Komunikācijas interfeisi

Šie interfeisi var būt gan paralēlie, gan seriālie, gan sinhronie, gan asinhronie. Šie interfeisi var darboties pēc pilnās dupleksēšanas vai pus duplēksēšanas, tiem var būt devēja – izpildītāja režīms, vai arī abas iekārtas darboties vienādā nozīmē. Turpmāk apskatīta tikai komunikācija pa vadiem .

Seriālais interfeiss pārraida vienu datu bitu vienā laika posmā, tas ir efektīvs.

Paralēlais datu interfeiss, savukārt izmanto katram bitam savu datu līniju. Parasti komunikācijai kontrolieri izmanto 4 vai 8 bitu līnijas.

Sinhronajā komunikācijā svarīga ir abu ierīču takts frekvences sinhronizācija. Signāla sinhronizācija notiek, izmantojot atsevišķu signāla līniju, vai arī, izmantojot datu formātu, no kura otra ierīce pati veido takts frekvenci. Otrās metodes priekšrocība, ka uztverošajai ierīcei nav vajadzīga signāla ģenerācija, kas novērš sinhronizācijas kļūdas. Asinhronā komunikācijā abas ierīces reģistrē datu saņemšanu un tām jābūt vienādi konfigurētām kā arī jāizmanto iztveršana, lai saņemtu signālu. Komunikācijai nepieciešami arī start un stop biti, lai ierīce zina, kad ir ziņojuma sākums, beigas. Asinhronā komunikācija ir lēnāka, jo tajā ir vairāk kļūdu un zināms laiks vajadzīgs signāla iztveršanai.

Busu topoloģijā vienai līnijai var pieslēgt vairāk kā divas ierīces, kurām datu pārraidei izmanto ierīču adreses.

Parasti komunikācija starp kontrolieri un ierīci ir abos virzienos.

Pilnās dupleksēšanas komunikācijā abas ierīces var sūtīt un saņemt ziņojumu vienlaicīgi, kas parasti prasa divus vai vairāk vadus.

Pusdupleksēšanai priekšrocība ir vadu ekonomija, taču šim savienojumam parasti viens uztver un otrs saņem.

Devēja un izpildītāja sistēmās parasti devējs inicializē savienojumu, bet izpildītāji gaida devēja atļauju piekļūt komunikāciju videi.

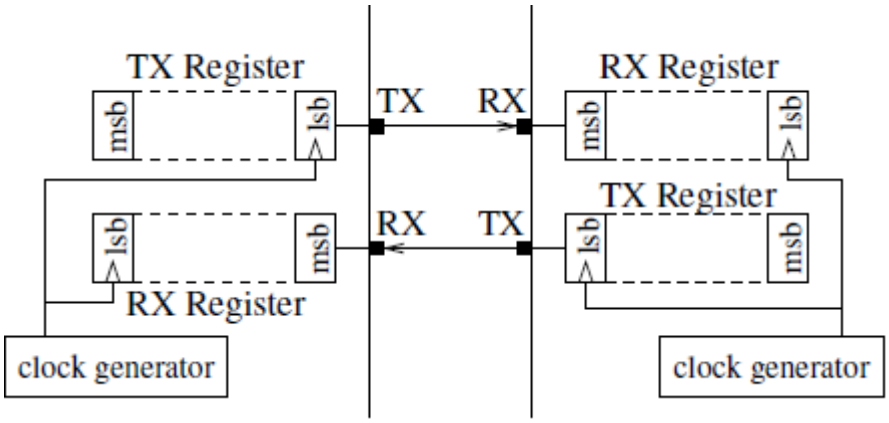
Sistēmām, kurās ir vienādas nozīmes lietotāji, nepieciešami papildu nosacījumi, lai organizētu tīklu.

Fiziskajā līmenī abas sistēmas var būt viena virziena vai diferenciālas, kur viena virziena ir ar kopīgu zemējumu, kas grūti, ja attālums starp ierīcēm ir liels.

Diferenciālie interfeisi izmanto divus vadus datu pārsūtīšanai. Diferenciālie interfeisi izmanto voltāžas izmaiņas datu pārsūtei, turklāt tos mazāk ietekmē traucējumi un tie atļauj ilgāku datu pārraidi.

3.1 SCI (UART)

Seriālais komunikācijas interfeiss (SCI) nodrošina asinhronas komunikācijas interfeisu (universālais asinhronais raiduztvērējs, UART). UART modulis izmanto divus ceļus, pārvades (TXD) un saņemšanas (RXD) līniju, lai veidotu pilnu vai daļēju divvirziena komunikāciju. 3.1 attēls parāda UART iekšējo struktūru. Būtībā modulis sastāv no pārraides un saņemšanas reģistriem, lai noturētu datus. Lai nodrošinātu pareizus asinhronos veidus, mezgla pārvadi un saņemšanu darbina ar tā vietējā pulksteņa ģeneratoru.



3.1 attēls: UART moduļa pamata struktūra

UART nav sakaru protokols ar sevi, bet ir modulis, kurš var tikt izmantots asinhroni seriāliem sakariem. Līdz ar to UART modulis robežojoties ar mikrokontrolleru ļauj kontrolēt lielu daļu no paša uzvedības. Konfigurējamie parametri iekļauj:

Datu bitu skaitu: atkarībā no UART, datu bitu skaits var tikt izvēlēts vairāk vai mazāk plašā apgabala robežās. Piemēram, ATmega sērijas atļauj starp 5 un 9 data bitiem. Citiem UARTiem var būt plašāki vai mazāki attālumi.

Paritātes bits: lietotājs var izvēlēties, vai būtu vajadzīgs paritātes bits, un ja jā, vai paritātei nepieciešams būt pāra vai nepāra. Ja paritāte ir uzstādīta kā pāra, tad paritātes bits ir 0, ja vienīnieku skaits starp datu bitiem ir pāra. Nepāra paritāte ir pretēja.

Apstāšanās bits: lietotājs parasti var izvēlēties vai nepieciešams viens vai divi apstāšanās bits.

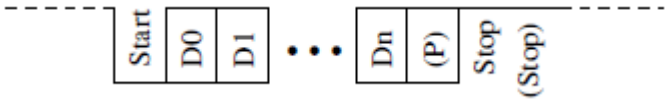
Boda ātrums (*Datu pārraides ātruma mērvienība, kas atbilst signāla diskrēto stāvokļu vai stāvokļu izmaiņu skaitam vienā sekundē. Bods nosaukts franču zinātnieka Ž.M.Boda vārdā.*): UART modulis satur reģistru, kas lietotājam atļauj izvēlēties konkrētu boda ātrumu (t.i., pārraides ātrumu bitu skaitam sekundē (bps)) no pieļaujamā. Iespējamie boda ātrumi parasti iekļaujas 9600 un 115200 boda robežās. Tomēr, tā kā iespējamie boda ātrumi ir atkarīgi no sistēmas takts frekvences, atšķirīgi takts ātrumi nozīmē dažādus pieejamos veidus boda ātrumos.

Nosaukumus izmanto, lai aprakstītu datu formātu, kas ir D{E|O|N}S, kur D ir datu bitu skaits un S ir apstāšanās bitu skaits. E|O|N norāda pāri, nepāri vai vispār neesošu paritāti. Piemēram, a

datu formāts ar 8 datu bitiem, norāda paritāti, un viens apstāšanās bits tiek atzīmēts kā 8E1. Ņemiet vērā, ka nav nepieciešams norādīt starta bitu skaitu, jo tas vienmēr ir viens.

Datu pārraide

Ziņojums tiek nosūtīts, izmantojot kodējumu *Nekādu Atgriešanos uz Nulli* (NRZ)[3], kas ir, piemēram, 1 atbilst pozitīvākam spriegumam un 0 atbilst negatīvākam (pozitīvā-loģika) vai otrādāk (negatīvā-loģika). Kamēr pārraide ir asinhrona, datus jāaplokšņo (*Ziņojuma daļa, kas satur informāciju, ko izmanto ziņojuma nogādei adresātam.*) ar uzbūvi, kas satur vismaz vienu starta bitu un vienu apstāšanās bitu. 3.2 attēls parāda vispārēju UART paketes uzbūves formātu.



3.2: UART uzbūves formāts.

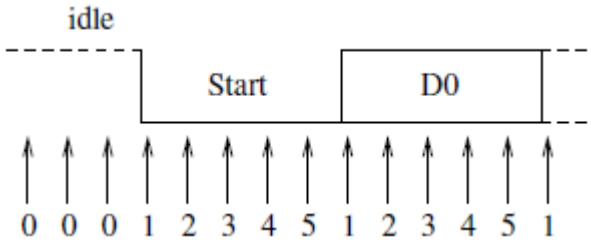
Dīkstāvē līnija ir augsta. Uzbūve sākas ar starta bitu, kas izraisa līnijai kļūt zemai. Šī krītošā fronte signalizē uztvērējam, ka ir sākusies jauna pārvade. Pēc starta bita, tiek pievienoti datu biti, sākot ar visnenozīmīgāko bitu. Datu bitu skaits ir maināms un nepieciešams uzstādīt tādā pašā vērtībā gan pārsūtītājā, gan uztvērējā. Pēc datu bitiem drīkst sekot paritātes bits. Uzbūve ir noslēgta ar vienu vai diviem apstāšanās bitiem. Apstāšanās biti atbilst augstam stāvoklim.

Sinhronizēšana un kļūdu atpazīšana

Kamēr pārraide ir asinhrona, sūtītāja un saņēmēja taktis ir pilnīgi neatkarīgas viena no otras. Ar boda ātruma uzstādīšanu, uztvērējs zina kādu bita ātrumu gaidīt, bet tas nezina, kad bits sāksies un līdz ar to nepieciešams sinhronizēt starta bitu uz krītošu fronti. Turklāt takts oscilators tendēts uz ne-nulles ievirzi, tas ir, tie atšķiras no to nominālās frekvences. Līdz ar to pat tad, ja saņēmējs sinhronizē sūtītāja taktij sākuma ziņojumu, tas var novirzīties pārvades laikā.

Lai sūtītājs iegūtu sākotnējās sinhronizācijas, saņēmējs izmanto iztveršanu, tas ir, RDX līnija iztver s reizes uz katru bitu. Tipisks s skaits ir 16. Kad uztvērējs notver krītošo fronti, tas pieņem, ka tas ir starta bita sākums un sāk skaitīt nolases, kā parādīts 3.3 attēlā.

[3] Nosaukums ir atvasināts no Atgriezt uz Nulli kodējumu, kur sprieguma līmenis atgriežas „nulles" stāvoklī katrā bita otrajā pusē.



3.3 attēls: UART bitu nolasīšana (s=5).

Ideālā gadījumā visiem s starta bitu iztvērumiem nepieciešams būt nullēm. Tomēr, lai būtu noturīgāki pret trokšņiem, UART izmanto dažus iztvērumus, lai noteiktu bita vērtību. Piemēram, ATmega16 s=16 un tā izmanto 8, 9, 10 iztvērumus un izmanto balsu vairākumu, lai noteiktu līnijas stāvokli. Ja divi vai vairāki iztvērumi ir augsti, starta bits tiek atmests uz līnijas, pretējā gadījumā tas tiek atzīts kā pārraides sākums.

Visi turpmākie pakešu biti atkal tiek iztverti s reizes un ar tādu pašu metodi kāda tika izmantota sākuma bitam, tiek noteiktas bitu vērtības. Datu biti tiek ievietoti saņemtajā pārbīdes reģistrā un parasti iekopēti bufera reģistra saņemšanas beigās. Bufera reģistrs atbrīvo pārbīdes reģistru priekš nākamā uztvertā datu pāra, ja vēl nav tikts nolasīts pēdējais dats. Pārvade tiek noslēgta ar apstāšanās bitu (-iem).

Var rasties dažas kļūdas: pirmā no visām, var iegūt bitu kļūdas trokšņu dēļ uz līnijām. Ja pakete ietver paritātes bitu, viena bita kļūdu (vai vairākas nepāra skaita bitu kļūdās) var konstatēt un paziņot to lietotājam. Joprojām datu bitus iekopē bufera reģistrā, bet paritātes kļūdas bits tiek uzstādīts UART stāvokļa reģistrā, lai norādītu, ka ir bijusi paritātes pārkāpums.

Otrkārt, var gadīties, ka sūtītāja un saņēmēja boda ātrumi ir pārāk atšķirīgi, līdz ar to saņēmējs pakāpeniski zaudē sinhronizāciju pārraides laikā. Tas var tikt atpazīts apstāšanās bitā, kur UART gaida augsta stāvokļa nolasījumu. Ja apstāšanās bits netiek atpazīts, formāta kļūda paziņo to darbībai. Bet, protams, var būt situācijas, piemēram, kad saņemtā takts ir lēna, kur līnijas dīkstāve ir kļūdaini novērtēta kā apstāšanās bits un kļūda netiek atpazīta.

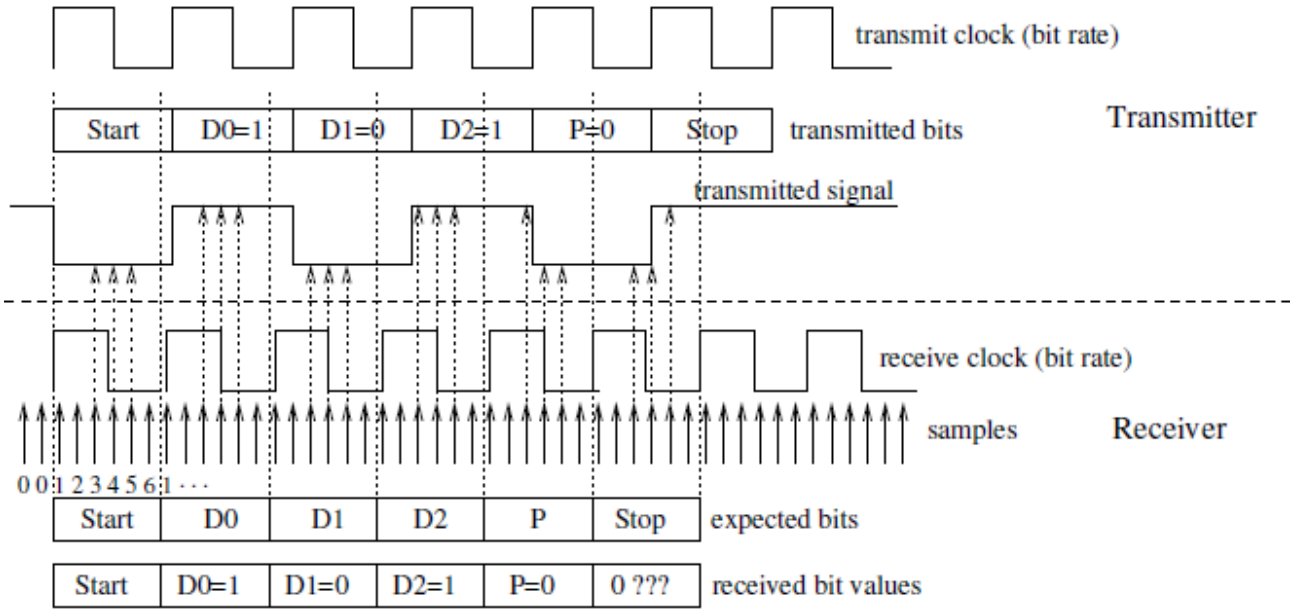
Visbeidzot, kaut arī UART parasti izmanto bufera reģistru, lai iegūtu vairāk laika nolasot ienākošās datus, tādējādi otrā pārvade var sākties pirms dati no pirmās ir apstrādāti. Var gadīties arī tā, ka trešā pārvade sākas pirms dati no pirmā ziņojuma vēl tikai tiek nolasīti. Šādā gadījumā notiek datu pārpilde un pārbīdes reģistrs pazaudē datus. Šādu datu pārpildi norāda ar karodziņa palīdzību UART stāvokļa reģistrā.

Boda pārraides ātruma ģenerēšana

Boda pārraides ātrums tiek iegūts no sistēmas skaitītāja takts veidā. Boda ātruma reģistrs, kas būtībā strādā ar tādu pašu funkciju kā izvades salīdzināšanas reģistrs, kas tika aprakstīts 2.6.3 sadaļā, tiek izmantots, lai izveidotu periodisku takts signālu. Šis signāls pēc tam tiek samazināts (par s) uz vēlamo boda ātrumu, izmantojot prescaler. Uztvērējā tiek izmantots tas pats

mehānisms, bet takts signāli tiek ņemti jau pirms prescaler. Tādējādi uztvērējs iztver s reizes ātrāk nekā bitu pārraides ātrums. Kamēr iztvērēja pārraides ātrums tiek ģenerēts no uztvērēja sistēmas takts, tikai tie signāli, kuriem bitu pārraides ātrums ir <=1/ s-to saņēmēja takts pārraides ātrumu var apstrādāt.

Kā blakusefekts ģenerējot boda pārraides ātrumu no sistēmas takts, iestatītie boda pārraides ātrumi tiek piedāvāti kā kontrolleri atkarīgi no to takts frekvences. Turklāt, ne katrs patvaļīgais boda pārraides ātrums tiek sasniegts precīzi. Piemēram, ja ir 8 MHz takts un izmanto iztveršanu, kur s=8, varam iegūt boda pārraides ātrumu precīzi 0.5 Mbps. No otras puses, 115.2 kbps boda pārraides ātrums nevar tikt ģenerēts ar kontrollera sistēmas takti tikmēr, kamēr neesam samazinājuši maksimāli sasniedzami boda pārraides ātrumu līdz (8 MHz/s)/115.2 kHz=8.68. Varam tikai iestatīt skaitītāja pretējās vērtības, tādējādi jebkuram boda pārraides ātrumam, kas satur daļskaitli, nevar precīzi veikt ģenerēšanu. 3.4 attēls parāda sekas, pārvadot lēnāku nekā paredzēts uztvērēja boda pārraides ātrumu.



3.4 attēls Boda pārvades ātrums ir pārāk lēns, uztvērējs pazaudē sinhronizāciju (3E1, s=6).

Piemērs izmanto s=6, ar 3E1 uzbūves formātu. 3, 4 un 5 iztvērumi tiek izmantoti izlemšanai. Augšējā daļa parāda raidītāja skatījumu, apakšējā daļa – uztvērēja. Kā varat redzēt, pats sinhronizējas sākuma bitam un pēc tam pakāpeniski zaudē sinhronizāciju, jo tas ir ātrāks nekā pārvadītā takts. Tomēr datu bitus joprojām var atzīt kā pareizus, pat ja iztvērumi nav pārvadīto bitu centrā, bet ar katru bitu pārvietojas tuvāk kreisajai malai. Kamēr iztvēruma algoritms izmanto balsošanas algoritmu, lai noteiktu bita vērtību, pat ja paritātes bits ir joprojām atzīts kā korekts, pāra parauga pirmais iztvērums nolasa datu bita D2 vērtību. Tikai apstāšanās bitā uztvērējs saprot, ka kaut kas ir nepareizi.

Ja sūtītājs būtu bijis nedaudz ātrāks, vai mēs izmantotu 2E1 datu formātu, uztvērējs atpazītu apstāšanās bitu (lai gan iespējams ne tieši tā, kā paritātes bits mūsu piemērā). Tātad acīmredzami nav vajadzīga precīza sūtītāja un saņēmēja takts sakrišana, tik ilgi, kamēr paturam

frekvences pietiekamā tuvumā. Ir arī skaidrs, ka garākai uzbūvei frekvencēm jābūt tuvāk kopā, jo uztvērējam jābūt pietiekami tuvu apstāšanās bitam, lai to pareizi noteiktu.

Ja mēs atgriežamies pie mūsu piemēra, kur B=115.2 kbps ar f=8 MHz takti un s=8, mēs varam iestatīt mūsu skaitītāju uz C=9, jo tas ir tuvāks skaitlim 8.68, kuru prasījām.

$$B' = \frac{f}{s \cdot C}$$

Rezultējošais boda pārraides ātrums atšķiras no vēlamā boda pārraides ātruma B par

$$(\frac{B'}{B} - 1) \cdot 100\%,$$

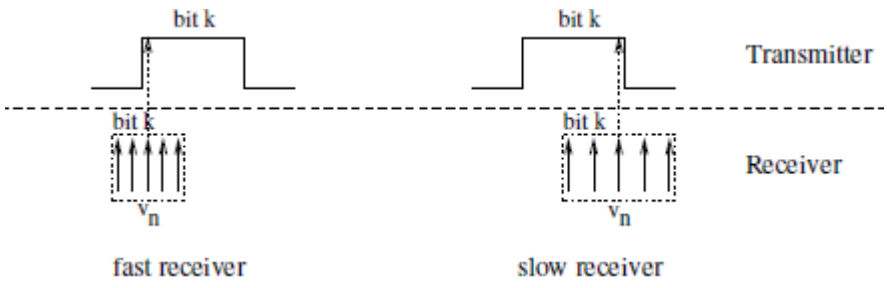
Līdz ar to šajā gadījumā tiks iegūta -3.5% kļūda. Lai noteiktu vai šis ir pieņemams mūsu freima formātam, nepieciešams aprēķināt cik daudzas reizes skaitītāja takts atšķirsies no pārvades laikā izsūtītās takts. Ja lielākā daļa iztvērumu ņemti apstāšanās bitā priekš viena bita krituma, pārraide būs notikusi veiksmīgi. Ja lielākā daļa iztvērumu ir izvadē, sinhronizācija tiks zaudēta.

Pieņemsim, ka sūtītājs darbojas ar iztvēruma frekvencēm f_t (tātad bitu pārraides ātrums ir f_t/s) un skaitītājs iztver ar frekvenci f_r. No s iztvērumiem s_0,..., s_{s-1}, kas tiek paņemti ar saņēmēju, iztvērumus $s_{v_1}, \ldots, s_{v_{2n+1}}, 0 \leq v_1 < \ldots < v_{2n+1} \leq s-1$ izmanto balsošanas algoritmā. Kā varam redzēt 3.4 attēlā, ja iztvērums ir paņemts no diviem dažādiem bitiem, tad arī vidējais iztvērums s_{vn} paņems no pareizā bita un balsošanā tiks izlemta pareizā vērtība, vai arī s_{vn} paņems no nepareizā bita, kura gadījumā balsošana cietīs neveiksmi. Tātad, lai uztvērējs atpazītu freima k-to bitu, k>=0, pareizi, iztvērumam s_{vn} jābūt no k bita. Lai to nodrošinātu, jāizpilda šādi nosacījumi:

$$\frac{k \cdot s}{f_t} < k \cdot \frac{s}{f_r} + \frac{v_n}{f_r} + o < \frac{(k+1) \cdot s}{f_t}, \tag{3.3}$$

Kur $o \in (0, 1/f_r)$ ir pirmā iztvēruma nobīde no starta bita krītošās frontes.

3.5 attēls ilustrē formulu. Ja mēs pieņemam, ka pārvades sākumā t=0, tad k starta bits nosūtītājā ir $t_k^s = k \cdot s/f_t$. Uztvērējā starta bita krītošā fronte ir atpazīstama ar nobīdi $o \in (0, 1/f_r)$. K bita sākums tiek sagaidīts $t_k^r = k \cdot s/f_r + o$, un iztvērums s_{vn} tiek paņemts laikā $t_k^r + v_n \cdot 1/f_r$. Ja šo iztvērumu nepieciešams pārraidīt, k bits (formula 3.3) nekavējoties sekos.



3.5 attēls: Freima k-tā bita iztveršana

Formula 3.3 ir pielietojama daudzām lietām. Piemēram, ja mēs esam piešķīruši freima garumu k bitiem, mēs varam izskaitļot relācijas stāvokļus no f_t līdz f_r. Formula 3.3 sniedz

$$\frac{k \cdot s + v_n}{f_r} > \frac{k \cdot s}{f_t} \Leftrightarrow \frac{f_r}{f_t} < 1 + \frac{v_n}{k \cdot s} \quad \text{and}$$
$$\frac{k \cdot s + v_n + 1}{f_r} < \frac{(k+1) \cdot s}{f_t} \Leftrightarrow \frac{f_r}{f_t} > \frac{k}{k+1} + \frac{v_n + 1}{(k+1) \cdot s},$$

kas noved pie attiecīgā f_r/f_t nosacījuma

$$\frac{k}{k+1} + \frac{v_n + 1}{(k+1) \cdot s} < \frac{f_r}{f_t} < 1 + \frac{v_n}{k \cdot s} \tag{3.4}$$

Ņemiet vērā, ka pie $k \to \infty$, iegūsim

$$\lim_{k \to \infty} \left(\frac{k}{k+1} + \frac{v_n + 1}{(k+1) \cdot s} \right) = 1 = \lim_{k \to \infty} \left(1 + \frac{v_n}{k \cdot s} \right) \tag{3.5}$$

Un tādējādi $\lim_{k \to \infty} f_r / f_t = 1$, tas ir, nosūtītajām un saņemtajām frekvencēm jābūt vienādām.

USART

Universālais sinhronais asinhronais raiduztvērējs (USART) paplašina UART funkcionalitāti ar sinhrono pārraides moduli. Tādēļ USART ir papildus trešā līnija, kas ietver takts signālu. Sinhronā režīmā, takts signāls tiek ģenerēts ar vienu no komunikācijas partneriem un to izmanto gan datu pārraidei, gan uztveršanai. Protams, šī sinhronā komunikācija asinhronajā modulī cauriztveršanas mehānismu pilnīgi nevajadzīgu, līdz ar to sinhronais režīms ir ar s faktoru ātrāks par asinhrono režīmu. USART modulis kombinē apvieno loģiku abu veidu komunikācijās. Ja tiek izmantots asinhronā komunikācija, takts līnija ir brīva un parasti tiek izmantota kā normāls I/O pins.