيعاب على اعتبار أنه متآلف كلياً مع التمثيل الرسومي ونمذجة العمليات بشكل مخططات منهجيةٍ أو تدفقية (والتي تتَّبع

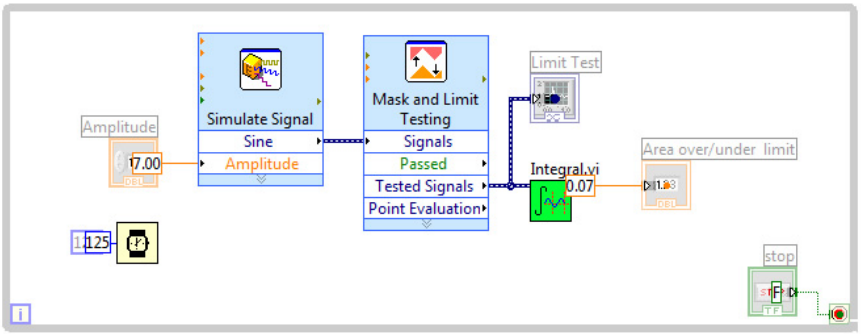
قواعد تدفُّق البيانات)، بالإضافة إلى ذلك، وبما أنَّ لغات البرمجة المقادة بالبيانات تتطلب من المبرمج أن يجعل تدفُّق هذه البيانات المحور الرئيس في البرنامج، فإنَّ هذا يشجِّع المبرمج بالتفكير في المسالة التي يحلُّها عوضاً عن التفكير في أسلوب برمجتها؛ على سبيل المثال، قد يبدأ برنامجٌ نموذجيٌ مكتوب بلغة G بتحصيل عدة قنوات تحمل بيانات عن درجة الحرارة، ثمَّ يقوم بتمرير هذه البيانات إلى تابع معالجة وحساب، وأخيراً يقوم بتخزين البيانات المعالجة على القرص، كما هو مبين على الشكل3-42 فإن تدفُّق البيانات والخطوات التي يتضمَّنها هذا البرنامج تُعتبر سهلة الفهم إجمالاً ضمن مخطط بيئة LabVIEW.



الشكل3-15 المخطط البرمجي لاستحصال بيانات ومعالجاتها وتخزينها في البرنامج LabVIEW

أدوات التنقيح والفحص التفاعلية (Interactive Debugging Tools):

بما أن لغة البرمجة الرسومية G في بيئة LabVIEW سهلة الفهم، فإنَّ هذا يجعل من المهام البرمجية الشائعة كتنقيح الأخطاء أمراً روتينياً وبديهياً أيضاً. على سبيل المثال، تقدم بيئة LabVIEW أدوات تنقيح فريدة من نوعها تتيح للمبرمج مشاهدة البيانات بشكلٍ تفاعلي وهي تنتقل عبر الأسلاك من عقدةٍ لأخرى (Execution Highlighting).



الشكل3-16 استخدام خاصية التنقيح "Execution Highlighting" لمراقبة تدفق البيانات بين العقد في البيئة LabVIEW

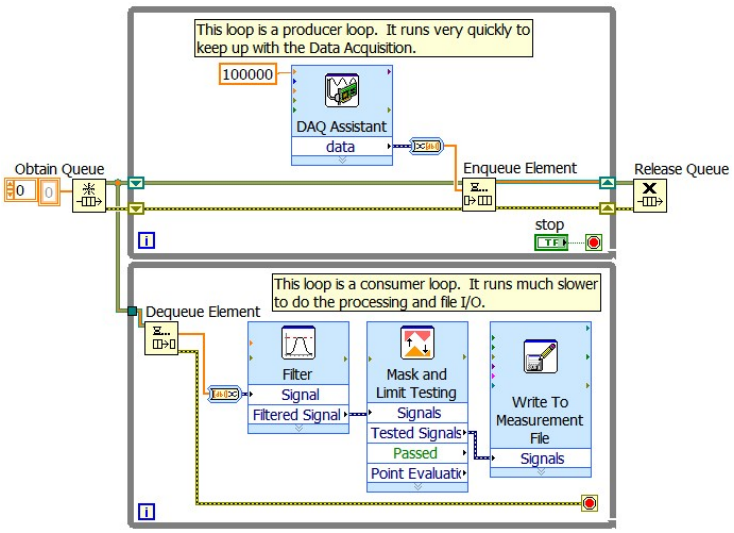
تحتوي البيئة LabVIEW أيضاً على أدوات تنقيح للغة G مماثلة لتلك الموجودة في بيئات البرمجة التقليدية الأخرى، تتضمَّن الأدوات: نقاط مراقبة (Probes)، نقاط توقُّف (Break Points)، تشغيل خطوة بخطوة (Step-by-Step).

تُمكِّن أدوات التنقيح الخاصة بلغة G المبرمج من استحصال البيانات من عدَّة أجزاء في البرنامج بنفس الوقت، كما تعطيه إمكانية الإيقاف الآني والدخول إلى برنامج فرعي بدون الحاجة إلى تعليمات برمجية معقَّدة؛ ورغم أنَّ هذه الإمكانيات متوفِّرة في لغات البرمجة الأخرى، إلاَّ أنَّ بيئة LabVIEW تجعل من السهل جداً تصوُّر حالة البرنامج والعلاقات بين الأجزاء التفرعية فيه بسبب الطبيعة الرسومية، كما تُعتبر

أداة المترجم الآني إحدى أبرز أدوات تنقيح الأخطاء المستخدمة في بيئة LabVIEW، حيث أنه أثناء قيام المبرمج بتطوير البرنامج، تقوم هذه الأداة بتفحُّص الأخطاء بشكلٍ آني، وتقدِّم اقتراحات للمبرمج حول الأخطاء البرمجية وطريقة حلها.

التوزيع التلقائي لمهام التنفيذ والأداء (Automatic Parallelism and Performance):

تسمح لغات البرمجة المقادة بالبيانات كما في بيئة LabVIEW بالحصول بشكلٍ تلقائي على تفرُّعٍ في التنفيذ. وبعكس لغات البرمجة التسلسلية كلغة C ولغة C++، فإنَّ البرامج الرسومية تحتوي بشكلٍ أساسي على معلومات عن أجزاء البرنامج التي تحتاج إلى التنفيذ على التوازي مع أجزاء أخرى. على سبيل المثال، يتعبر نمط المنتج/المستهلك (Producer/Consumer) أحد أساليب التصميم الشائعة في لغة البرمجة G، ففي هذا الأسلوب تعمل حلقتا While على التوازي وبشكل مستقل، بحيث تكون الحلقة الأولى مسؤولةً عن توليد البيانات والحلقة الثانية مسؤولةٌ عن معالجتها، ويتم تبادل البيانات بين الحلقتين باستخدام أدوات تدعى بـQueues، وهي من أنماط البيانات التقليدية في لغات البرمجة عامة الاستخدام. الشكل3-44 يوضح النمط البرمجي Producer/Consumer.

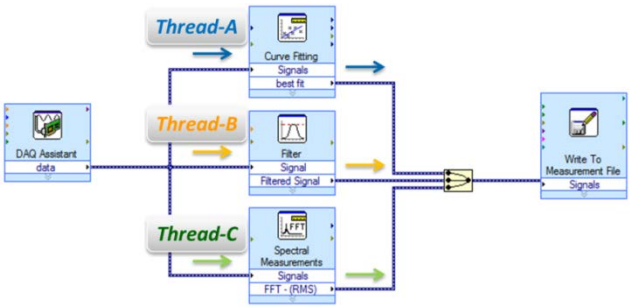


الشكل3-17 تنفيذ المهام التفرعية في البيئة LabVIEW باستخدام النمط Producer/Consumer

إن خاصية التفرعية تُعتبر أمراً بالغ الأهمية في برامج الحاسب، وذلك لكونها قادرة على تخطي حدود الأداء الناتجة عن ضعف البرامج التسلسلية في التعامل مع التطورات الأخيرة في تصاميم معالجات الحواسيب. على مدى أكثر من 40 عام، قام مصنعو المعالجات الحاسوبية بزيادة تردد عمل المعالج لزيادة أدائه، في أيامنا هذه لم يعد هذا الأمر ممكناً نتيجةً للضوابط التي تحدُّ من الاستطاعة المستهلكة والطاقة الحرارية المبدَّدة في هذه المعالجات، ونتيجةً لهذا قام مصنعو المعالجات بالانتقال إلى تصاميمٍ جديدة تستخدم عدة نوى معالجة على شريحة واحدة.

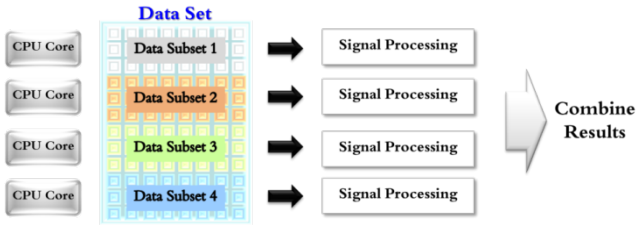
حتى يستفيد المبرمج من الأداء الكبير الذي تقدمه المعالجات متعددة النوى، يجب أن يكون قادراً على استخدام التقنيات البرمجية المتقدمة (Pipelining, Task and Data Parallelism) وتوزيع المهام (Multithreading) في برنامجه (أي بمعنى آخر تقسيم البرنامج إلى مقاطع منفصلة يمكن أن تُنفَّذ بشكلٍ مستقلٍ عن بعضها البعض). وبالتالي فإنه عند استخدام لغات البرمجة النصية التقليدية، سيصبح

المبرمج مسئولاً بشكلٍ مباشر عن إنشاء المسارات وإدارتها من أجل الحصول على مزايا التفرعية، وهو ما يُعتبر تحدِّياً كبيراً للمبرمجين المحترفين وغير المحترفين. على العكس تماماً، فإن خواص التفرعية الطبيعية في لغة البرمجة G تبسط استخدام تعدد المهام (Multitasking) في البرامج، حيث تُقوِّم البيئة LabVIEW آنياً أثناء التنفيذ الأجزاءَ التفرعية من البرنامج، وكلما صادفت تفرُّعاً في أحد الأسلاك، أو توضُّعاً متوازياً للعقد، تُنفذ البرنامج بشكلٍ تفرعي عبر استخدام عددٍ من المسارات التي تتحكم بها. تُدعى هذه الطريقة في الاصطلاحات العلمية الحاسوبية بالتوازي الضمني "Implicit Parallelism"، حيث أنَّ المستخدم لا يكتب برنامجه بهدف تنفيذه بشكلٍ تفرعي، وإنَّما تقوم لغة G بتنفيذ التفرعية تلقائياً من خلال تقسيم التطبيق إلى مسارات تنفيذ مستقلة موزعة. الشكل3-45 يبين التفرعية الطبيعية لتنفيذ المهام.

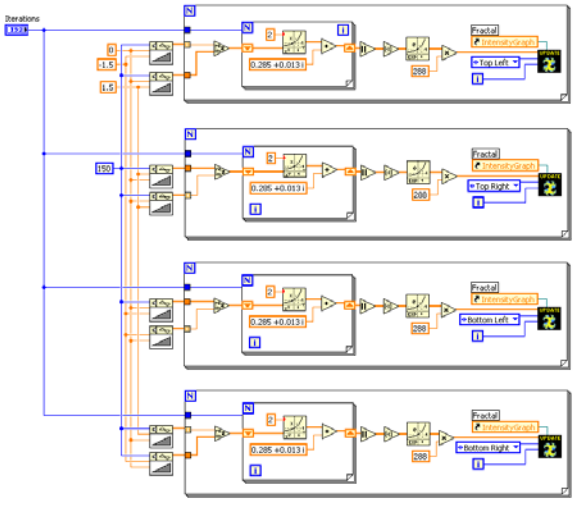


الشكل3-18 توزيع المهام التلقائي (Automatic Multithreading) في البيئة LabVIEW

الشكل3-46 يبين مبدأ توزيع البيانات المتوازي على نوى المعالجات وهذا المبدأ يستخدم عندما يراد معالجة ونقل كميات كبيرة من البيانات. الشكل3-47 يبين تطبيق "Data Parallelism" على معالج Quad-core في LabVIEW.



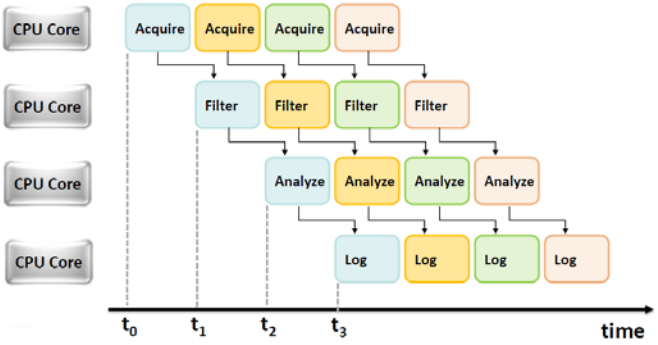
الشكل 3-19 مبدأ "Data Parallelism" على معالج Quad-core



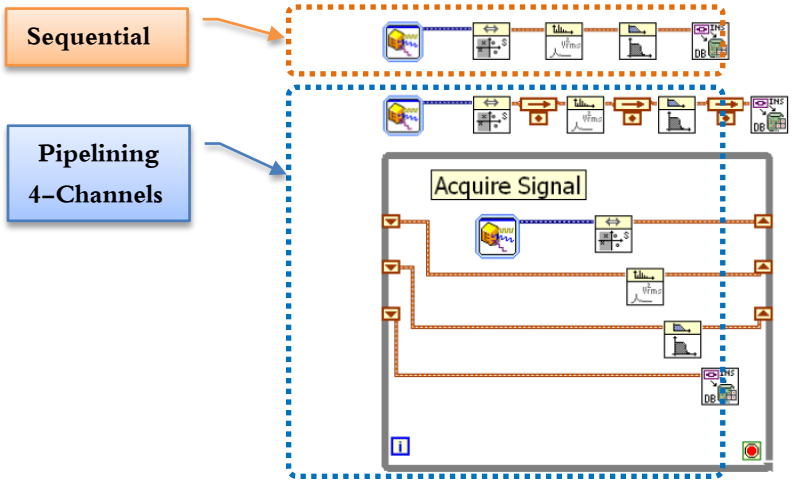
الشكل3-20 تطبيق "Data Parallelism" على معالج Quad-core في LabVIEW

الشكل3-48 يبين مبدأ المعالجة المتزامنة "Pipelining" في توزيع المهام البرمجية على معالج ذو نواة وحيدة تدعم أربع مستويات متزامنة.

الشكل3-49 يبين تطبيق مبدأ "Pipelining" بأربع مستويات في بيئة البرنامج LabVIEW.

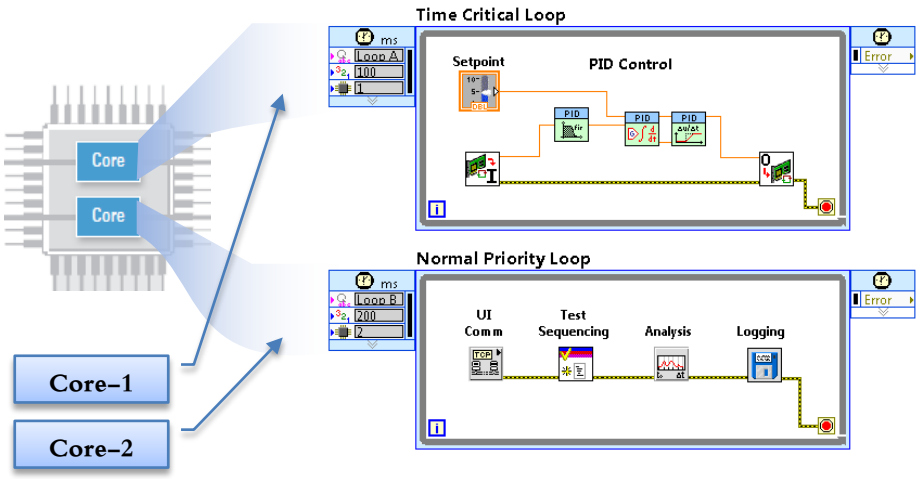


الشكل3-21 تمثيل المبدأ العام للمعالجة المتزامنة "Pipelining" بأربع مستويات



الشكل3-22 مقارنة بين المعالجة التسلسلية (Sequential) والمعالجة المتزامنة (4L.Pipelining) في بيئة البرنامج LabVIEW

الشكل3-50 يبين البرمجة والتوزيع المتوازي في الزمن الحقيقي (Real-time) للمعالجات متعددة النوى باستخدام الحلقات المتزامنة، وفيها يمكن تحديد نواة المعالجة المعنية بتنفيذ الحلقة في إعدادات الحلقة بإسناد رقم النواة (1,2,…n).

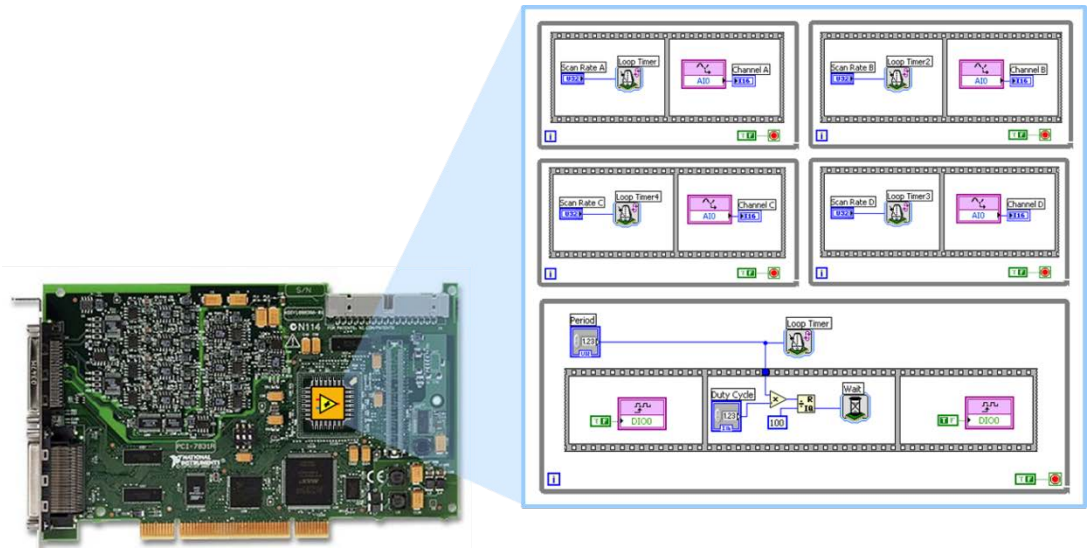


الشكل3-23 البرمجة في الزمن الحقيقي للمعالجات متعددة النوى في البيئة LabVIEW

البرمجة التفرعية على شرائح FPGA (Parallelism on FPGA Chips):

قبيل سنوات كانت مهمة برمجة شرائح البوابات المنطقية القابلة للبرمجة (FPGAs) سابقاً منوطة فقط بالخبراء المختصّين الذين يتمتَّعون بإلمامٍ كبيرٍ بلغات تصميم الكيان الصلب. وبشكلٍ متزايد، أصبح لدى المهندسين الذين يعملون في مجال برمجة المتحكمات المصغرة هاجس لتعلم برمجة تقنية الFPGA، بهدف استخدام هذه التقنية في المشاريع التي تتطلَّب إمكانيات ومتطلبات خاصة، وتحتاج دقة حتمية في التنفيذ، غير أن برمجة الFPGA تختلف كلياً عن برمجة المعالجات المصغرة، فمن أجل تصميم خوارزمية معينة باستخدام الFPGA؛ فإنه من الضروري عند كتابة البرنامج أخذ التنفيذ المتزامن للعمليات، والتنفيذ التفرعي، ومحدودية مصادر التخزين وغيرها بعين الاعتبار[433].

تُعتبر لغة G مناسبة بشكل فريد لبرمجة شرائح الFPGA، وذلك لأنها تعبِّر بشكل صريحٍ وواضح عن مبدأ التفرعية وتدفُّق البيانات، حيث تمكِّن لغة G المستخدم من الحصول على تنفيذٍ تفرعي حقيقي غير محدود (Parallelism) ، حيث يتم إسناد كل مهمة إلى قسم مخصَّص من الشريحة السيليكونية، ولكن في هذه الحالة لا يوجد محدودية في الأداء وفقاً لعدد نوى المعالجة المتوفّرة، وإنما يمكن بناء عدد كبير من المسارات المتوازية ككيان صلب، كما أن الأداء في أحد أجزاء البرنامج لا يتأثر سلباً بإضافة المزيد من مهمات المعالجة. الشكل3-51 يبين برنامجاً باستخدام البيئة LabVIEW FPGA مكون من خمس حلقات تنفيذ تعمل على التوازي.



الشكل3-24 برنامج باستخدام البيئة LabVIEW FPGA

اختصار المهمات والعمليات منخفضة المستوى (Abstraction of Low-Level Tasks):

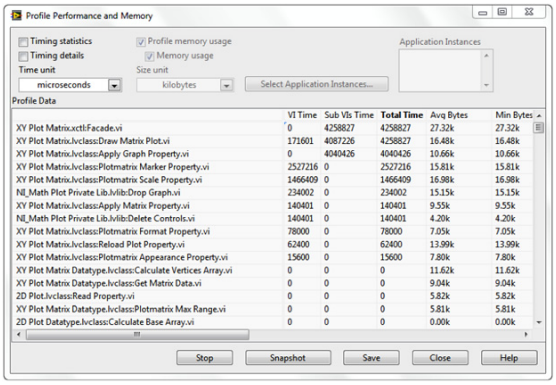
تُعدُّ عملية الاختصار والتجريد إحدى المزايا الأساسية في اللغات عالية المستوى، حيث أنَّها تعبِّر عن البرامج بطرق أخرى أكثر عفوية وأقرب إلى فطرة المبرمج وتفكيره. تقوم لغة البرمجة G تلقائياً بأداء الكثير من المهام التي يتوجب على المبرمج القيام بها في لغات البرمجة النصية (كالتعامل مع الذاكرة مثلاً)، حيث يتوجب على المبرمج في لغات البرمجة النصية حجز المواقع الذاكرية قبل التعامل معها، كما يتوجب عليه إنهاء حجز هذه المواقع عندما تنتهي الحاجة إليها. على المبرمج أيضاً أن يكون حذراً بحيث لا يتجاوز المواقع الذاكرية المحجوزة

عند الكتابة على الذاكرة. إن الفشل في حجز المواقع المطلوبة في الذاكرة، أو حجز مساحة غير كافية، يُعدُّ من أكبر الأخطاء الشائعة والصعبة التنقيح في لغات البرمجة النصية.

تُعتبر خاصية التعامل الآلي مع الذاكرة من أهم مزايا البرمجة باللغة G، حيث لا يحتاج المبرمج إلى حجز المتحولات أو التصريح عنها، كما لا يحتاج إلى الكتابة إلى هذه المتحولات أو القراءة منها، وإنما تقوم العقد التي تولِّد البيانات في بيئة LabVIEW تلقائياً بحجز الأماكن الذاكرية لهذه البيانات، وعندما تنتهي الحاجة إلى استخدامها يتم إلغاء حجز المواقع الذاكرية بشكل آلي. كذلك عند إضافة معلومات جديدة إلى مصفوفة أو سلسلة معرفية، يتم حجز مقدار إضافي من الذاكرة بشكلٍ تلقائي ليتسع لهذه المعلومات المضافة.

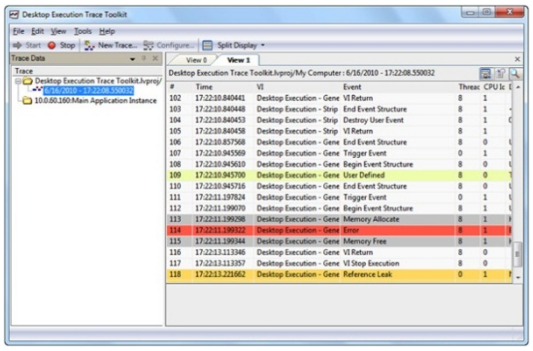
إن رفع مهام ومسائل إدارة الذاكرة منخفضة المستوى عن عاتق المبرمج، يحرِّره من دراسة القواعد المعقَّدة اللازمة لمنع حدوث أخطاء تشغيل في البرنامج، من أجل أن يركِّز اهتمامه على المسألة التي يقوم بحلِّها.

على الرغم من ذلك، فإن المبرمج يستطيع الوصول إلى تحكم دقيق باستخدام الذاكرة في اللغة G عند المستوى الأدنى باستخدام أدوات إدارة الذاكرة "Profile Performance & Memory" المدمجة ضمن بيئة LabVIEW؛ فإذا قرَّر المبرمج أنَّ استهلاك الذاكرة يشكِّل عاملاً مهماً في بيئة LabVIEW، يمكنه أن يتدخَّل ليخفِّض كمية الذاكرة المستهلكة عبر استخدام عدة تقنيات برمجية متقدِّمة. الشكل3-52 لوحة التحكم بالأداء وموارد الذاكرة في البيئة LabVIEW.



الشكل 3-25 لوحة التحكم بالأداء وموارد الذاكرة في البيئة LabVIEW

عندما تُظهر لغة البرمجة G سلوكاً غير متوقع لا يمكن حلُّه بسهولة باستخدام أدوات التنقيح المذكورة سابقاً، فعندها بإمكان المبرمج استخدام أدوات تنقيح أكثر تطوُّراً "LabVIEW Desktop Execution Trace Toolkit". تقدم هذه الأدوات إمكانيات أقوى للمبرمجين المحترفين الذين يحتاجون تحليلاً ديناميكياً للبرنامج عند مستويات منخفضة، مثل: كشف التسريبات في الذاكرة، عزل المصدر المسبِّب لحدث معيَّن أو سلوك غير مرغوب، تفحُّص البرامج بحثاً عن المواضع التي تُمكِّن من تطوير الأداء، إيجاد آخر عملية نداء حصلت قبل وقوع خطأ معين، التأكُّد من كون أداء برنامج معينَّ هو نفسه على أنظمة تشغيل ومنصات عمل مختلفة. الشكل3-53 لوحة الأداة "Execution Trace".

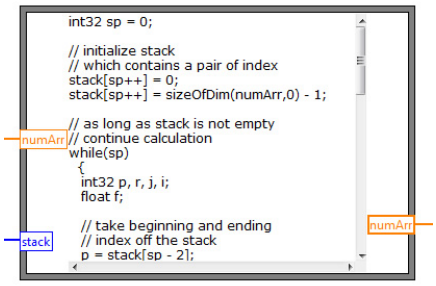


الشكل 3-26 أداة متقدمة للفحص وتتبع الأخطاء "Execution Trace" في البيئة LabVIEW

الجمع بين لغة G ولغات البرمجة الأخرى (Combining G with Other Languages):

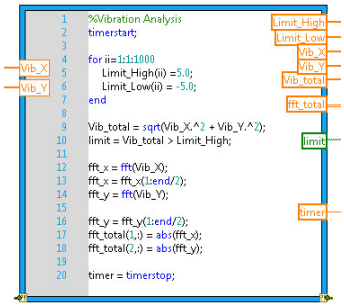
بالرغم من أنَّ لغة البرمجة G تقدِّم تمثيلاً ممتازاً للعمليات التفرعية، وتحرِّر المبرمج من تعقيدات فهم ذاكرة الحاسب والتعامل معها، إلا أنَّها غير مناسبة بالضرورة لأداء جميع المهام. بشكلٍ خاص، يمكن للعلاقات والصيغ الرياضية أن تمثَّل نصيًّاً بإيجازٍ وسهولة أكبر في بعض الأحيان، لهذا السبب، تتيح بيئة LabVIEW إمكانية الجمع بين البرمجة الرسومية وبين عدة أنواع من لغات البرمجة النصية، إذ يستطيع المبرمج في بيئة LabVIEW الاختيار بين البرمجة النصية والبرمجة الرسومية أو الجمع بينهما.

على سبيل المثال، تتيح بيئة LabVIEW استخدام ما يُسمَّى بعقد الصيغ الرياضية (Formula Node)، والتي تمكِّن المبرمج من كتابة صيغ رياضية نصية شبيهة بتلك المستخدمة في لغة C ضمن المخطط الصندوقي للبرنامج، بإمكان تلك الصيغ الرياضية أن تُنفَّذ جنباً إلى جنب وبشكل متكامل مع الوحدات البرمجية (الرسومية) في بيئة LabVIEW. الشكل3-54 يبين العنصر C-node المخصصة لكتابة برامج بلغة الـC/C++ ضمن بيئة LabVIEW.



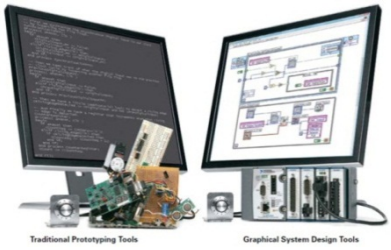
الشكل3-27 كتابة برامج بلغة الC ضمن بيئة LabVIEW باستخدام العنصر البرمجي C-node

بشكلٍ مشابه، تضيف عقدة النصوص الرياضية (MathScript Node) البرمجة النصية الرياضية إلى بيئة LabVIEW، وهي متوافقة بشكل عام مع صيغة الملفات ".m file" (Matlab) شائعة الاستخدام. الشكل3-55 يبين العنصر MathScript-Node المخصصة للتعامل مع صيغ الملفات من النوع ".m file" ضمن بيئة LabVIEW. إضافةً إلى ذلك يمكن تضمين برنامج وصف كيان صلب HDL باستخدام العقدة البرمجية HDL-Node.

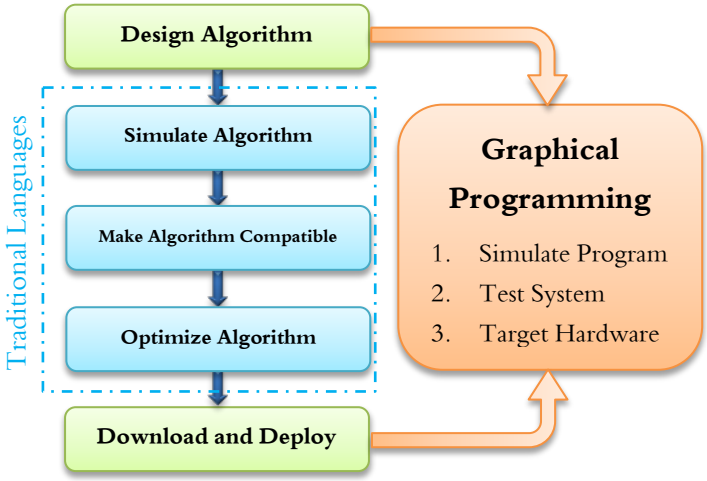


الشكل 3-28 التعامل مع الملفات ".m file" ضمن بيئة LabVIEW باستخدام العنصر MathScript-Node

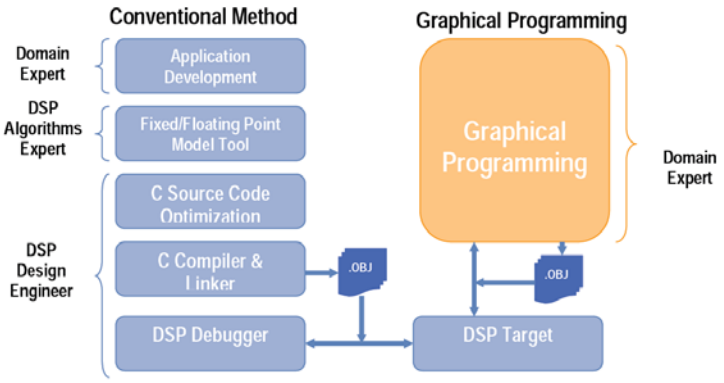
مقارنة بين لغات البرمجة الرسومية والنصية (Textual vs. Graphical Programming):



كما هو مبين على الشكل3-56 فإن على المبرمج الخوض في العديد من المراحل للوصول إلى مرحلة تشغيل البرنامج على الكيان الصلب، وهذه المراحل تتضمن كل منها منصة عمل مستقلة تحتاج إلى خبرة مرتبطة بالوظيفية البرمجية. في حين أنه وباستخدام البيئة LabVIEW فإن كامل عملية البرمجة والتحليل والتطوير تتم على منصة عمل وحيدة، وأما تفاصيل ومراحل توليد الملف البرمجي للكيان الصلب، فتتم بشكل مؤتمت من خلال تجريدها إلى مستوى البناء الأخفض – الذي يتم آلياً[444]. إن هذه الميزة تتيح للطلاب في الفروع الهندسية إمكانية تصميم النماذج وتنفيذها مباشرة على الكيان الصلب من خلال مستوى أعلى من التجريد لبيئة التصميم.



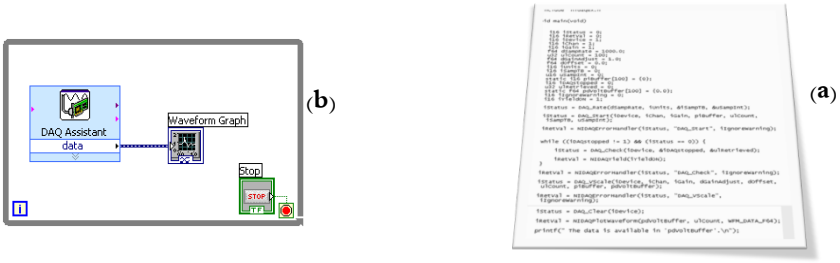
الشكل3-29 مقارنة الخطوات البرمجية بين لغات البرمجة النصية واللغات الرسومية – مستوى تجريد أعلى باستخدام لغات البرمجة الرسومية



الشكل3-30 يبين مقارنةً للخطوات البرمجية بين لغات البرمجة الرسومية واللغات النصية لبرمجة شريحة DSP.

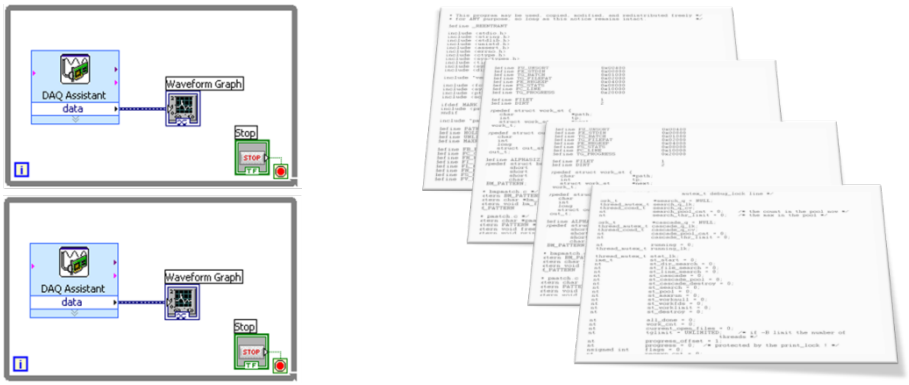
الشكل3-57 يبين مقارنةً بين البيئة LabVIEW-DSP واللغات التقليدية النصية للخطوات المطلوبة لبرمجة تطبيق عملي لشرائح معالجات الإشارة الرقمية، باستخدام البيئة LabVIEW يمكن تصميم التطبيق بدون الحاجة إلى كون المصمم متخصص في خوارزميات معالجة وتحليل الإشارة الرقمية؛ وذلك لأن معظم هذه الخوارزميات ستكون مبنية بالكامل على شكل صناديق وظيفية في بيئة LabVIEW، وكل ما سيحتاجه المصمم هو ضبط البارامترات الوظيفية لهذه العناصر، كما لن يحتاج المصمم الخوض في تعقيدات توليد الملف البرمجي للكيان الصلب ومسائل تبع الأخطاء.

الشكل3-58 يبين تطبيقاً لاستحصال البيانات (DAQ) يقوم على قراءة البيانات من جهاز القياس الموصول مع الحاسب عبر أحد منافذ الاتصال التسلسلي ويعرض النتائج على راسم إشارة على شاشة الحاسب. إن بناء مثل هذا التطبيق باستخدام برنامج LabVIEW سيستغرق أقل من عشر دقائق وسيكون مؤلفاً من عنصر التخاطب مع الجهاز وعنصر رسم الإشارة وحلقة تنفيذ تكرارية (58-3a). في حين أنه وباستخدام اللغات النصية فإن الأمر سيتطلب كتابة برنامج مؤلف من 50 سطراً من التعليمات البرمجية لا يتضمنها برنامج رسم الإشارات على الشاشة (58-3b).



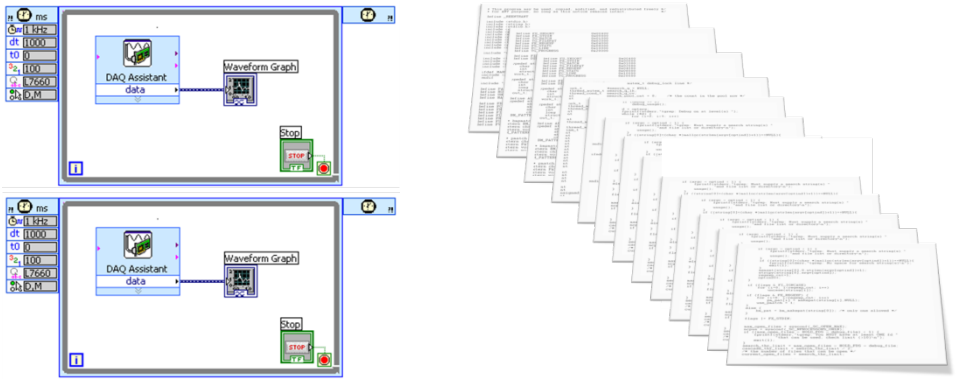
الشكل3-31 مقارنة بين اللغات النصية والرسومية لبرمجة حلقة استحصال بيانات من جهاز قياس وعرضها على راسم إشارة

من أجل تطوير التطبيق المبين على الشكل3-58 لاستحصال البيانات من حلقتين على التوازي – كما هو مبين على الشكل3-59- فإنه يكفي تكرار الحلقة الرسومية الأولى في بيئة LabVIEW وسيقوم البرنامج بتنظيم التنفيذ التفرعي وفقاً لعدد نوى المعالج. أما باستخدام اللغة النصية فإن الأمر سيتطلب إعادة بناء وتنظيم البرنامج وسيتضاعف حجمه خمس مرات.



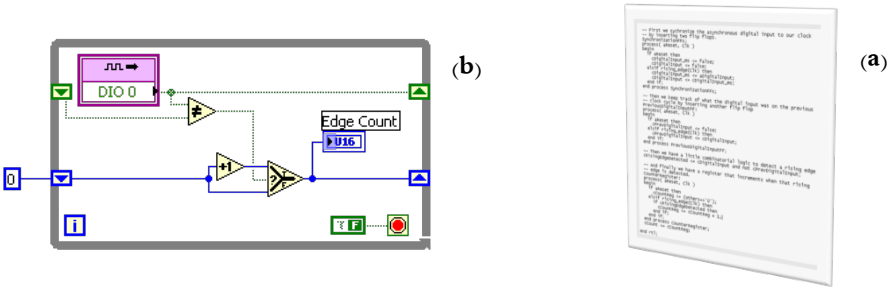
الشكل3-32 مقارنة بين اللغات النصية والرسومية لبرمجة حلقتي استحصال بيانات من جهازي قياس على التوازِ وعرضها

بفرض أننا نريد تطوير التطبيق المبين في الشكل3-59 لاستحصال البيانات عند معدلات مختلفة، فإننا سوف نحتاج إلى حلقة تكرارية زمنية لكل جهاز يتم معايرتها (زمن التكرار) وفقاً لمعدل القراءة المطلوب – الشكل3-60. إن هذا التعديل يمكن أن يتم بلغات البرمجة الرسومية بشكل بسيط جداً وذلك بتغيير الحلقة "Do-while" إلى حلقة زمنية "Times-loop"، في حين أن الأمر سيصبع معقداً جداً بالنسبة للغات الإجرائية النصية.



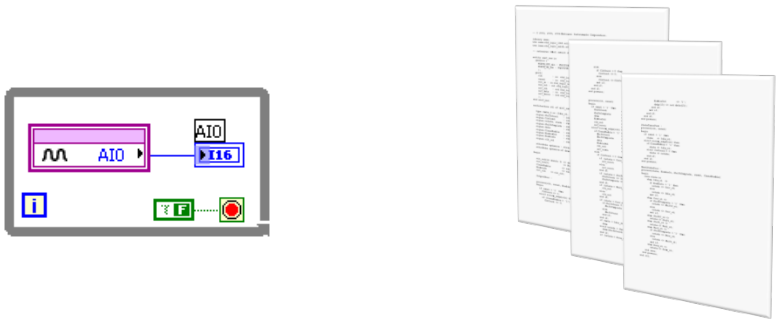
الشكل3-33 مقارنة بين اللغات النصية والرسومية لبرمجة حلقتي استحصال بيانات على التوازِ وبمعدلات استحصال مختلفة

بالانتقال إلى برمجة شرائح الFPGA، وبفرض أنه يراد قراءة بيانات من قطب رقمي (DIO: Data In/out) لشريحة الFPGA، فإن الشكل3b-61 يمثل التطبيق المطلوب. أما باستخدام لغات البرمجة النصية عالية المستوى فإن المسألة ستحتاج إلى كتابة برنامج مؤلف من 60 سطراً، أضف إلى ذلك تعقيدات توليد الملف البرمجي اللازم برمجة على الشريحة.



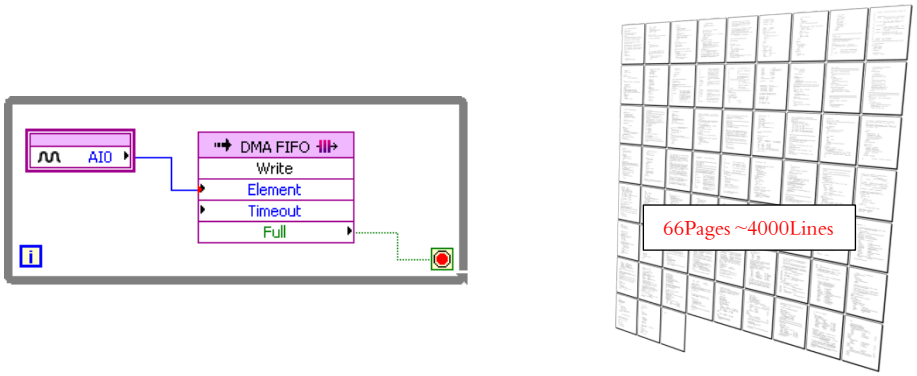
الشكل3-34 مقارنة بين اللغات النصية والرسومية لقراءة بيانات رقمية من قطب شريحة FPGA

من أجل تطوير التطبيق المبين في الشكل3-61 ليتم القراءة من قطب تشابهي، فإن التعديل سيكون بسيطاً جداً من خلال تعديل عنصر الدخل الرسومي إلى قطب تشابهي – الشكل3-62؛ في حين أنه في اللغات النصية سيتطلب الأمر تعقيداً كبيراً حول مسائل التعامل مع الفواصل العشرية (Floating point) وسيمتد البرنامج النصي إلى مئات الأسطر من التعليمات، كما أن تتبع الأخطاء في البرنامج سيتطلب وقتا أكبر.



الشكل3-35 مقارنة بين اللغات النصية والرسومية لقراءة بيانات تشابهية من قطب شريحة FPGA

لتطوير التطبيق المبين في الشكل3-62 بحيث يتم تخزين البيانات المستحصلة من القطب التشابهي في خلايا ذاكرية SRAM على شريحة الFPGA، فإن التعديل في بيئة LabVIEW سيكون بإضافة عنصر DMA-FIFO – الشكل3-63، في حين أنه باستخدام اللغات النصية سيمتد البرنامج إلى أكثر من 66 صفحة يحوي على 4000 سطر من التعليمات!



الشكل3-36 مقارنة بين اللغات النصية والرسومية لقراءة بيانات تشابهية من قطب شريحة FPGA وتخزينها

نتيجة (Conclusion):

في الوقت الذي تزداد فيه كثافة الترانزستورات على شريحة سيليكونية وحيدة – وفقاً لقانون Moor، فإن كلفة الترانزستورات على المستوى السيليكوني بانحدار، وبالتالي فإن العناصر المتكاملة المعقدة البنية (FPGAs, Multi-core MPUs, SoCs) أصبحت أكثر استخداماً وشيوعاً في التطبيقات، وهذا بدوره أدى إلى حجم تعقيد برمجي أكبر بكثير ودورة تصميم أطول بكثير.

إن تصميم الأنظمة المدمجة أصبح في هذا الوقت أمراً أساسياً في المناهج التعليمية الهندسية، وإن استخدام لغات البرمجة التقليدية النصية يعيق الطلاب والباحثين من الاستفادة من العديد حلول الكيان الصلب المتوفرة (مثل: الFPGAs) والتي يمكن أن تجعل العملية التعليمية

أكثر فعالية وواقعية. إن هذا الأمر يبدو جلياً وواضحاً في الفروع الهندسية التي لا تركز على البرمجيات (مثل: الفروع الهندسية كافةً عدا الهندسة المعلوماتية وهندسة الحواسيب)، حيث أن الطلاب لا يملكون الخبرة الوافية والكافية في لغات البرمجة النصية.

وعليه فإن الأبحاث على تؤكد ضرورة تطوير وتبني بيئات برمجية جديدة على مستوى جديد، وذلك بعيداً عن اللغات النصية لأن حجم البرنامج يزداد طولاً وتعقيداً – مثل: البيئات الرسومية – إضافةً إلى البيئة الأساسية بلغة الC بحيث يمكن البرمجة بكلا المنحيين بنفس الوقت وضمن بيئة برمجية واحدة، بما في ذلك مراحل التحليل والفحص والتنفيذ. لقد أثبتت لغات البرمجة الرسومية (LabVIEW) فعاليتها على لغات البرمجة النصية، كما أنها أسرع بخمس مرات من اللغات النصية في تطوير التطبيقات. علاوةً على ذلك فإن لغات البرمجة الرسومية تعزز الإنتاجية لدى الباحثين ومطوري التطبيقات بغض النظر عن مستوى خبرتهم البرمجية، وذلك لأن اللغات الرسومية تعطي تنظيماً بدهياً، وتجعل المعلومات واضحة ومرئية، وهو السبب الذي يجعلها محط الاهتمام في آلاف الجامعات حول العالم.

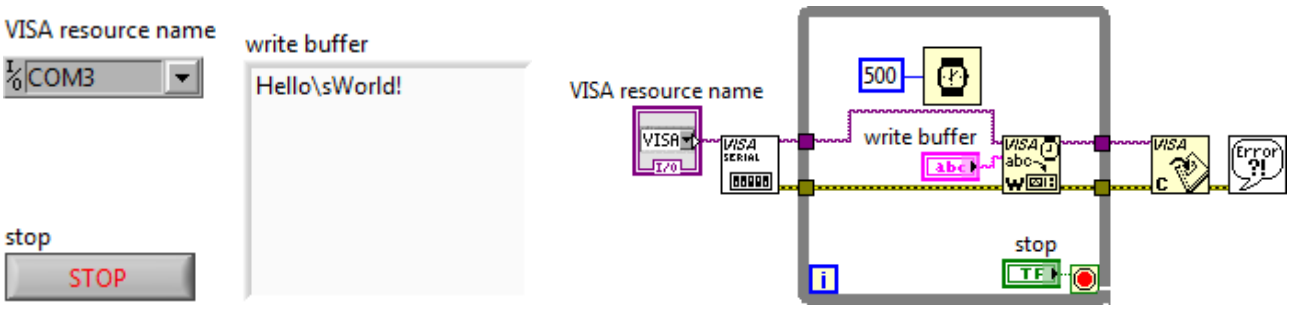
برمجة المنفذ التسلسلي في البيئة LabVIEW:

من أجل برمجة المنفذ التسلسلي في البيئة LabVIEW فإنه يجب تنصيب الموديولات التالية:

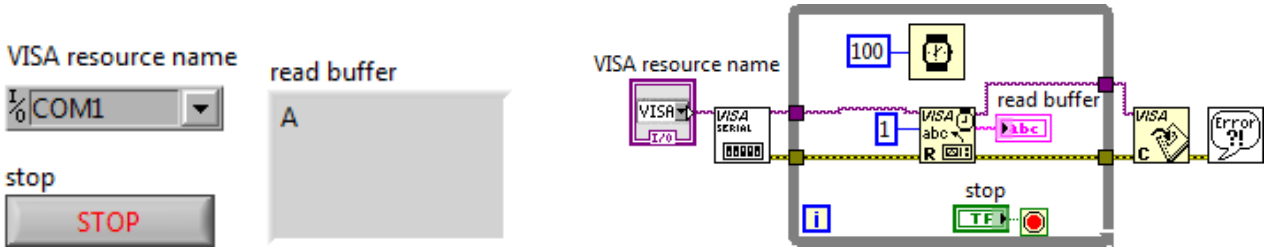
- NI LabVIEW 2011
- NI VISA 511full
- NI-VISA_Runtime

طريقة البرمجة تم شرحها في ملف الفيديو للجلسة الثالثة، حيث تم إنشاء برنامج للكتابة (Write.vi) إلى المنفذ التسلسلي COM وبرنامج آخر للقراءة (Read.vi) من المنفذ التسلسلي COM أيضاً...

واجهة المستخدم وبرنامج الكتابة إلى المنفذ التسلسلي في البيئة LabVIEW2011:



واجهة المستخدم وبرنامج القراءة من المنفذ التسلسلي في البيئة LabVIEW2011:



تمّ بنعمة من الله وفضل

وليد بليد

حلب – الإثنين، 30 نيسان، 2012