INTERFEȚE DE COMUNICAȚIE SERIALĂ LA MICROCONTROLLERE

INTERFEȚE DE COMUNICAȚIE

- Se foloseşte comunicația serială între microcontroller şi alte subsisteme, pentru economie de pini la capsulă
- Multe MC au incluse atât interfețe de comunicare sincrone cât şi asincrone
  - Interfața serială asincronă este numită de obicei “serial communication interface” (SCI sau UART)
  - Interfața serială sincronă e numită de obicei “serial peripheral interface” (SPI)
- De obicei (dar nu obligatoriu) o magistrală serială sincronă include o linie separată de clock
  - se simplifică astfel interfața între emițător şi receptor
  - linia de ceas e susceptibilă la zgomot în cazul distanțelor mari
- La magistrala serială asincronă ceasurile emițătorului şi receptorului sunt independente şi se face o re-sincronizare pentru fiecare bit supă bitul de start

INTERFEȚE SERIALE

- Utilizate, în general, pentru a conecta circuite periferice la microcontroller, în cadrul unui EmS
- Aceste interfețe permit conectarea de dispozitive diverse, cum ar fi:
  - ceas de timp real (RTC - real-time clock)
  - memorii nevolatile pentru stocarea unor parametrii
  - interfețe cu senzori
  - … şi multe altele
- Sunt interfețe de preț redus şi uşor de implementat

SPI - Serial Peripheral Interface

- SPI a fost dezvoltată de Motorola pentru a furniza o interfață simplă şi de cost redus între microcontrollere şi dispozitive periferice
- SPI e numită uneori interfață cu patru fire
- Poate fi utilizată pentru interfațarea diverselor dispozitive, cum ar fi:
  - memorie (stocare date)
  - convertoare analog-digitale
  - convertoare digital-analoge
  - RTC şi calendare RTC
  - LCD drivers
  - sensori
  - chip-uri audio
  - alte microcontrollere
  - .... domeniul componentelor cu interfață SPI creşte continuu

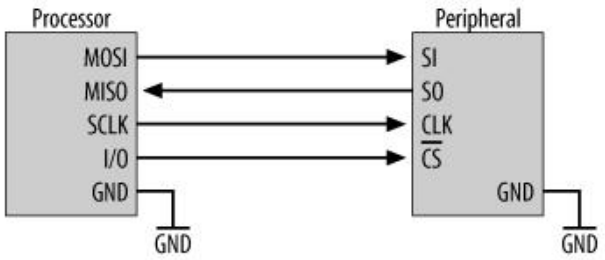
Serial Peripheral Interface

- SPI este o interfață serială sincronă
  - toate transmisiile sunt sincronizate cu un semnal de ceas comun furnizat de master (microcontroller)
- Perifericul receptor (“sclav”) utilizează semnalul de ceas la achiziția fluxului de biți seriali
- Există posibilitatea de a conecta mai multe dispozitive la aceeaşi interfață SPI a master-ului
  - Un master selectează un sclav prin activarea intrării sale de chip select
  - Un periferic ne-selectat nu va lua parte la transferul SPI

Serial Peripheral Interface

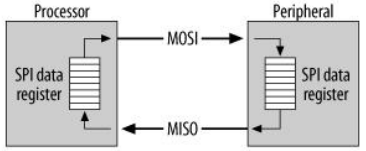
- SPI utilizează patru semnale:
  1. Master Out Slave In (MOSI)
  2. Master In Slave Out (MISO)
  3. Serial CLock (SCLK sau SCK)
  4. Chip Select (/CS) for the peripheral
- Unele procesoare au un semnal de chip select dedicat pentru SPI, numit Slave Select (/SS)
- MOSI este generat de master şi recepționat de scalv
  - La unele chip-uri MOSI este numit Serial In (SI) sau Serial Data In (SDI)
- MISO este produs de sclav, dar generarea sa este controlată de master
  - MISO este uneori numit Serial Out (SO) sau Serial Data Out (SDO)
- Semnalul de chip select către periferice este de obicei generat cu ajutorul unui pin de I/O digitală a masterului

INTERFAȚA SPI



TRANSMISIA SPI

- Master-ul şi scalvii conțin câte un registru cu deplasare serială a datelor (serial shift register)
- Master-ul începe transferul unui octet prin scrierea acestuia în registrul său SPI de serializare.
- Pe măsură ce se face transmisia serială către sclav (prin linia MOSI), sclavul transferă conținutul registrului său de serializare (prin linia MISO) către master
  - În acest fel se face schimb al informațiilor stocate în cele două registre de serializare
- Dacă se doreşte doar o scriere, master-ul va neglija octetul recepționat
  - În acest fel se face schimb al informațiilor stocate în cele două registre de serializare

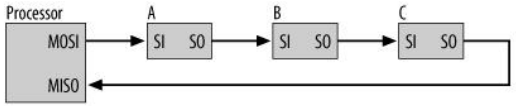


SPI

- Unele periferice cu interfață SPI permit transfer de mai mulți octeți
  - se transferă un flux continuu de date de la master
  - multe memorii cu interfață serială SPI lucrează în acest fel
  - la acest tip de transfer semnalul /CS rămâne activ pe toată durata transferului
  - de exemplu: o memorie așteaptă ca o comandă de “write” să fie urmată de patru octeți de adresă (adresa de start) şi apoi octeții de date ce vor fi stocați
- Alte periferice sclav aşteaptă doar un singur octet de control, cum ar fi de exemplu un convertor A/D
- Unele dispozitive sclav permit conectarea în lanț de priorități (daisy-chained)

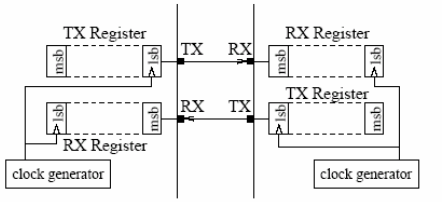
DAISY-CHAINED TRANSFER

- În acest exemplu procesorul (master) transmite trei octeți prin interfața SPI
  - Primul octet e transferat în sclavul A
  - La transferul celui de-al doilea octet în A, primul e transferat către B
  - Al treilea octet în A, al doilea în B şi primul în C
- Dacă masterul doreşte să citească un rezultat din A, el va transfera o secvență fictivă de 3 octeți prin MOSI
- Conectarea în lanț nu funcționează cu orice dispozitiv SPI (niciodată cu dispozitivele de memorie care cer un transfer multi-octet)

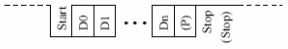


SCI

- SCI (UART) este o interfață de comunicație asincronă (Universal Asynchronous Receiver Transmitter, UART)
- UART utilizează două fire transmit (TXD) şi receive (RXD), pentru comunicație full- sau half-duplex.
- Standardul RS-422 e proiectat de asemenea pentru comunicație punct la punct dar utilizează două linii diferențiale pentru transmisie (atât RXD cât şi TXD constau din perechi de fire torsadate)
- RS-422 se foloseşte la distanțe mai mari şi este o alegere bună pentru medii cu zgomot.
- UART :

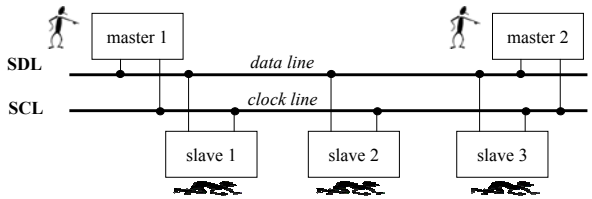


UART frame format



INTERFEȚE SERIALE SINCRONE

- USART (Universal Synchronous Asynchronous Receiver Transmitter) are o a treia linie pentru ceas.
- IIC (I^2C) – (The Inter-IC bus) este o magistrală sincronă multi-master ce funcționează pe principiul master – slave.
  - Foloseşte două linii, pentru comunicație half-duplex :
  - SCL (Serial Clock Line)
  - SDL (Serial Data Line), numită uneori SDA (Serial DAta)

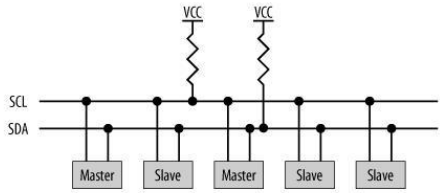


Generalități I²C

- I²C (Inter-Integrated Circuit numit şi IIC) este o magistrală ieftină şi eficientă
- I²C este o magistrală sincronă, bidirectională, de viteză mică
- Conectarea / deconectarea unor dispozitive la I²C nu afectează alte dispozitive
- I²C este o magistrală multi-master
- Viteza datelor în mod standard este de 100 kbps, iar în mod extins de 400 kbps
- Fiecare dispozitiv conectat la I²C are o adresă unică şi poate lucra ca transmițător sau receptor
- I²C mai este numită TWI (Two-Wire serial Interface)

Generalități I²C

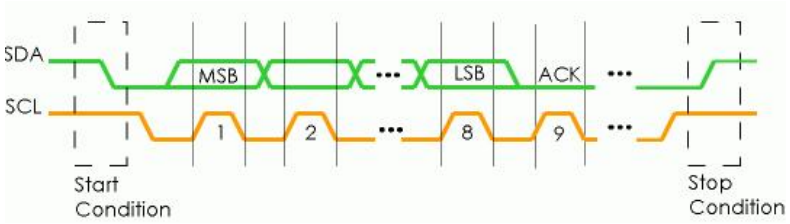
- Ambele linii (SDA şi SCL) sunt activate de circuite open-drain
  - Liniile sunt conectate la tensiunea pozitivă prin rezistoare de pull-up, astfel că în repaus sunt la nivel HIGH



Transfer de date pe I²C

- Când linia este liberă atât SDA cât şi SCL sunt la nivel SUS
- O tranzacție pe I²C începe cu SDA tras JOS, urmat de semnal pe SCL
- Dacă SCL este JOS, iar pe SDA se face o tranziție SUS-JOS urmează o transmisie ("START condition")
- Pentru fiecare bit de date transmis (SDA) valoarea logică trebuie să se păstreze neschimbată pe durata SCL în stare SUS
- Valoarea binară de date e eşantionată pe frontul crescător al SCL.
- Tranzacția se încheie cu SCL SUS şi o tranziție jos-sus pe SDA ("STOP condition")
- Doar un master poate genera semnal de ceas pe SCL

Transfer de date pe I²C

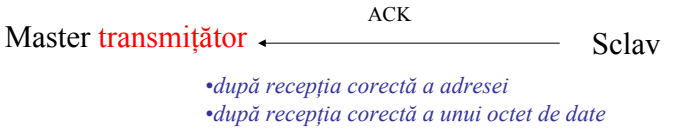


Aspecte funcționale

- Sclavul poate prelungii perioada JOS a ceasului (nu şi pe cea SUS) atât cât este necesar. Oricare receptor poate trage, dacă este nevoie, SCL jos (bit/eveniment de **WAIT**) pentru a întârzia un octet de date care urmează unui octet ce a sosit la aceeaşi adresă.
- Dacă două dispozitive încearcă să controleze aceeaşi linie (SDL sau SCL) circuitele cu colector în gol previn erorile fizice, iar fiecare din cei doi masteri trebuie să asculte magistrala în timpul transmisiei pentru a fi sigur că nu interferează cu alt mesaj
- După fiecare 8 biți transmişi pe SDL urmează un semnal (eveniment) de **ACKNOWLEDGE** (confirmare). Pentru a confirma recepția corectă, prin ACK, *receptorul trage SDL jos pe timpul celui de-al nouălea impuls de ceas*

Semnalizare de acknowledgement (bit ACK)

- Generat întotdeauna de receptor (master sau sclav)

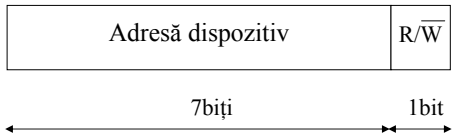


Nivelul legăturii de date

- Fiecărui dispozitiv I²C i se alocă o adresă. Adresele unice ale dispozitivelor sunt determinate de proiectantul sistemului, de obicei ca parte a programului driver pentru I²C.
- O adresă de dispozitiv are 7 biți conform standardului I²C (la I²C extins adresele pot fi de 10 biți)
- Adresa 0000000 este utilizată pentru a semnala un apel general ("general call") sau emisie pentru magistrală ("bus broadcast"), care poate fi folosită pentru a semnala / transmite simultan către toate dispozitivele.
- Adresele 11110XX sunt rezervate pentru schema extinsă cu adresă pe 10 biți; există de asemenea şi alte adrese rezervate.

Nivelul legăturii de date

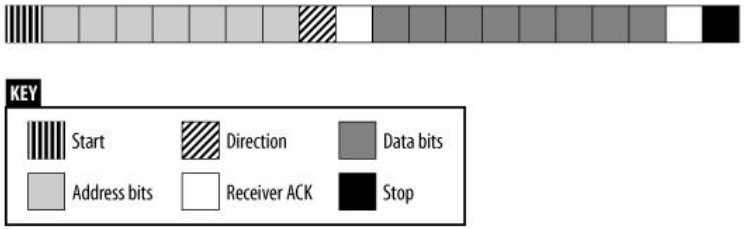
- O tranzacție de magistrală constă dintr-o serie de octeți transmişi: o adresă şi apoi unul sau mai mulți octeți de date.
- Format pentru transmisia adresei la I²C:



ARBITRARE I²C

- I²C este o magistrală multi-master
  ⇒ Mai multe dispozitive master pot porni transmisia în acelaşi timp !!
- Master transmite:
  - un bit 0 dacă trage SDA jos
  - un bit 1 dacă lasă linia SDA în stare sus
- Dacă doi masteri încep transmisia simultan, cel ce transmite 1 şi citeşte 0 pe magistrală va renunța pentru că a detectat o transmisie cu prioritate mai mare
- Protocolul de arbitrare e numit protocol cu bit dominant, sau ***Binary Countdown***

Un pachet I²C



Adrese speciale

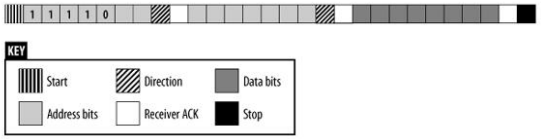
- Adresa 0000000 cu bit de direcție 0, este un apel general ("general call" sau "bus broadcast") care porneşte transmisia către toate dispozitivele I²C
- Prin acest apel general dispozitivul master determină ce sclavi sunt disponibili
  - Al doilea octet al transmisiei indică scopul apelului general
  - După acest al doilea octet sclavii află dacă comanda le este adresată şi răspund cu ACK
  - Sclavii pentru care comanda nu e aplicabilă neglijează apelul general
- Dacă al doilea octet este 0x06 (00000110), se indică că sclavii vor face un reset şi apoi vor răspunde cu adresa lor.
- Dacă al doilea octet este 0x04 (00000100), sclavii vor răspunde cu adresa lor, dar nu vor face şi reset
- Dacă cel mai puțin semnificativ bit la octetului doi este un 1, atunci, prin apelul general, masterul se identifică către alți masteri din sistem prin transmiterea adresei sale
- Ceilalți biți ai celui de-al doilea octet reprezintă adresa masterului

Adrese speciale

- Adresa 0x01 (00000001) este o adresă specială cunoscută ca şi octet de START
- Acest octet e utilizat pentru a indica că urmează un transfer lung de date
- Această adresă specială este importantă pentru dispozitivele master ce nu au interfață specializată I²C şi care prin software trebuie să testeze continuu magistrala
- Atunci când un asemenea master detectează un octet de START generat de alt master, el poate reduce rata de interogare, alocând mai mult timp pentru alte sarcini software

Adrese speciale

- Adresa 11110XX este adresa specială care indică că se va genera o adresă de dispozitiv pe 10 biți
- Se foloseşte în modul extins al I²C
- Teoretic se pot controla până la 1,024 periferice
- În acelaşi sistem pot exista dispozitive cu adrese pe 7 şi 10 biți
- La adrese pe 10 biți, adresarea se face prin doi octeți succesivi (ultimii 2 biți ai adresei speciale + 8 biți din al doilea octet)
- În acest caz perifericele cu adrese pe 7 biți vor ignora tranzacția



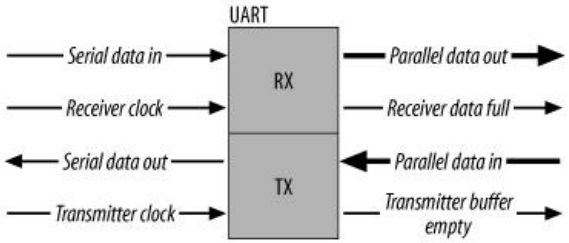
Transmițător / receptor asincorn universal (UART)

- Transferul serial de I/O se face pe un singur fir pentru fiecare direcție
- Toate interfețele transmițător seriale convertesc datele paralele într-un flux binar serial, iar receptoarele fac conversia inversă
- O asemenea comunicație serială se poate face de exemplu între:
  - calculator şi printer
  - calculator şi un control la distanță
  - calculator şi terminal pentru rețea Ethernet
  - .....
- La EmS interfața serială este metoda cea mai simplă şi ieftină pentru conectarea unui calculatorul gazdă, fie ca parte a aplicației, fie pentru scop de depanare
- Cea mai simplă interfață serială asincrponă este UART (the Universal Asynchronous Receiver Transmitter)
- UART este numită de asemenea ACIA (Asynchronous Communication Interface Adapter)

UART

- UART este asinconă pentru că nu se transmite semnal de ceas prin linia de date serială
- Receptorul recunoaşte valorile binare individuale fără o linie comună de ceas
- Interfața UART constă din două părți:
  - un receptor (receiver - Rx) care converteşte un flux serial de biți în date paralele (cuvinte) pentru microprocesor
  - un transmițător (transmitter - Tx) care converteşte date paralelă de la microprocesor într-un flux serial de biți, pentru transmisie
- UART furnizează şi informație de stare, cum ar fi "receptorul este plin (datele au ajuns) sau transmițătorul este gol (o transmisie în curs s-a terminat)

Diagramă funcțională UART



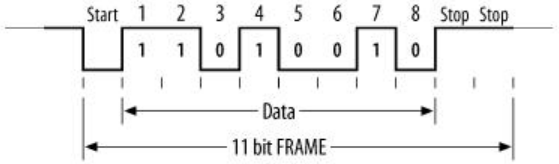
Transmisie serială

- O problemă importantă asociată cu transmisia serială este reconstrucția datelor la capătul de recepție
- Apar dificultăți în determinarea granițelor dintre biții individuali
  - De exemplu, dacă linia serială este jos pentru un anumit timp, receptorul trebuie să fie capabil să identifice dacă informația reprezintă "00" sau "000"
  - Trebuie să se știe unde se oprește un bit și unde începe celălalt
- Fiecare dispozitiv implicat în comunicația serială are propriul semnal de ceas local
- Dispozitivele emițător și receptor trebuie să lucreze la aceeași frecvență, iar în plus trebuie să existe logică pentru detectarea fazei datelor transmise și pentru sincronizarea ceasului receptorului la această fază
- Transmisia asincronă este utilizată în special la transmisa caracterelor și la sisteme unde intervalul dintre transmisa octeților se poate modifica în timp

Transmisie serială

- Formatul datelor transmise utilizează la începutul transmisiei pentru fiecare caracter un bit de start și la sfârșit unul sau doi biți de stop
- Receptorul își sincronizează ceasul după recepționarea bitului de start și apoi eșantionează biții de date (7 sau 8 în funcție de configurația sistemului)
- Dacă receptorul nu primește secvența de stop, presupune că ceasul său este defazat față de al emițătorului și se declară o eroare de cadru
- Rezolvarea erorii cade în seama aplicației software

Asynchronous serial data



Detecția erorilor de paritate

- Pentru a detecta erorilor, multe sisteme folosesc bitul de paritate, pentru verificarea datelor recepționate
- Bitul de paritate e calculat de transmițător și e inclus în cadrul de date
- Receptorul calculează paritatea pentru fiecare octet recepționat
  - Dacă bitul de paritate calculat este identic cu cel recepționat, receptorul presupune că nu sunt erori
- Pentru o transmisie corectă, emițătorul și receptorul trebuie să utilizeze același tip de paritate (even parity sau odd parity)
- Erorile de paritate trebuie tratate de programatorul sistemului
  - UART nu ia nici o măsură la detectarea erorii de paritate

RS-232C

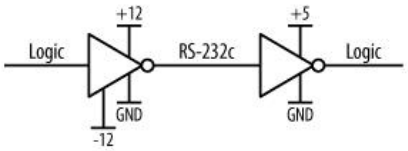
- RS-232C este un standard de interfață de comunicație serială apărut încă din anii ’60 ai secolului trecut
- RS-232C este utilizată pentru interfațarea dispozitivelor seriale printr-un cablu cu lungime maximă de 25 m şi o rată maximă de 38.4 kbps
- Se poate folosi pentru conectare la:
  - Alte calculatoare
  - Modem
  - Imprimante seriale
  - Plottere seriale
  - .....

RS-232C

- RS-232C foloseşte tensiuni referite față de masă (transmisie unipolară) :
  - Un logic high este un semnal cu tensiunea în domeniul -5 la -15 V (tipic -12 V)
  - Un logic low este un semnal cu tensiunea în domeniul +5 la +15 V (tipic+12 V)
- Terminologia utilizată pentru RS-232C datează de asemenea din anii 60.
  - Un semnal high /tensiune negativă (1 logic) era numit "space" iar low (0) era numit "mark“
- Unele sisteme RS-232C încă utilizează cadre de date pe 7 biți, iar altele 8 biți

RS-232C

- O legătură RS-232C constă dintr-un driver şi un comparator:



RS-232C

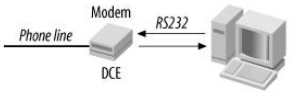
- RS-232C defineşte de asemenea conectorii şi alocarea pinilor
- RS-232C a fost concepută inițial pentru interconectarea a două tipuri de echipamente:
  - Data Terminal Equipment (DTE)
  - Data Communication Equipment (DCE)
- Înainte de apariția PC-urilor un DTE era un terminal sau un teletype, iar un DCE era un modem

RS-232C

conectare a unui terminal sau a unui modem la intzerfața serială a calculatorului

- Cum conectăm un terminal sau un modem la interfața serială a unui calculator ?
  - Calculatorul este DTE sau DCE?
- Standardul RS-232C spune că dacă un terminal este la un capăt al legăturii, la celălalt capăt este un DCE
- Dacă se conectează un terminal la o stație Unix workstation, standardul RS-232C spune că stația de lucru este un DCE

- Dacă se conectează un modem la un calculator, calculatorul va fi DTE



Conectări RS-232C

- Pentru a conecta un PC la un modem e nevoie de un cablu DTE-DCE
- Pentru a conecta un PC la un terminal e nevoie de un cablu DTE-DTE
- Pentru a conecta o stație Sun la un PC e nevoie de un cablu DCE-DTE null modem (unde Rx şi Tx sunt inversate)
- Pentru a conecta două PC-uri e nevoie de un cablu DTE-DTE null modem
- ……..
- Pentru doar două tipuri de dispozitive (DTE şi DCE) e nevoie de o sumedenie de cabluri cu permutări ale terminalelor

Conexiuni RS-232C

- Conexiuni standard pentru conectori cu 25 pini şi 9 pini. Numele semnalelor sun referite la DTE. De exemplu, Tx se referă la datele transmise de la DTE şi recepționate la DCE

Signal	Function	25-pin	9-pin	Direction
Tx	Transmitted Data	2	3	From DTE to DCE
Rx	Received Data	3	2	To DTE from DCE
RTS	Request To Send	4	7	From DTE to DCE
CTS	Clear To Send	5	8	To DTE from DCE
DTR	Data Terminal Ready	20	4	From DTE to DCE
DSR	Data Set Ready	6	6	To DTE from DCE
DCD	Data Carrier Detect	8	1	To DTE from DCE
RI	Ring Indicator	22	9	To DTE from DCE
FG	Frame Ground (chassis)	1	-	Common
SG	Signal Ground	7	5	Common

Shake Hands

- Atunci când se face transmisie serială, trebuie să existe o cale de a împiedica transmițătorul să transmită noi date înainte ca receptorul să fi avut şansa să termine prelucrarea datelor anterioare
- Protocolul este numit comunicare cu confirmare (handshaking), sau controlul fluxului de date (flow control)
- Conform acestui protocol după transmiterea unui octet (sau pachet de date), transmițătorul nu va mai trimite nimic până la confirmarea că receptorul este gata pentru o nouă transmisie
- Există trei forme de handshaking:
  - Hardware
  - Software
  - Fără confirmare

NO-HANDSHAKING

- Se utilizează atunci când transmisia este mult mai lentă decât procesul de recepție
  - De exemplu, dacă se foloseşte un sistem embedded simplu ce lucrează la 1 MHz şi care transmite date către un calculator de mare viteză ce lucrează la 4 GHz
  - Chiar dacă viteza de prelucrare a receptorului este foarte mare, dacă maşina rulează un sistem de operare obişnuit, pot exista situații când nu face față la fluxul de date de intrare
  - În aceste situații, o soluție practică bună este să se includă handshaking
- Dacă se foloseşte un port serial pentru a furniza o interfață a calculatorului cu un utilizator uman, este sigur că omul nu poate tipări suficient de repede ca să pună calculatorul în dificultate
- Ca urmare, dacă utilizați porturile seriale doar pentru accesul utilizator, sau pentru depanare, nu este nevoie de handshaking

Hardware handshaking

- Utilizează două semnale ale interfeței RS-232C:
  - RTS (Request To Send)
  - CTS (Clear To Send)
- Când transmițătorul doreşte să transmită, activează RTS, anunțând receptorul că sunt date de furnizat
- Receptorul activează CTS atunci când este gata să recepționeze noi date

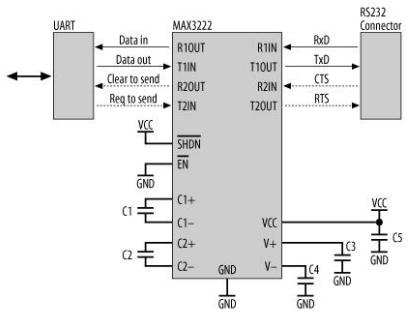
  ⇒ fluxul de date este limitat la viteza cu care acestea pot fi prelucrate.

Software handshaking

- Este cunoscut şi sub numele XON/XOFF
- Se utilizează doar la sistemele unde nu se poate implementa protocolul hardware, aşa cum se întâmplă la transmisia prin linie telefonică
- Software handshaking foloseşte două caractere pentru a indica o cerere de suspendare a transmisiei ("suspend transmission“) , respectiv un caracter de gata pentru reluare ("clear to resume”)
- Acestea sunt caracterele Ctrl-S (0x13) şi Ctrl-Q (0x11)
- Dacă se transmite doar text ASCII, totul va fi OK
- Dacă se transmit date binare, datele pot fi interpretate ca şi caractere de control
  - Soluția obişnuită constă în pre-procesarea datelor înainte de transmisie, prin conversie în caractere ASCII
  - De exemplu octetul 0x2F se transformă în caracterul ASCII "2" (0x32) plus "F" (0x46)
  - Programele de la receptor vor face conversia inversă în date binare

Implementarea unei interfețe RS-232C

- Multe MC incorporează cel puțin un UARTpe chip, astfel că este necesară doar logică externă pentru conversia la / de la nivelurile de tensiune RS-232C
- De exemplu se poate utiliza un transceiver MAX3222

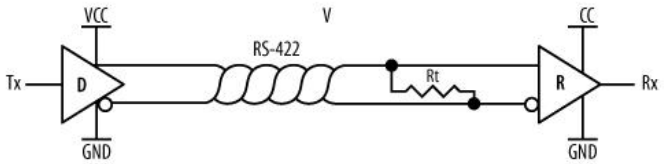


RS-422

- RS-422 este o interfață ce utilizează tensiune diferențială (între două linii) pentru a transmite datele binare
- Permite lucrul în mediu cu interferență, pentru că diferența de tensiune între firele torsadate rămâne aceeaşi
- RS-422 poate face transmisii de date pe cablu cu lungime de până la 1.200 m

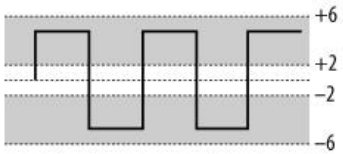
RS-422

- Legătura RS-422 conține un driver (D) şi un receptor (R) legate printr-o pereche de fire torsadate
- Rt – rezistență terminare, anti-reflexie, de obicei de 100-120 Ω.

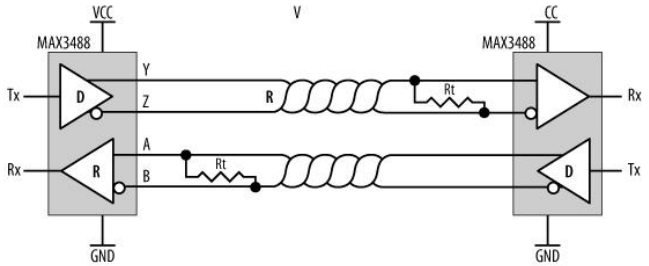


RS-422

- Diferențele de tensiune dintre cele două fire se încadrează în intervalul ±4 V şi ±12 V

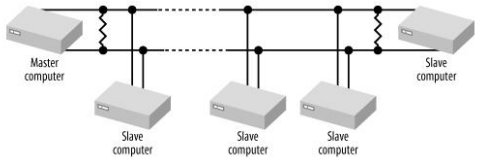


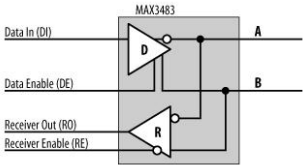
Interfață RS-422 bidirecțională



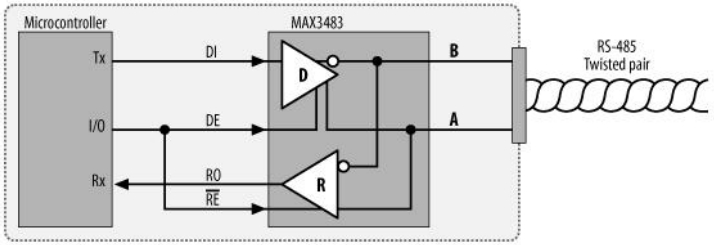
RS-485

- RS-485 este o variantă a RS-422 utilizată pentru legarea dispozitivelor într-o rețea de cost scăzut în aplicații industriale
- Permite mai multor noduri să schimbe informații prin două fire
- RS-485 este o arhitectură a master-slave
  - Toate tranzacțiile sunt inițiate de master, iar sclavul transmite doar atunci când i se cere acest lucru
- Rețeaua RS-485 foloseşte circuite transceiver, cum este Maxim MAX3483





Conectarea unui MAX3483 la un MC



PWM

- Este controllerul folosit de obicei pentru operații de conversie DA
- PWM = Pulse Width Modulation
  - De exemplu pentru un factor de umplere de 50% şi o amplitudine a formei de undă dreptunghiulare de 5 V, tensiunea medie pe ciclu este 2.5 V
  - De exemplu pentru un factor de umplere de 10% şi o amplitudine a formei de undă dreptunghiulare de 5 V, tensiunea medie pe ciclu este 0.5 V
- La ieşirea PWM se foloseşte în acest caz un FTJ
- PWM poate fi folosit pentru a controla viteza unui motor electric
- PWM poate fi utilizat pentru a comanda intensitatea luminoasă (reglabilă) printr-un LED
- PWM poate fi utilizat pentru generarea de tonuri audio