Серийни асинхронни интерфейси

Автор: гл. ас. д-р инж. Любомир Богданов

Европейски съюз

ПРОЕКТ BG051PO001--4.3.04-0042
„Организационна и технологична инфраструктура за учене през целия живот и развитие на компетенции”
Проектът се осъществява с финансовата подкрепа на
Оперативна програма „Развитие на човешките ресурси”,
съфинансирана от Европейския социален фонд на Европейския съюз
Инвестира във вашето бъдеще!

ЕСФ България
Европейски социален фонд

Съдържание

Преобразуване на паралелна информация в серийна

Паралелните интерфейси позволяват големи скорости на обмен на информация, но имат един съществен недостатък – **изискват голям брой проводници**.

Серийните интерфейси позволяват този брой да намалее значително, с което намалява цената на преносната среда, както и вероятността за “преслушване” (cross talk) между отделните линии.

Преобразуването на паралелната информация в последователна става с помощта на **преместващи регистри** (shift register)**.**

Преобразуване на паралелна информация в серийна

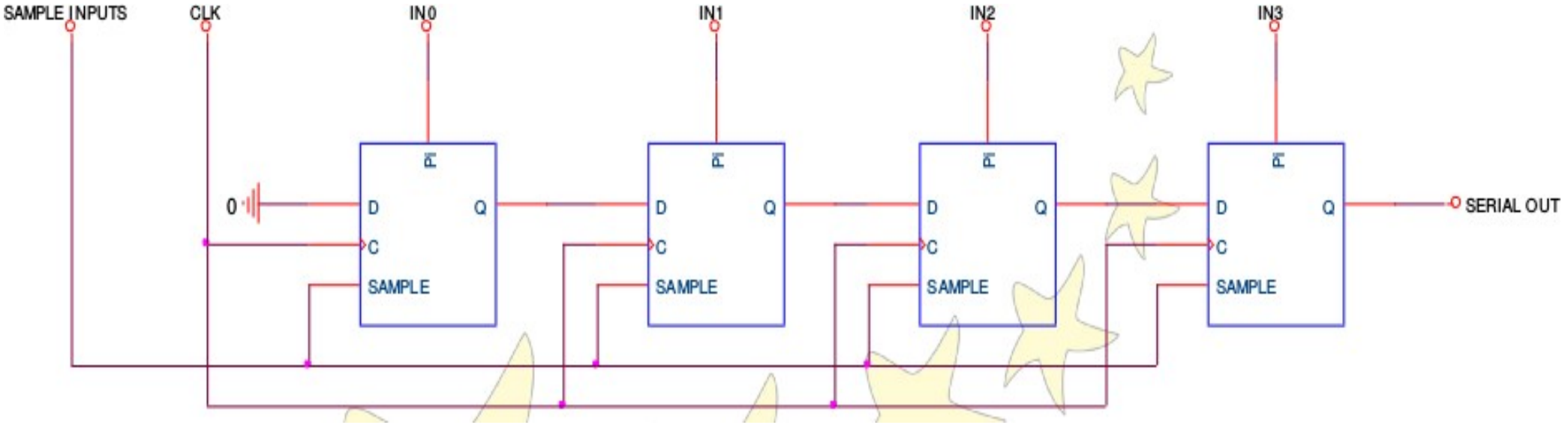
Преместващите регистри са изградени на базата на **последователно свързани тригери** [1].

Информацията подадена на входа на преместващия регистър се прехвърля от един тригер в друг под влиянието на тактов сигнал.

На следващия слайд е представено преобразуването на 4-бита информация в паралелен и сериен вид с помощта на преместващи регистри, реализирани с D-тригери.

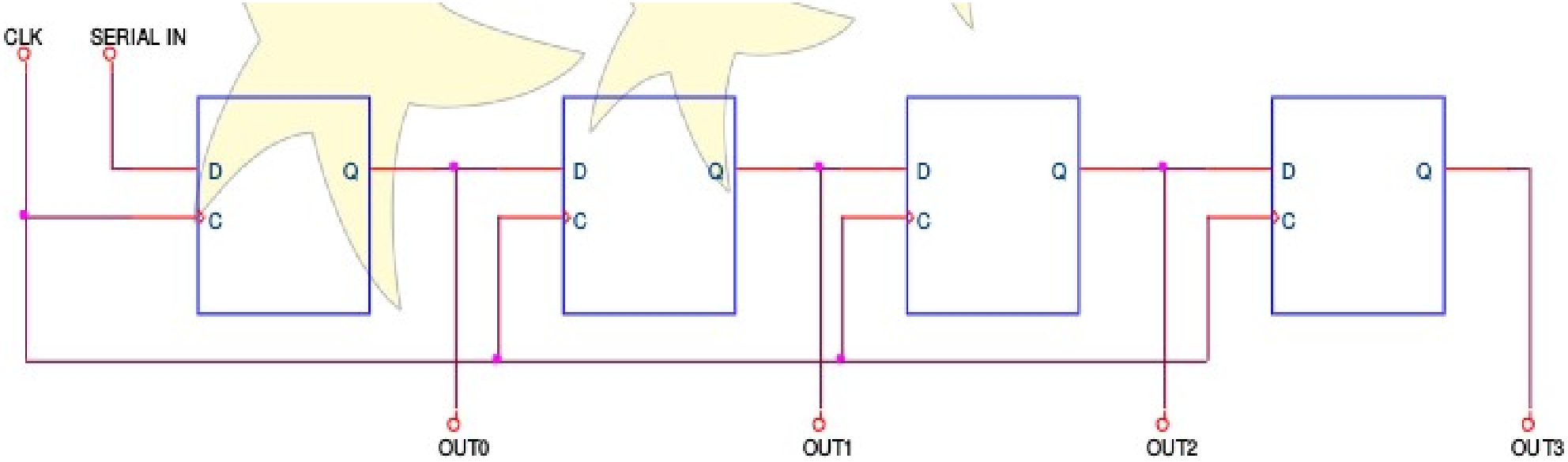
С помощта на ИС 4014 и 74HC595 може да се преобразуват 8-битови данни в сериен и паралелен вид.

Преобразуване на паралелна информация в серийна



Преобразуване **паралелна → серийна** информация.

Преобразуване на паралелна информация в серийна



Преобразуване **серийна → паралелна** информация.

Преобразуване на паралелна информация в серийна

Интерфейсите се разделят на **асинхронни** и **синхронни** в зависимост от това дали използват синхронизиращ тактов сигнал или не.

Асинхронни са тези интерфейси, които не използват тактов сигнал за предаването на данните. При тях се разчита, че главното и подчиненото устройство притежават стабилен източник на тактова честота и **скоростта на обмен на данни е предварително известна**. Така, когато едното устройство започне да предава, другото знае за колко време се предава един бит информация и само се синхронизира спрямо първото.

При тези интерфейси **стартовото условие** служи като репер за начало на обмена.

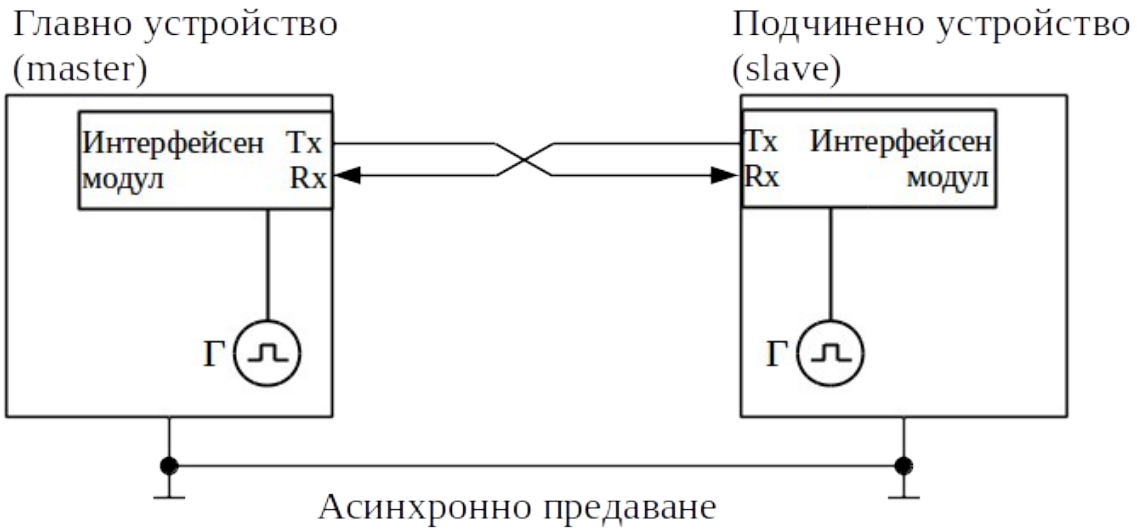
Преобразуване на паралелна информация в серийна

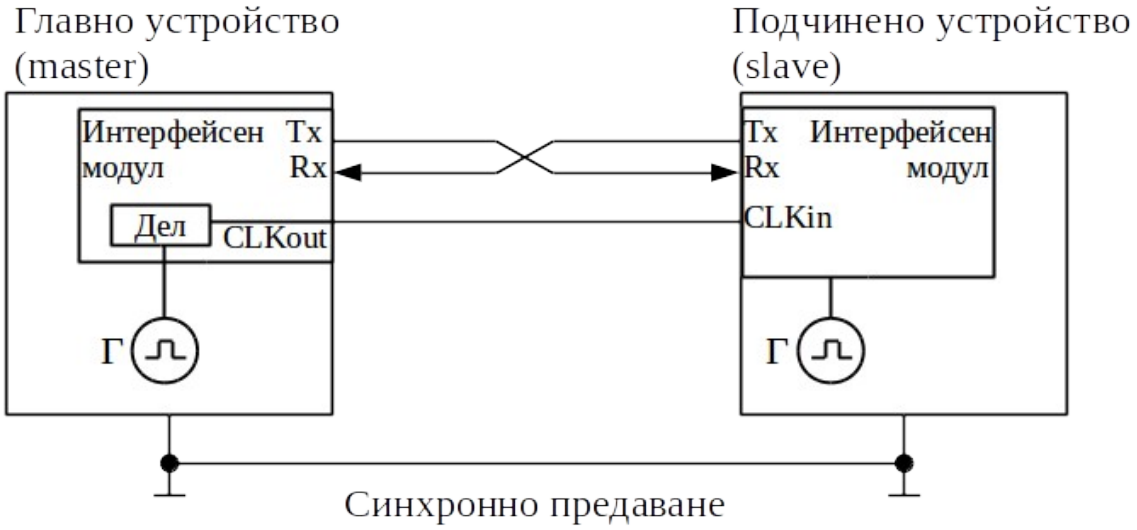
Синхронни са тези интерфейси, които използват тактов сигнал за предаване на данните. При тях едното устройство генерира сигнал на отделен проводник, по който се синхронизира предаването на отделните битове.

Преобразуване на паралелна информация в серийна

На фигурата са демонстрирани двата варианта за предаване на информация.

При синхронното предаване обикновено тактът е по-нискочестотен от системния, затова на фигурата е включен делител.





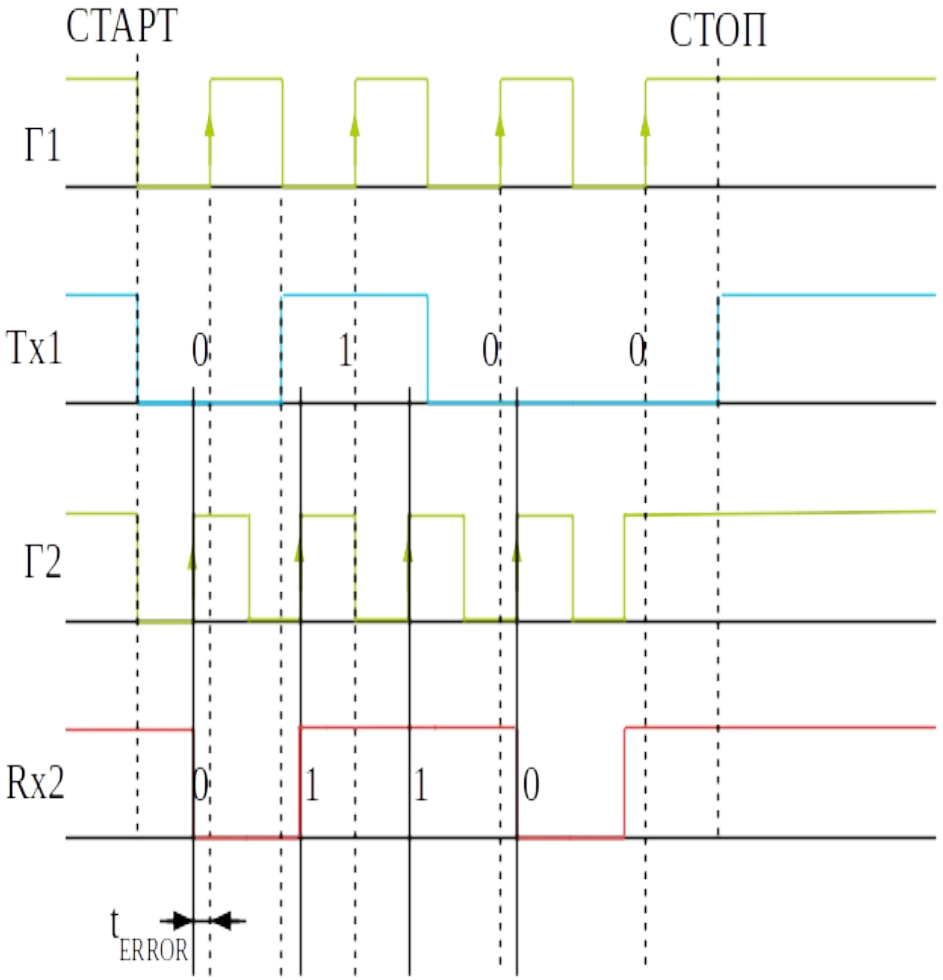
Преобразуване на паралелна информация в серийна

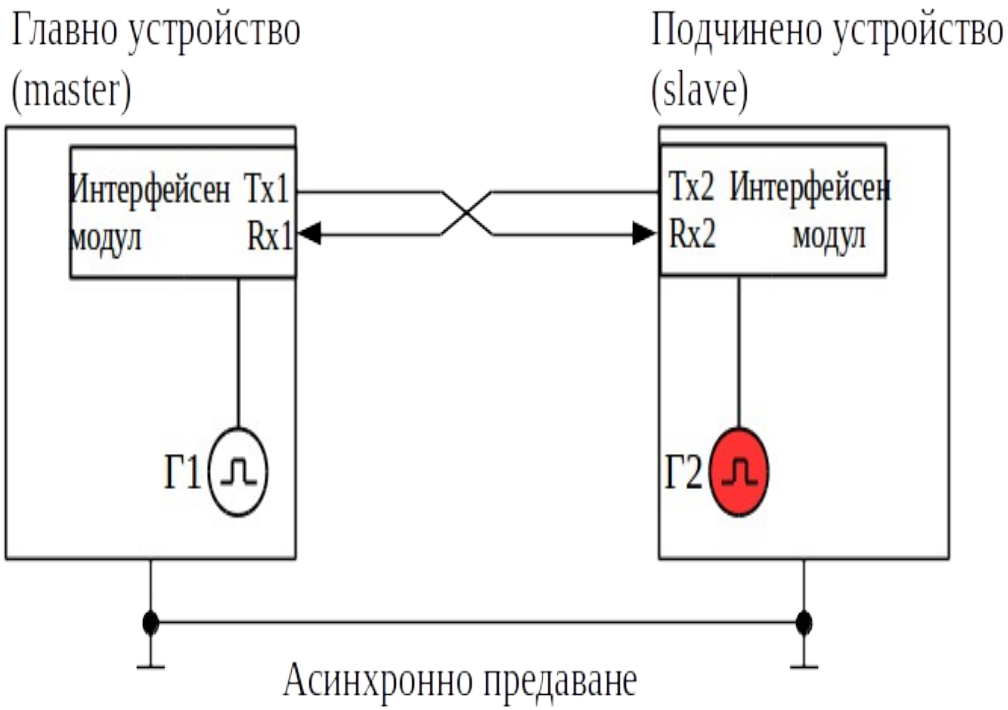
При асинхронното предаване на данни **стабилността на вътрешният генератор** на комуникиращите устройства е от изключителна важност за правилното предаване на информацията.

На фигурите на следващия слайд е показано предаването на данни по нарастващ фронт на вътрешния такт на двете устройства. Вижда се, че ако на приемащото устройство честотата е малко по-висока (например), то подчиненото устройство ще приеме грешни данни.

Обикновено при разсинхронизиране, ако честотата е грешна, но не се изменя в рамките на предаваната дума, първите няколко бита се приемат правилно, а последните няколко – грешно.

Преобразуване на паралелна информация в серийна





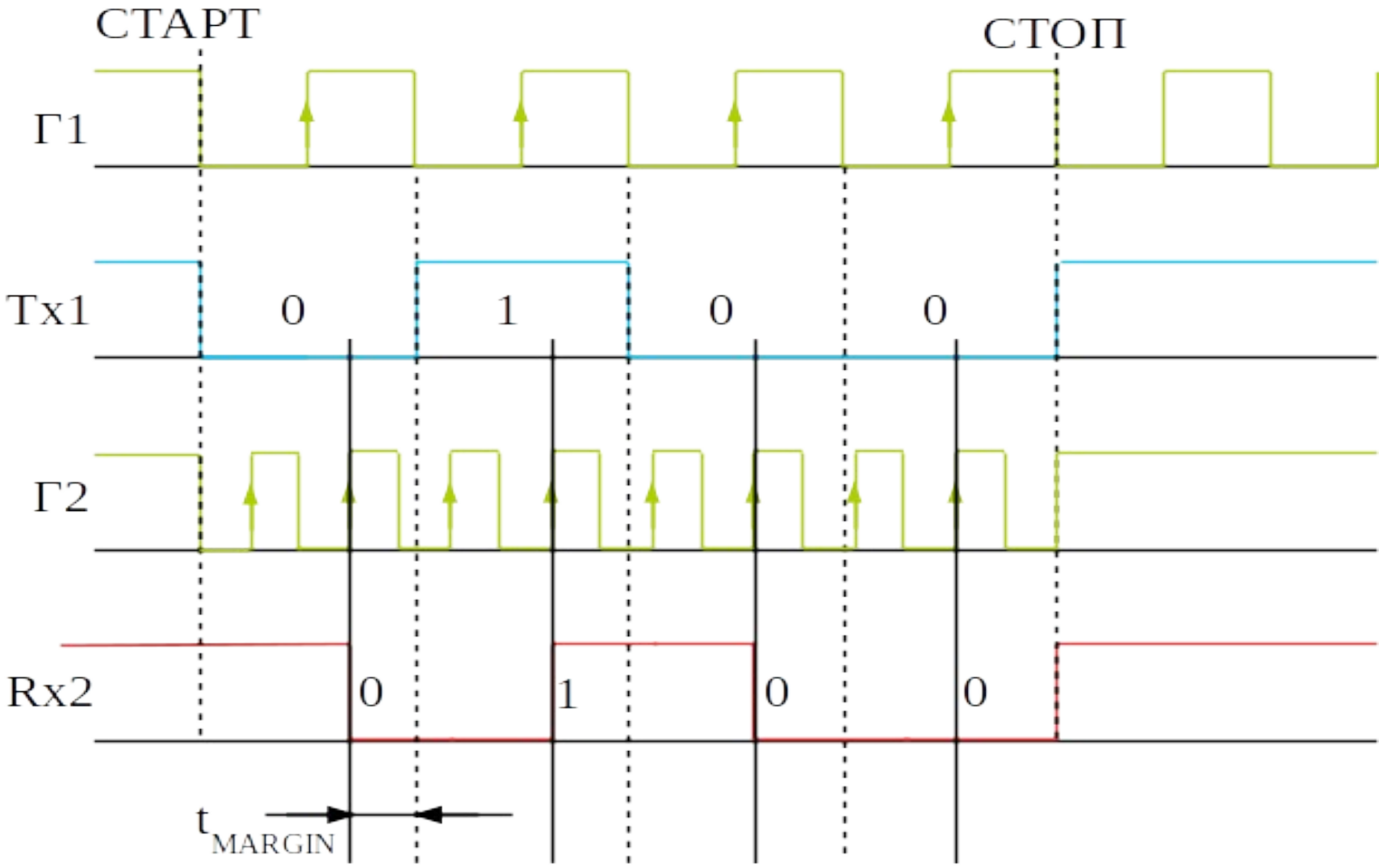
Преобразуване на паралелна информация в серийна

За да се реши проблемът с честотната нестабилност, обикновено асинхронните интерфейсни модули използват т.нар. **наддискретизация** (oversampling).

При метода с наддискретизация се използва **по-висока честота** от честотата на прехвърляните данни. Всеки бит се чете в средата на периода на данните.

Типична честотата на наддискретизация може да е x8 или x16 по-висока. Колкото по-висока е тя, толкова по-малък е шансът за възникване на грешки, но и толкова повече се увеличава консумацията на модула.

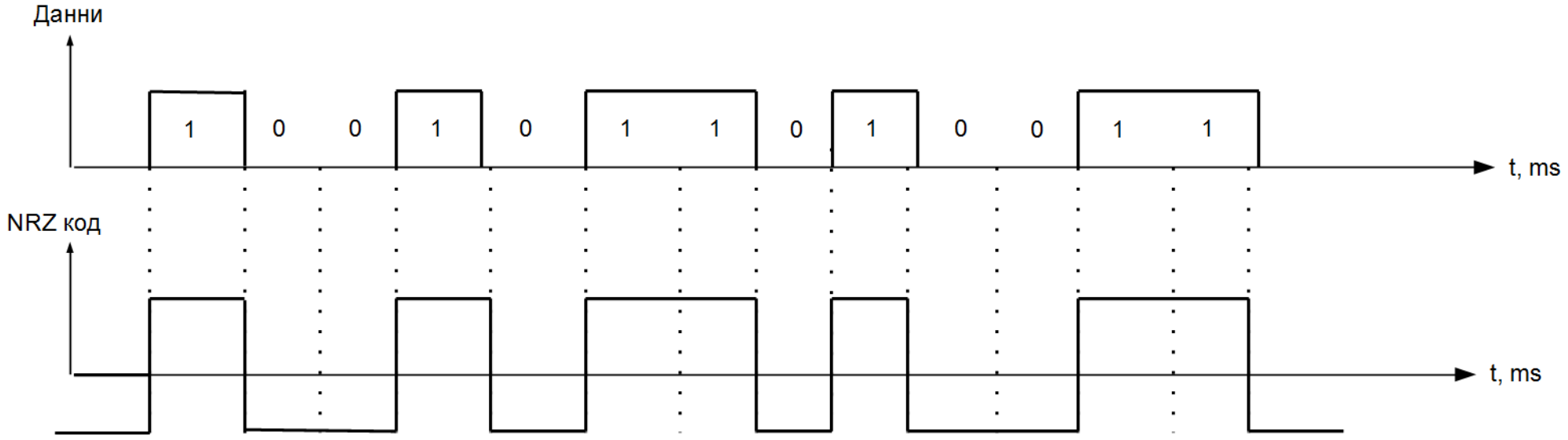
Преобразуване на паралелна информация в серийна



Асинхронно предаване с наддискретизация x2

Интерфейс RS232

NRZ (non-return to zero) кодиране е специално представяне на данните, при което логическата единица е представена с едно отличително състояние (например положително напрежение), а логическата нула – с друго такова (например отрицателно напрежение). Характерното за този код е, че по интерфейса няма вариант, в които сигнала да се връща в нулево положение (напрежение 0 V). На фигурата по-долу е показан пример за такова кодиране:



Интерфейс RS232

RS232 (IEEE Recommended Standard 232) е асинхронен, напрежителен, сериен интерфейс.

Използва се за осъществяване на връзка между **2 устройства**:
*микроконтролер и персонален компютър
*микроконтролер и друга система/модул.

Предаването е **пълен дуплекс** (full duplex), т.е. едновременно може да се предават и приемат данни.

Интерфейс RS232

Куплунгът на RS232 е стандартизиран и съответстващите му сигнали са показани на фигурата на следващия слайд.

Съществуват два варианта на куплунга:

***25-изводен**

***9-изводен**

Във вградените системи най-често се използва 9-изводния вариант и често само 3 проводника от него – RxD, TxD и GND.

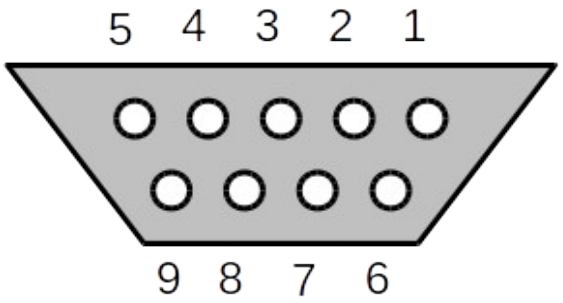
Сигналите на интерфейса се отбелязват с RxD и TxD. Тяхното предназначение е

***RxD** се използва за приемане на данни

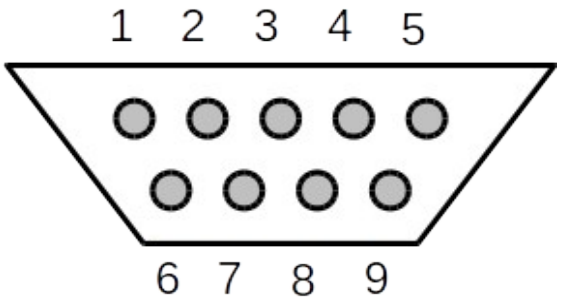
***TxD** се използва за предаване на данни

При свързване на два куплунга RxD и TxD трябва да се разменят както е показано на фигурата.

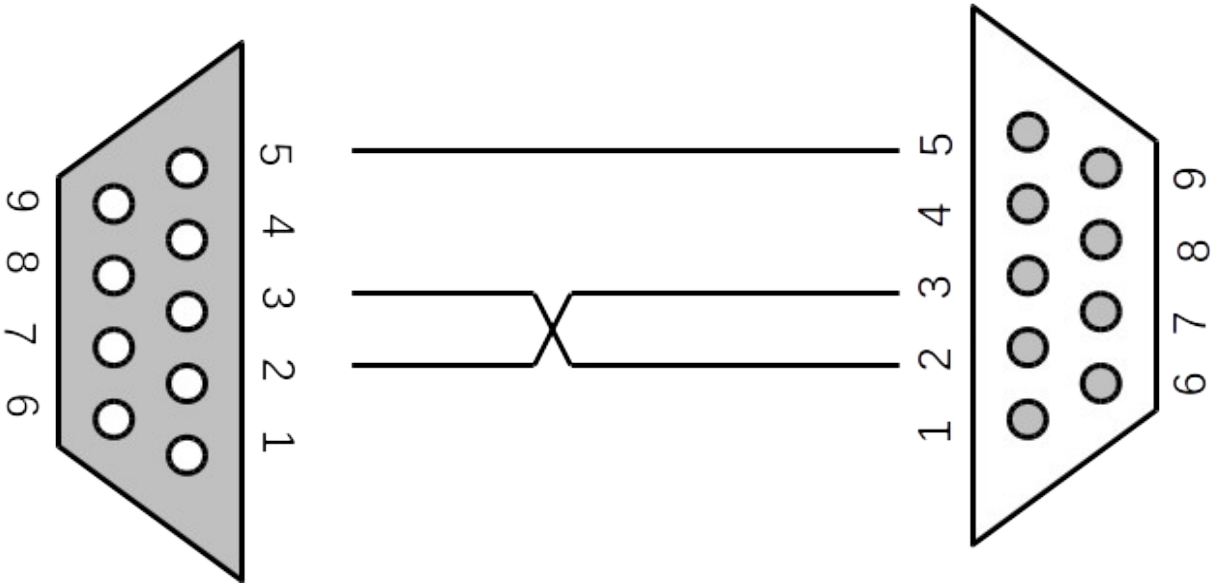
Интерфейс RS232



Женски куплунг DB-9
1 – DCD 5 – GND 9 – RI
2 – RXD 6 – DSR
3 – TXD 7 – RTS
4 – DTR 8 – CTS



Мъжки куплунг DB-9
1 – DCD 5 – GND 9 – RI
2 – RXD 6 – DSR
3 – TXD 7 – RTS
4 – DTR 8 – CTS



Интерфейс RS232

DTE (**D**ata **T**erminal **E**quipment) – главно устройство, използва мъжки куплунг DB-9. Персоналният компютър се счита за DTE.

DCE (**D**ata **C**ircuit-terminating **E**quipment) – подчинено устройство, използващо женски куплунг DB-9. За DCE може да се приеме например вградена система или измервателен уред, свързан към компютъра.

Интерфейс RS232

Някои GSM или Wi-Fi модули, които използват този интерфейс, включват и сигналите за хардуерно съгласуване (hardware flow control) [2]:

*RTS (**R**equest **T**o **S**end) - главното устройство DTE е готово да приема данни
*CTS (**C**lear **T**o **S**end) – подчиненото устройство DCE е готово да приема данни

*DTR (**D**ata **T**erminal **R**eady) – главното устройство DTE е инициализирано и готово за работа (след включване на захранването)
*DSR (**D**ata **S**et **R**eady) – подчиненото устройство DCE е инициализирано и готово за работа (след включване на захранването)

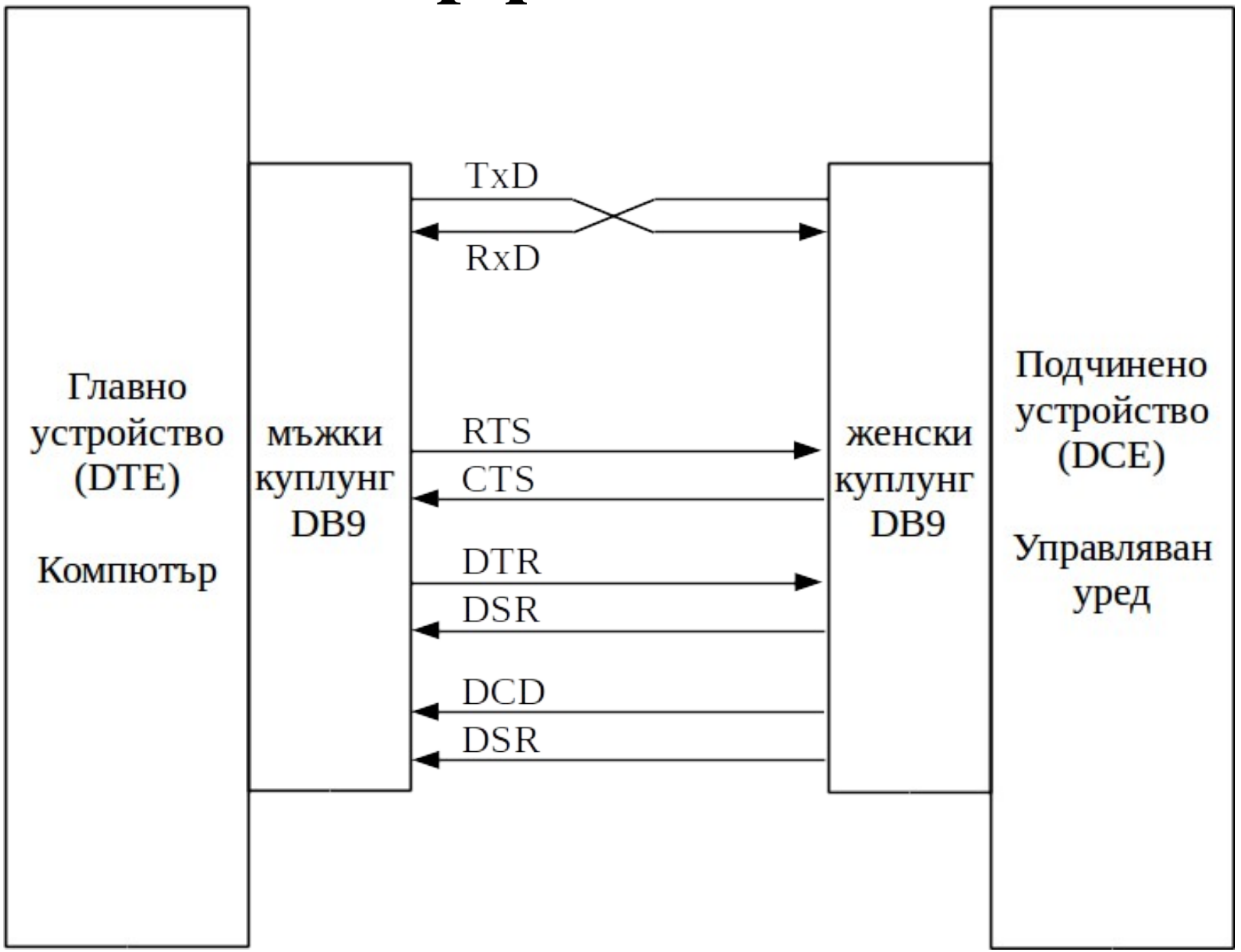
Интерфейс RS232

*DCD (**D**ata **C**arrier **D**etect) – изход на подчиненото устройство DCE, който предизвиква хардуерно прекъсване в главното DTE. Използва се от програмата на DCE като индикация за осъществена връзка с отдалечено устройство.

*RI (**R**ing **I**ndicator) – изход на подчиненото устройство DCE, който предизвиква хардуерно прекъсване в главното DTE. Използва се от програмата на DCE като индикация за позвъняване от телефон.

Исторически – интернет по телефонната линия минаваше през модем, свързан към RS232 на компютъра. При заспиване на компютъра той можеше да бъде събуден чрез телефонно позвъняване и да се свърже към интернет автоматично. Това е "предшественика" на wake-on-LAN функцията в модерните компютри.

Интерфейс RS232



Интерфейс RS232

!!!ВНИМАНИЕ!!! Названията “главно” и “подчинено” устройство се използват само за улеснение. Понеже линията е пълен дуплекс, и DTE, и DCE могат да стартират обмен на данни.

Интерфейс RS232

Логическите нива се представят с отрицателни и положителни напрежения за разлика от TTL нивата, където са само положителни.

Логическата 0 (наричана Space) се представя с напрежения ≥ +5 V.

Логическата 1 (наричана Mark) се представя с ≤ -5 V.

Максималните нива на напреженията трябва да са в границите ± 15 V, но чиповете, използващи този интерфейс трябва да са способни да издържат до ±25 V (ако случайно се насложат смущения към полезния сигнал).

Приемниците трябва да регистрират и затихнали сигнали с амплитуда от ± 3 V [2].

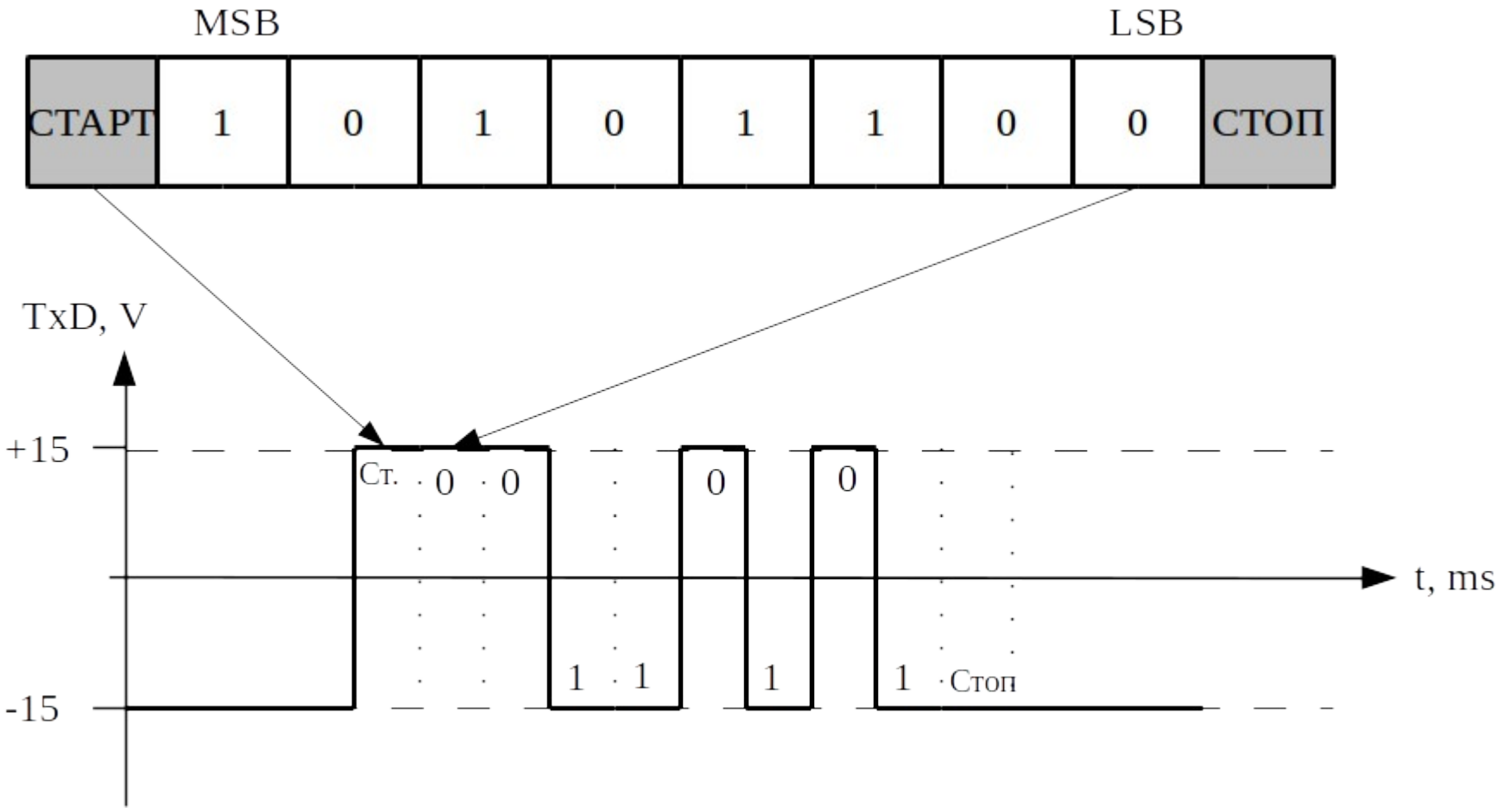
Когато **не се предават данни** TxD се държи в **логическа 1**.

Интерфейс RS232

Повечето инженери са свикнали да представят байтовете с най-старшия бит отляво, а най-младшия – отдясно. При RS232, обаче, **изпращането** на данните **започва с най-младшия бит**.

При наблюдение на осцилоскоп данните от RS232 изглеждат обърнати. Например числото 10101100 ще се види като 00110101.

Интерфейс RS232



Интерфейс RS232

Скоростите на предаване са **стандартизирани** и се дават в бодове (bauds). При този интерфейс 1 бод предава 1 бит информация (докато при QPSK модулацията, например, 1 бод предава 4 бита информация).

Някои от по-често използваните скорости са – 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bauds/sec. По-високи скорости на обмен се осъществяват с USB-UART емулатори.

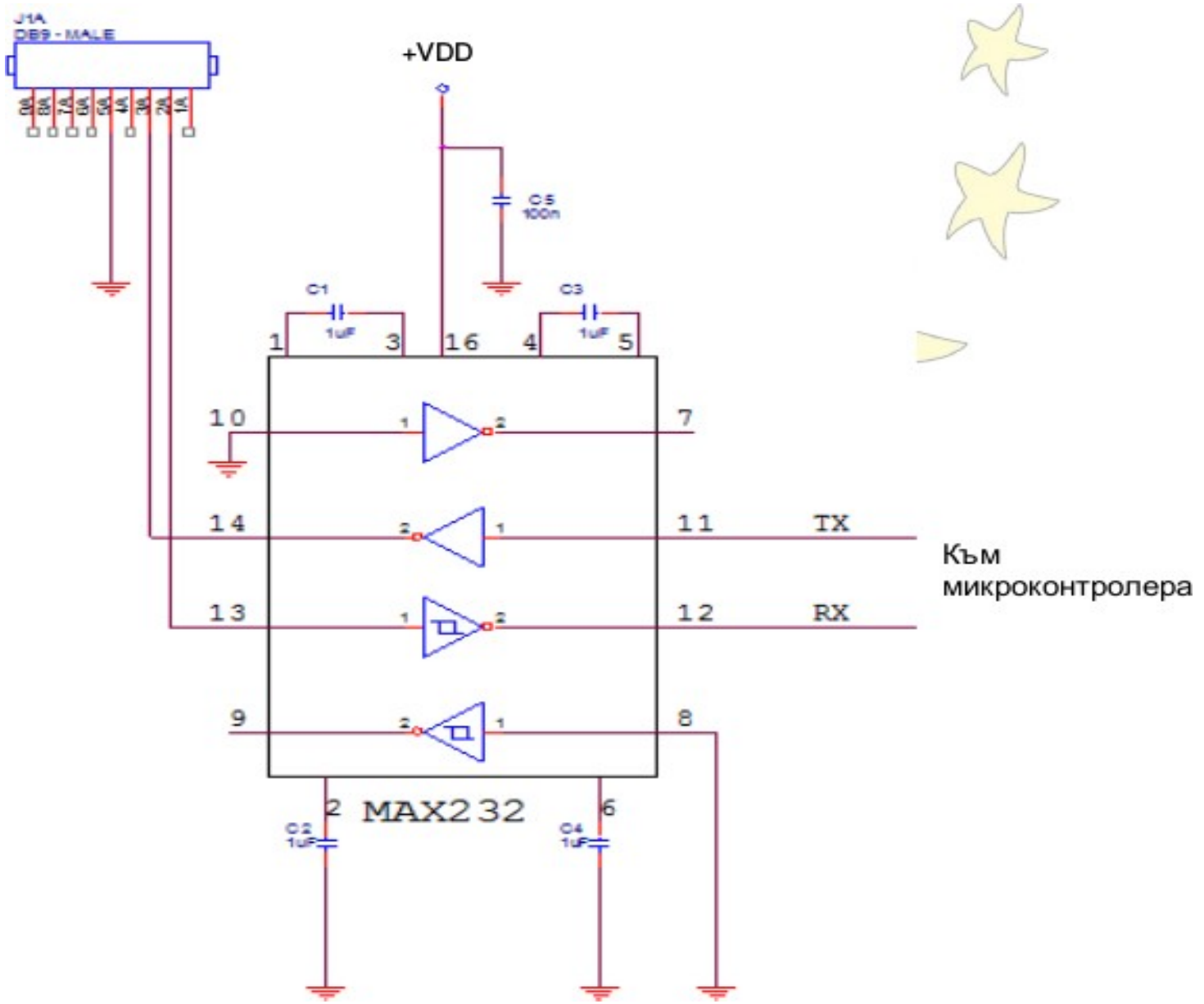
Максималната дължина на кабела между две устройства е **10 m**.

Интерфейс RS232

Микроконтролерите се захранват с напрежения до 5 V и не могат да изработят двуполярни сигнали с амплитуда ≥ ±5 V. Също така не биха могли да приемат такива сигнали без да се повредят. Затова се използват специални **транслаторни схеми,** които изработват необходимите нива на сигналите и за интерфейса, и за микроконтролера.

Една примерна такава ИС е показана на следващия слайд. Това е MAX232, а на схемата е дадена и вътрешната му структура. Инверторите са специални и те осигуряват транслацията на нивата. Използват се **капацитивни постояннотокови преобразуватели** (charge-pump) за повишаване и изработване на отрицателни напрежения, като за целта се свързват кондензаторите C1, C2, C3 и C4.

Интерфейс RS232



Интерфейс RS232

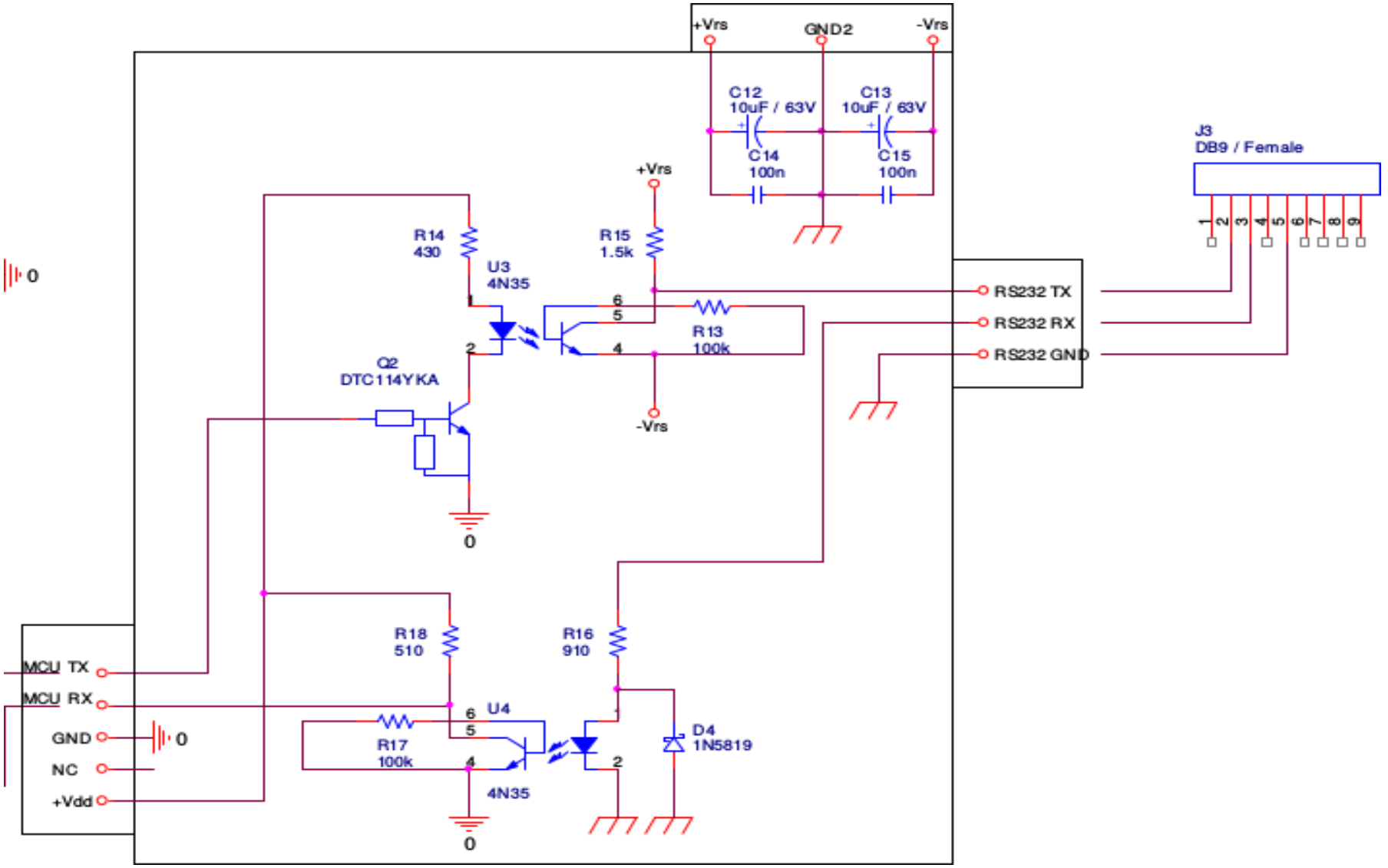
На следващият слайд е показана схема с **галванично разделяне**. Използват се два оптрона. Обърнете внимание, че има два символа за маса. Те не са свързани никъде в схемата.

При галванично разделяне трябва да се осигури **отделно захранване** на схемата с оптроните. В случая този проблем е решен с отделен маломощен трансформатор T4. U1 и U2 захранват аналогово-цифровата част на уреда. D5 и D6 са ценерови диоди, осигуряващи захранването на оптроните.

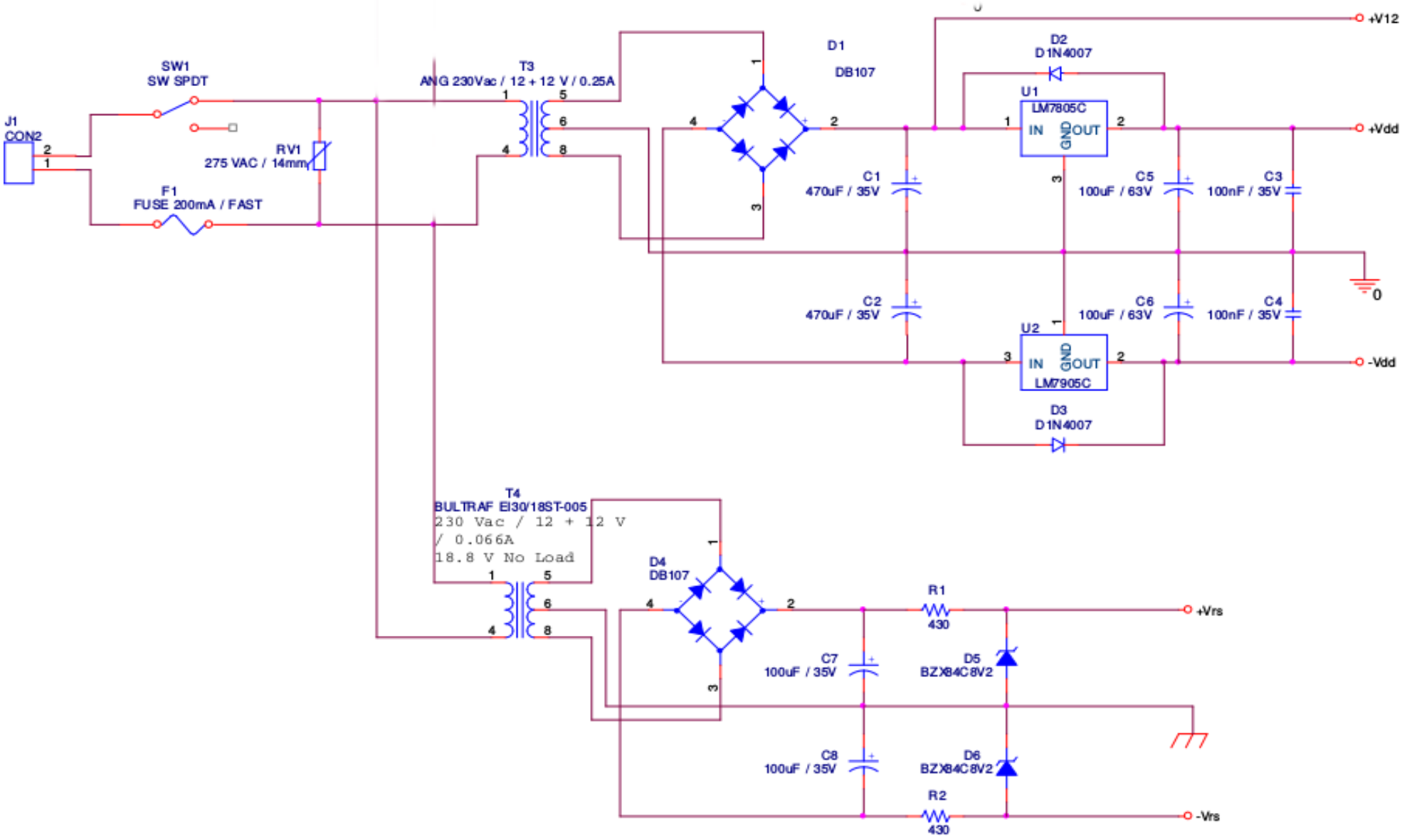
D4 предпазва фотодиода от отрицателните напрежения, когато се предава логическа 1.

R13 и R17 са слаби, издърпващи резистори. Фототранзисторът ще работи и без тях.

Интерфейс RS232



Интерфейс RS232



UART модул

Микроконтролерните модули, които отговарят за осъществяване на RS232 комуникацията чрез специален протокол се наричат UART (**U**niversal **A**synchronous **R**eceiver/**T**ransmitter). Среща се още и с класическото съкращение SCI (**S**erial **C**ommunication **I**nterface).

MSP430-базираните микроконтролери използват съкращението USART (**U**niversal **S**ynchronous/**A**synchronous **R**eceiver/**T**ransmitter), защото един и същ модул може да се използва и за синхронни интерфейси (например SPI, I 2 C).

UART модул

Предаването на данни се осъществява дума по дума, като всяка една съдържа:

*един **стартов бит** (винаги 0)

***даннови битове** на думата (5 ÷ 8)

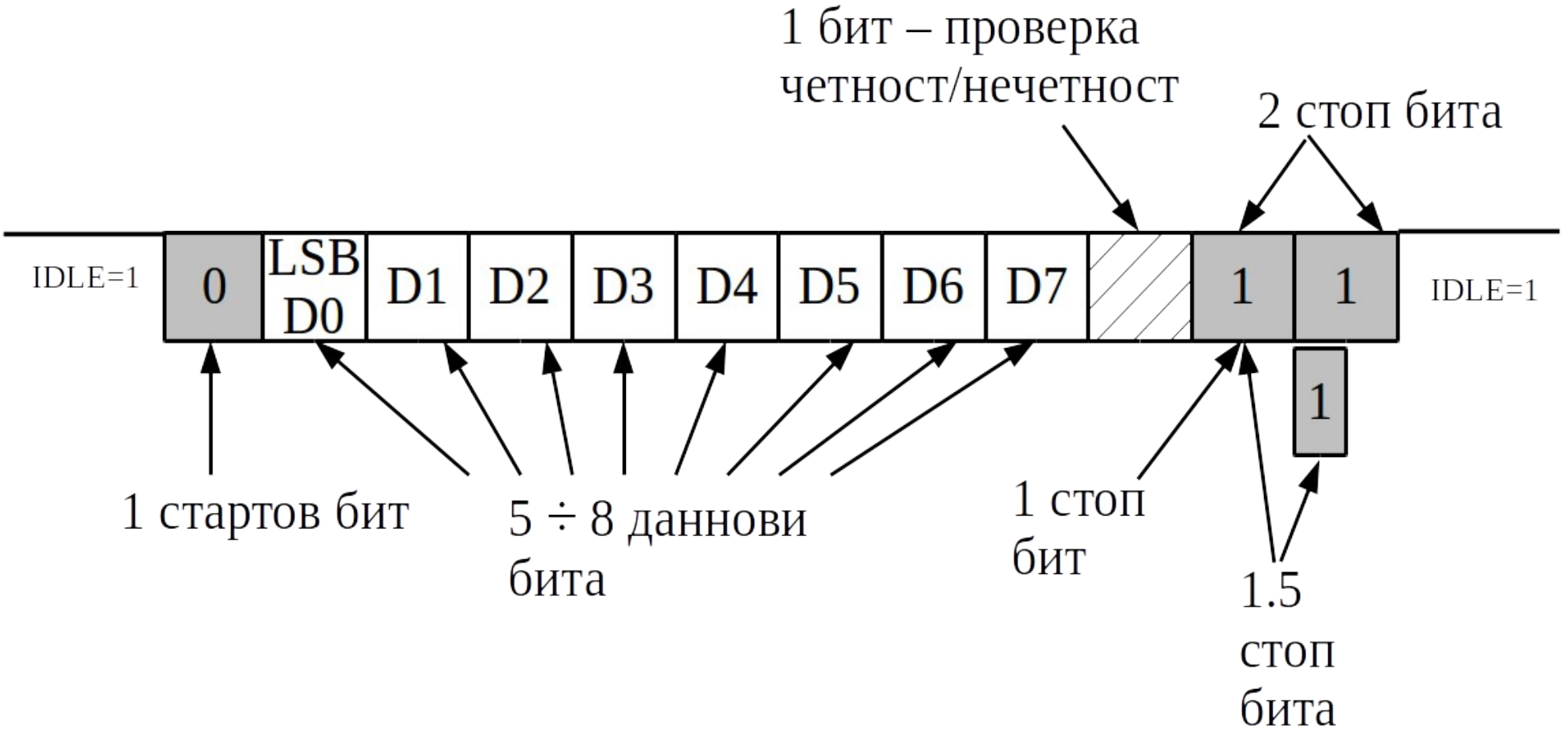
***бит проверка** за грешки по четност/нечетност (незадължителен)

*един, един и половина, или два **стопови бита** (винаги 1).

Изпращането на данните се извършва от LSB към MSB.

Когато не се изпращат данни (IDLE), TxD линията на всяко устройство е в логическа 1.

UART модул


1 бит – проверка
четност/нечетност
2 стоп бита
IDLE=1
0
LSB
D0
D1
D2
D3
D4
D5
D6
D7
1
1
IDLE=1
1
1 стартов бит
5 ÷ 8 даннови
бита
1 стоп
бит
1.5
стоп
бита

UART модул

***Стартовият бит** се реализира с логическа 0 и указва началото на предаването на потребителските данни.

***Стоповите битове** се реализират с логическа 1 и указват края на предаването на потребителските данни. Използват се :

- един стопов бит;
- един и половина стопови бита;
- два стопови бита.

Един и половина и два стоп бита се използват при по-бавни приемащи устройства, които имат нужда от малко по-дълъг, във времето, край преди новия старт бит, за да обработят приетите данни.

***Данновите битове** са полезната информация, която се пренася. Те могат да варират от 5 до 8. Най-голямото число, което може да се предаде при 5-битов трансфер е $2^5 - 1 = 31$, а при 8-битов трансфер е $2^8 - 5 = 255$.

UART модул

***Бит проверка за грешки** се реализира с един бит и може да бъде 4 вида:

→ проверка по четност (even parity) – битът се установява в 1 или 0, така че сумата от всички единици в изпращания байт да е четно число;

→ проверка по нечетност (odd parity) – битът се установява в 1 или 0, така че сумата от всички единици в изпращания байт да е нечетно число;

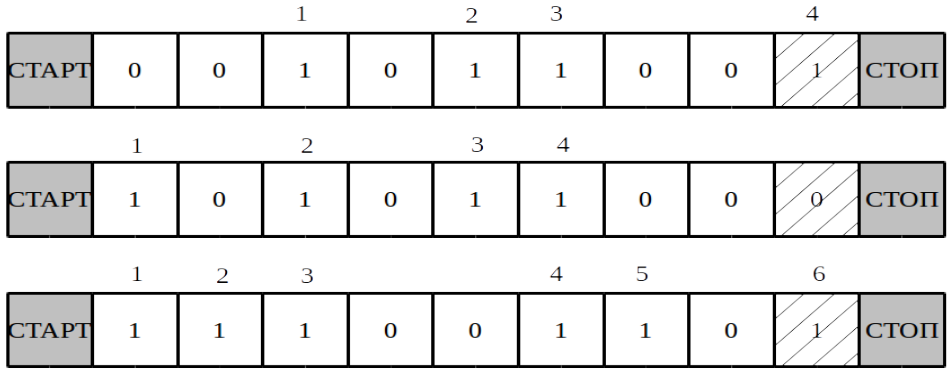
→ проверка с единица (mark parity) – битът е винаги единица;

→ проверка с нула (space parity) – битът е винаги нула.

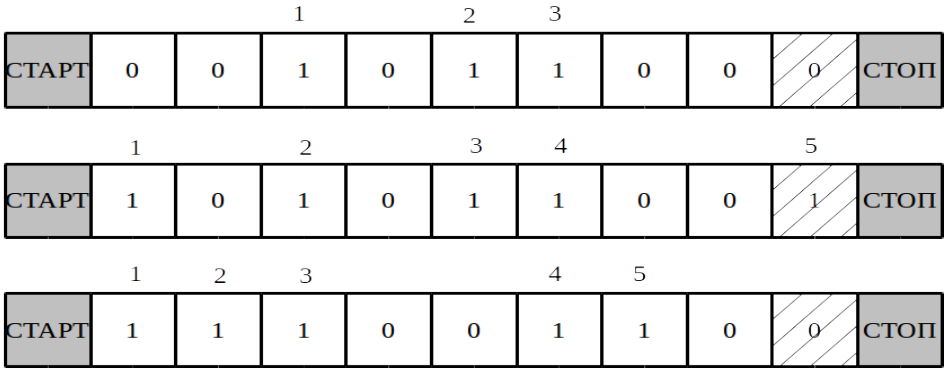
UART модул

На фигурите по-долу са дадени няколко примера за проверка на
правилното предаване на данни.

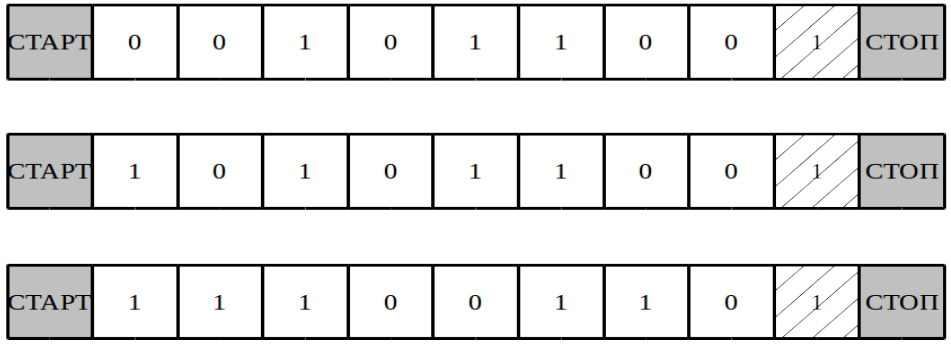
Проверка по четност



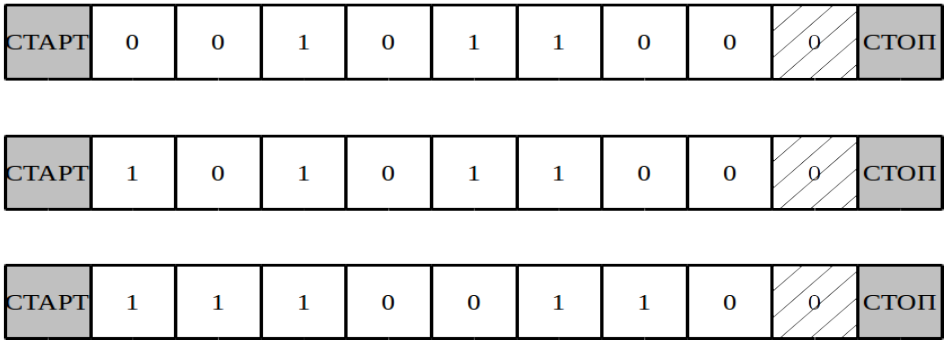
Проверка по нечетност



Проверка с единица



Проверка с нула



UART модул

Може да се изграждат **мрежи**, при които се предават два вида думи - адрес на устройството и данни, за конкретното устройство.

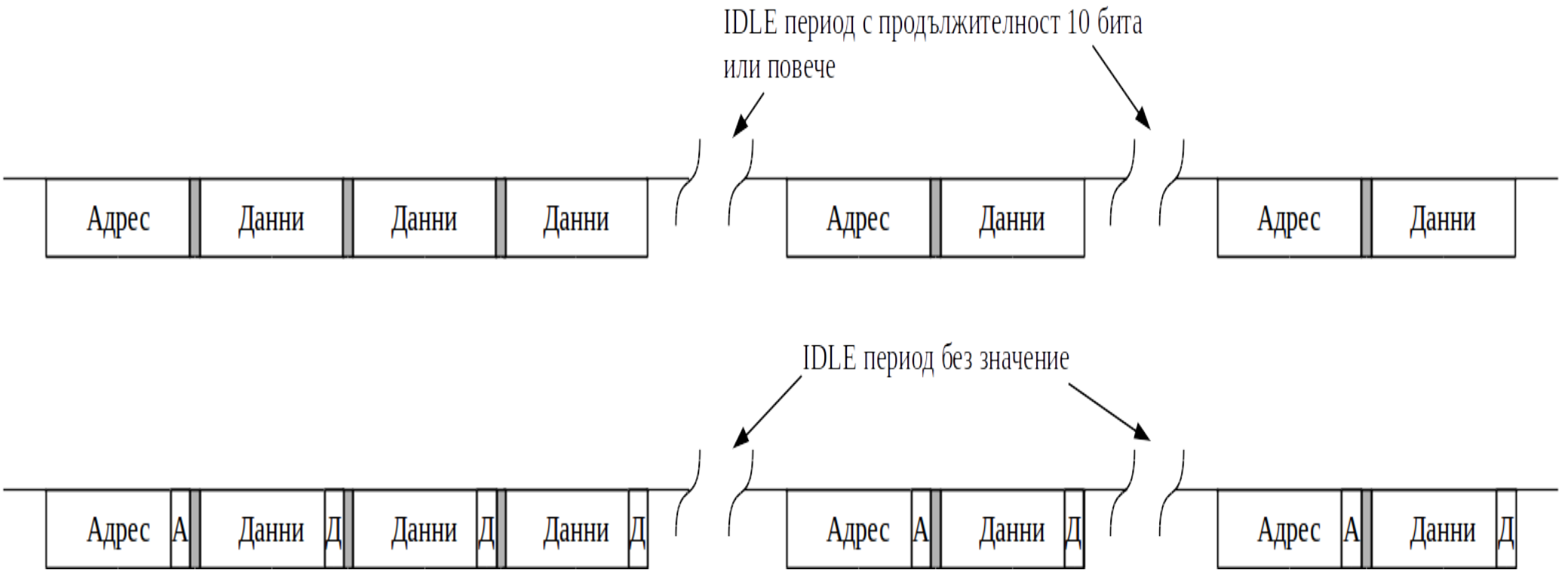
Разликата между адреси и данни се указва с един допълнителен, девети бит в пакета. Когато той е **0 се предават данни**, а когато е **1 се предава адрес**.

UART модул

При мрежите има два варианта за указване на адрес[3]:
*комуникационен протокол от тип **свободна линия** (idle-line) – винаги първата дума е адрес. Ако между думите има време по-голямо или равно на времето, за което се изпращат 10 бита, тогава се счита, че комуникацията е прекратена. След това, първата дума ще бъде възприета като адрес, а думите след нея – като данни.

*комуникационен протокол от тип **адресен бит** (address-bit) – всяка дума съдържа допълнителния, девети адресен бит, който указва дали се изпращат адреси или данни. Думите може да се изпращат през произволно дълги интервали, продължителността на IDLE периодите са без значение.

UART модул

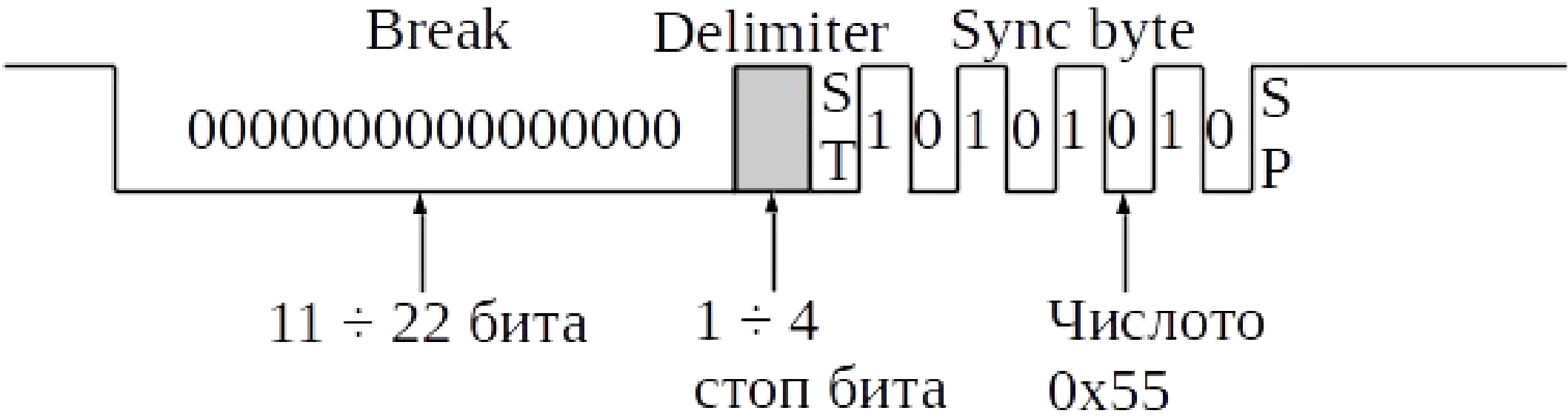

IDLE период с продължителност 10 бита
или повече
Адрес
Данни
Данни
Данни
Адрес
Данни
Адрес
Данни
IDLE период без значение
Адрес
А
Данни
Д
Данни
Д
Данни
Д
Адрес
А
Данни
Д
Адрес
А
Данни
Д

UART модул

Автоматична детекция на скоростта (auto baud rate detection) – специален протокол, чрез който две устройства могат да си настроят автоматично скоростта на обмен на данни.
Първо се изпращат 11 ÷ 22 нули, след това 1 ÷ 4 стоп бита, след това числото 0x55.

Детекцията може да стане веднъж при **инициализацията** на устройствата или след това, в **работен режим**. Полето Break сигнализира, че започва процедура по автоматична детекция.

UART модул


Break
Delimiter
Sync byte
0000000000000000
S
T
1 0 1 0 1 0 1 0
S
P
11 ÷ 22 бита
1 ÷ 4
стоп бита
Числото
0x55

UART модул

Работата и конфигурацията на UART модулите се извършва посредством регистри.

Всеки UART модул съдържа най-малко 6 паралални регистъра, които μPU може да достъпва:

*__входен__ – съдържа пристигналия байт

*__изходен__ – в него се записва байт, който трябва да бъде изпратен

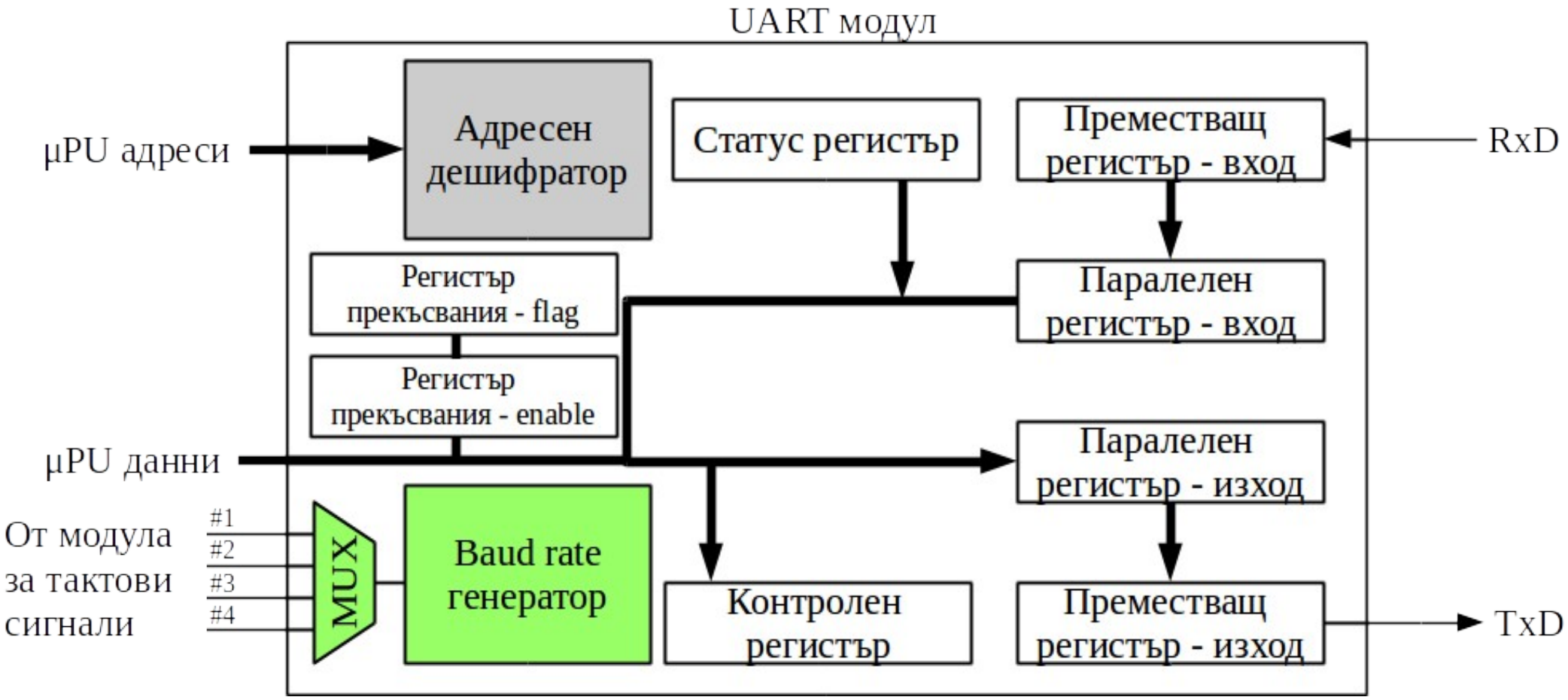
*__контролен__ – съдържа битове, които конфигурират модула

*__статус__ – съдържа битове, отразяващи състоянието на модула в даден момент

*__флагове на прекъсванията –__ съдържа битове, отразяващи състоянията на сигналите за прекъсвания

*__разрешаване на прекъсванията –__ съдържа битове, които разрешават или забраняват дадено прекъсване

UART модул


UART модул
μPU адреси
Адресен дешифратор
Статус регистър
Преместващ регистър - вход
RxD
Регистър прекъсвания - flag
Регистър прекъсвания - enable
Паралелен регистър - вход
μPU данни
Паралелен регистър - изход
От модула за тактови сигнали
#1
#2
#3
#4
MUX
Baud rate генератор
Контролен регистър
Преместващ регистър - изход
TxD

UART модул

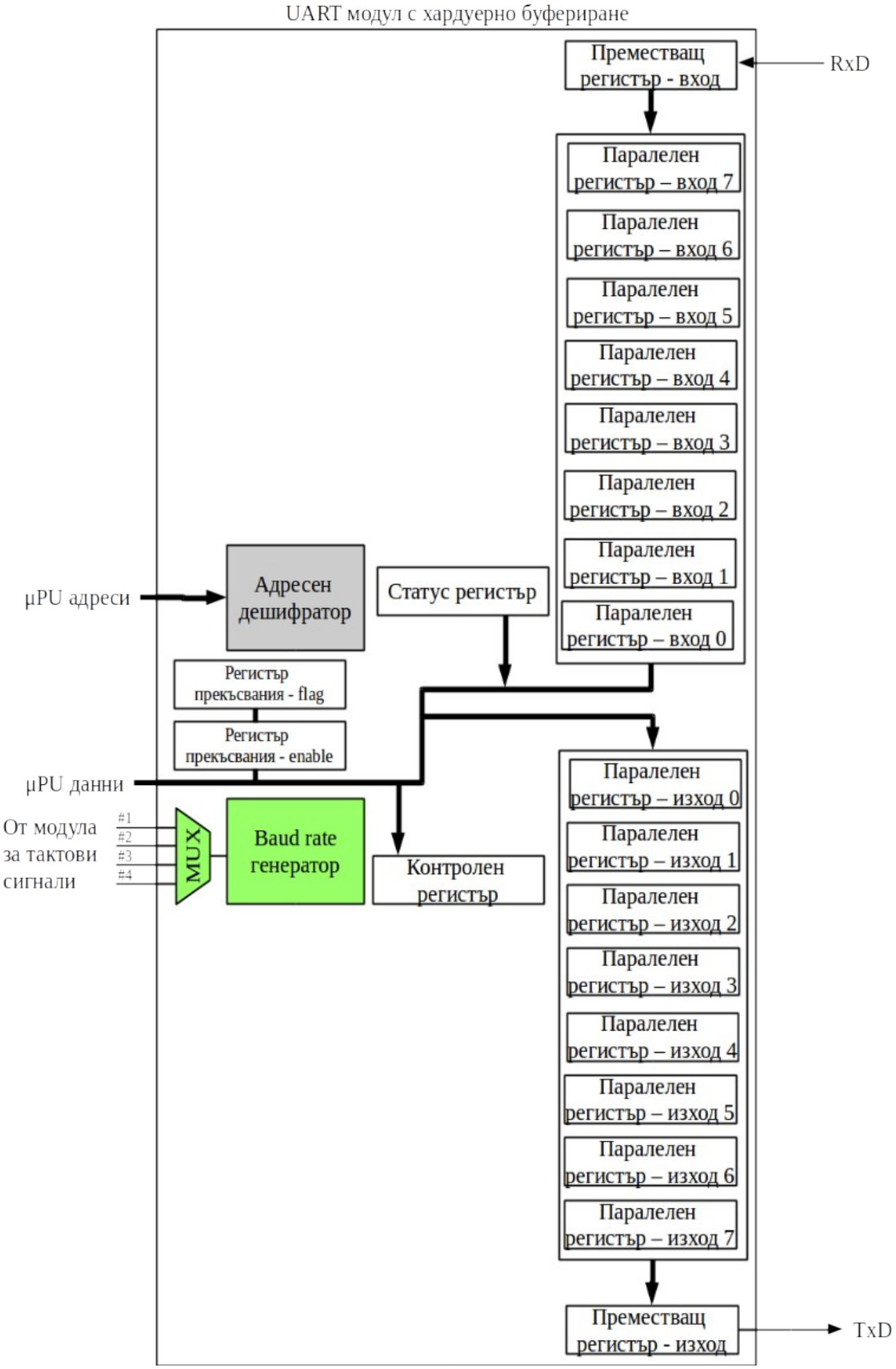
Изпращането на данни от UART модула е **бавен процес**, в сравнение с бързодействието на микропроцесор, работещ на няколко десетки/стотици мегахерца. Едно 8-битово число, без проверка за грешки и при скорост на предаване 9600 b/s, ще се изпраща за около 1042 μs. Ако микропроцесорът трябва да изпрати две такива числа, то той ще запише първото в изходния регистър и докато то се изпраща ще запише и второто. Това ще доведе до загуба на информация и объркване на данните.

UART модул

Затова в микроконтролерите се реализират UART модули с **повече от един входен/изходен регистър**. Тези регистри се групират и се свързват по схемата първи влязъл – първи излязъл (**FIFO**). Благодарение на тях микропроцесорът може да запише няколко байта данни и да продължи изпълнението на програмата. Данните автоматично ще се изпратят от UART модула без намесата на μP. На следващия слайд е демонстрирано 8-байтово **буфериране** на входно-изходните данни в UART модул.

Пример – LM3S9B92 има UART с 16 нива на FIFO. От контролния регистър дълбочината на FIFO може да бъде регулирана на x2, x4, x8, x12, x14 и x16.

UART модул


UART модул с хардуерно буфериране
Преместващ регистър - вход
RxD
Паралелен регистър – вход 7
Паралелен регистър – вход 6
Паралелен регистър – вход 5
Паралелен регистър – вход 4
Паралелен регистър – вход 3
Паралелен регистър – вход 2
Паралелен регистър – вход 1
Паралелен регистър – вход 0
µPU адреси
Адресен дешифратор
Статус регистър
Регистър прекъсвания - flag
Регистър прекъсвания - enable
µPU данни
От модула за тактови сигнали
#1
#2
#3
#4
MUX
Baud rate генератор
Контролен регистър
Паралелен регистър – изход 0
Паралелен регистър – изход 1
Паралелен регистър – изход 2
Паралелен регистър – изход 3
Паралелен регистър – изход 4
Паралелен регистър – изход 5
Паралелен регистър – изход 6
Паралелен регистър – изход 7
Преместващ регистър - изход
TxD

UART модул

Поради **нестабилността** на тактовите генератори и **различните честоти**, на които трябва да работи един UART модул (не винаги може да се получи точната честота на предаване от системната честота), е възможно да възникнат **грешки** при изпращане/приемане. Вероятността това да се случи се дава в техническата документация от производителя.

Пример – на следващия слайд е дадена част от таблица на UART модула на MSP430FR6989. Вижда се, че в някой случай грешките стигат до 37 %.

UART модул

Table 30-5. Recommended Settings for Typical Crystals and Baud Rates[1]

BRCLK	Baud Rate	UCOS16	UCBRx	UCBRFx	UCBRSx[2]	TX Error[2] (%)		RX Error[2] (%)	
						neg	pos	neg	pos
32768	1200	1	1	11	0x25	-2.29	2.25	-2.56	5.35
32768	2400	0	13	-	0xB6	-3.12	3.91	-5.52	8.84
32768	4800	0	6	-	0xEE	-7.62	8.98	-21	10.25
32768	9600	0	3	-	0x92	-17.19	16.02	-23.24	37.3
1000000	9600	1	6	8	0x20	-0.48	0.64	-1.04	1.04
1000000	19200	1	3	4	0x2	-0.8	0.96	-1.84	1.84
1000000	38400	1	1	10	0x0	0	1.76	0	3.44
1000000	57600	0	17	-	0x4A	-2.72	2.56	-3.76	7.28
1000000	115200	0	8	-	0xD6	-7.36	5.6	-17.04	6.96
1048576	9600	1	6	13	0x22	-0.46	0.42	-0.48	1.23
1048576	19200	1	3	6	0xAD	-0.88	0.83	-2.36	1.18
1048576	38400	1	1	11	0x25	-2.29	2.25	-2.56	5.35
1048576	57600	0	18	-	0x11	-2	3.37	-5.31	5.55
1048576	115200	0	9	-	0x08	-5.37	4.49	-5.93	14.92
4000000	9600	1	26	0	0xB6	-0.08	0.16	-0.28	0.2
4000000	19200	1	13	0	0x84	-0.32	0.32	-0.64	0.48
4000000	38400	1	6	8	0x20	-0.48	0.64	-1.04	1.04
4000000	57600	1	4	5	0x55	-0.8	0.64	-1.12	1.76
4000000	115200	1	2	2	0xBB	-1.44	1.28	-3.92	1.68
4000000	230400	0	17	-	0x4A	-2.72	2.56	-3.76	7.28
4194304	9600	1	27	4	0xFB	-0.11	0.1	-0.33	0
4194304	19200	1	13	10	0x55	-0.21	0.21	-0.55	0.33
4194304	38400	1	6	13	0x22	-0.46	0.42	-0.48	1.23
4194304	57600	1	4	8	0xEE	-0.75	0.74	-2	0.87
4194304	115200	1	2	4	0x92	-1.62	1.37	-3.56	2.06
4194304	230400	0	18	-	0x11	-2	3.37	-5.31	5.55
8000000	9600	1	52	1	0x49	-0.08	0.04	-0.1	0.14
8000000	19200	1	26	0	0xB6	-0.08	0.16	-0.28	0.2
8000000	38400	1	13	0	0x84	-0.32	0.32	-0.64	0.48
8000000	57600	1	8	10	0xF7	-0.32	0.32	-1	0.36
8000000	115200	1	4	5	0x55	-0.8	0.64	-1.12	1.76
8000000	230400	1	2	2	0xBB	-1.44	1.28	-3.92	1.68

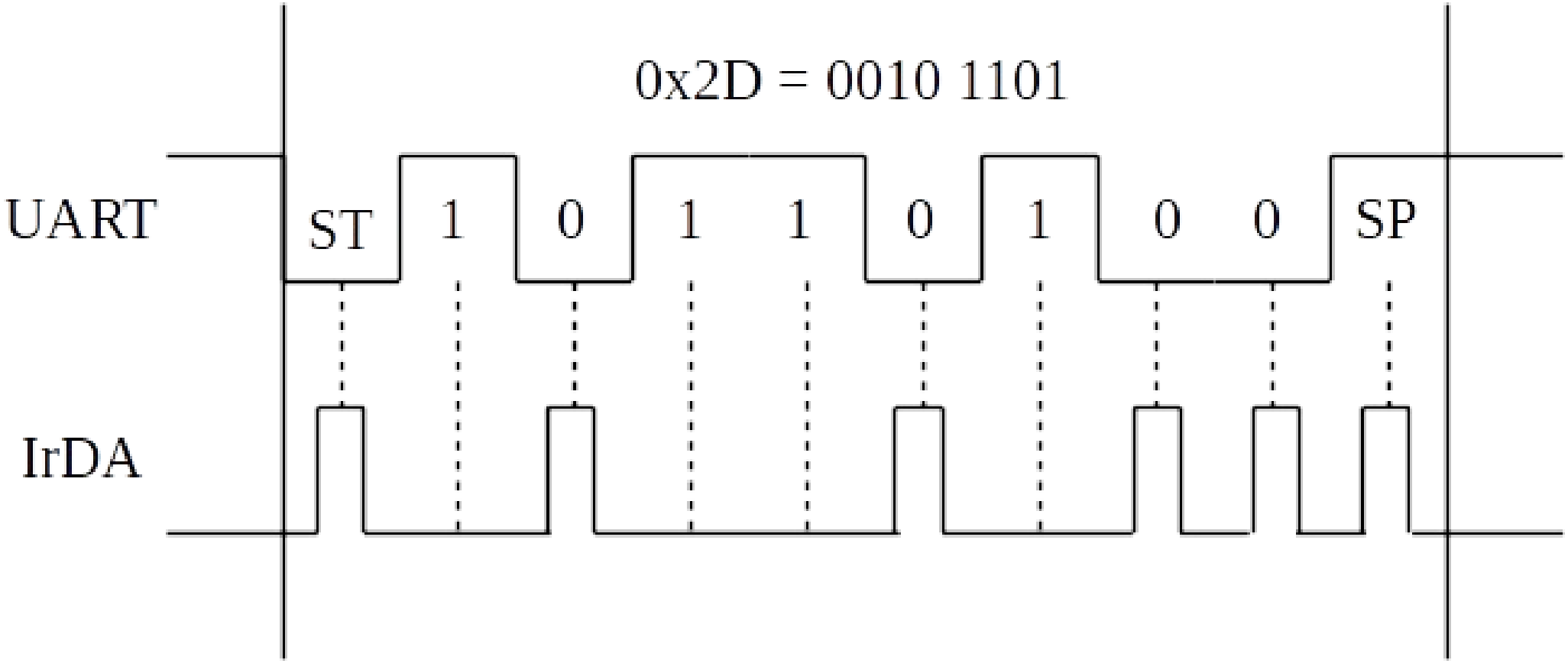
UART модул

IrDA интерфейсът използва UART модула и при него преносната среда е инфрачервен лъч. Приемникът е фотодиод/фототранзистор. Използва се в приложения, изискващи комуникация при разстояния до 2 метра (TV дистанционни).

При IrDA нулите се представят с **кратък импулс** от логическа 1 в средата на бита. Продължителността на импулса е стандартизирана – 3/16 от продължителността на UART битовете (9600 → 104 μs → 3/16*104 = 19.5 μs). Единиците се представят с **липсата на импулс**.

Използват се скорости в интервала 9600 ÷ 115200 baud/s.

UART модул

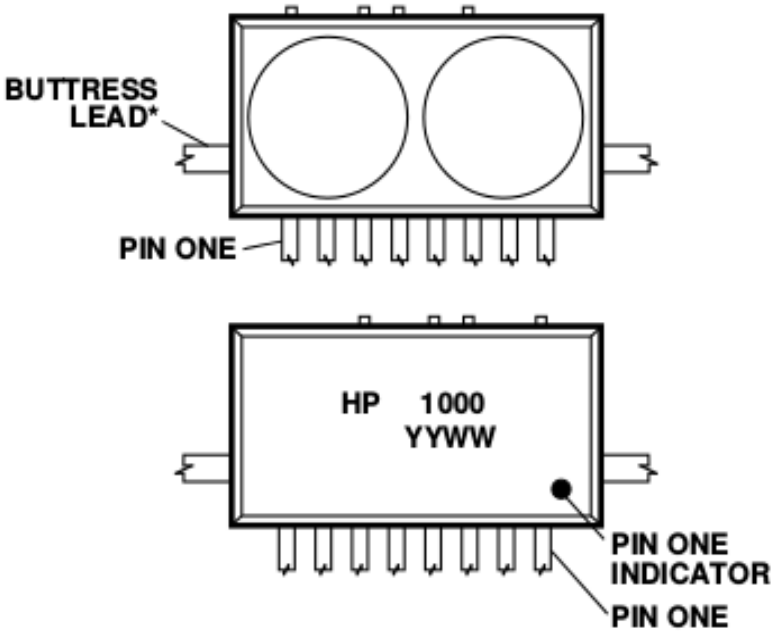


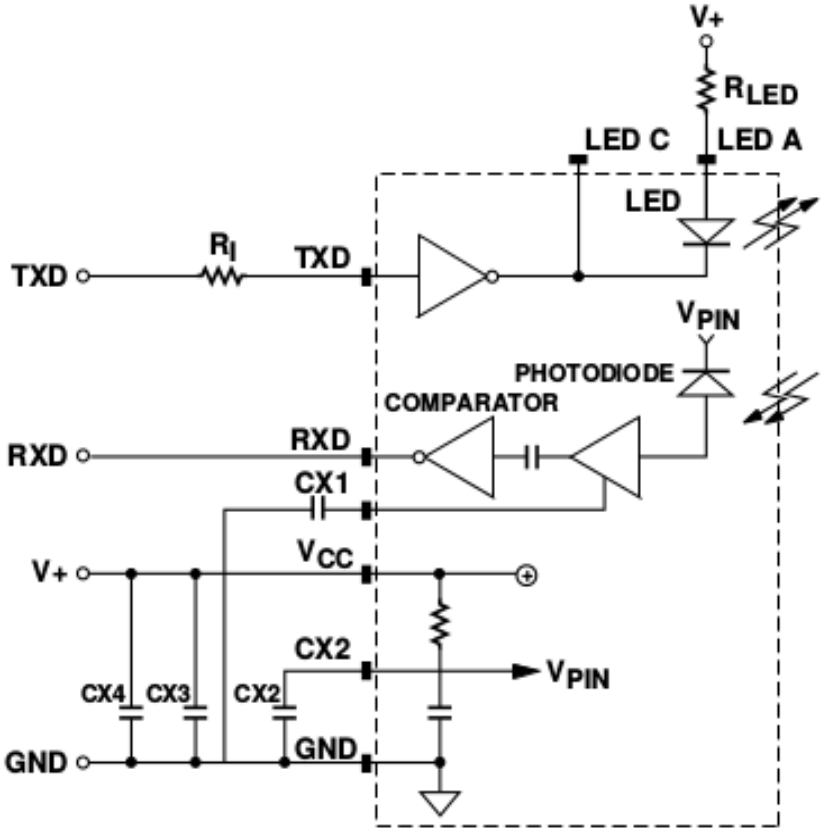
UART модул

Сензор на HP – HSDL1000, пълен дуплекс.

Notebook Computers
Subnotebooks

Schematic





* SIDE BUTTRESS LEADS ARE FOR MECHANICAL STABILITY AND SHOULD NOT BE CONNECTED TO ANY ELECTRICAL POTENTIAL.

UART модул

IrDA интерфейсът има няколко разновидности, които дефинират различни скорости на предаване:
*SIR (до 115.2 kbit/s)
*MIR (до 1 Mbit/s)
*FIR (до 4 Mbit/s)
*VFIR (до 4 Mbit/s)

Интерфейс RS485

RS485 (IEEE Recommended Standard 485) е сериен, асинхронен, диференциален, напрежителен интерфейс [2], [6].

Позволява свързване на разстояние до **1200 метра**.

Максимална скорост на предаване – **10 Mbit/s** (зависима от дължината).

Позволява да се изграждат **мрежи с до 256 устройства**.

Интерфейс RS485

Няма стандартизиран куплунг (инженерът сам решава какъв да сложи).

Няма стандартизиран протокол за обмен на данни (може да се използва UART).

Предаването е от тип **полу-дуплекс** (може да се предават данни в двете посоки, но в даден момент от време може само в едната) или **пълен дуплекс**.

Интерфейс RS485

Диференциалните сигнали се **отбелязват с A и B**.

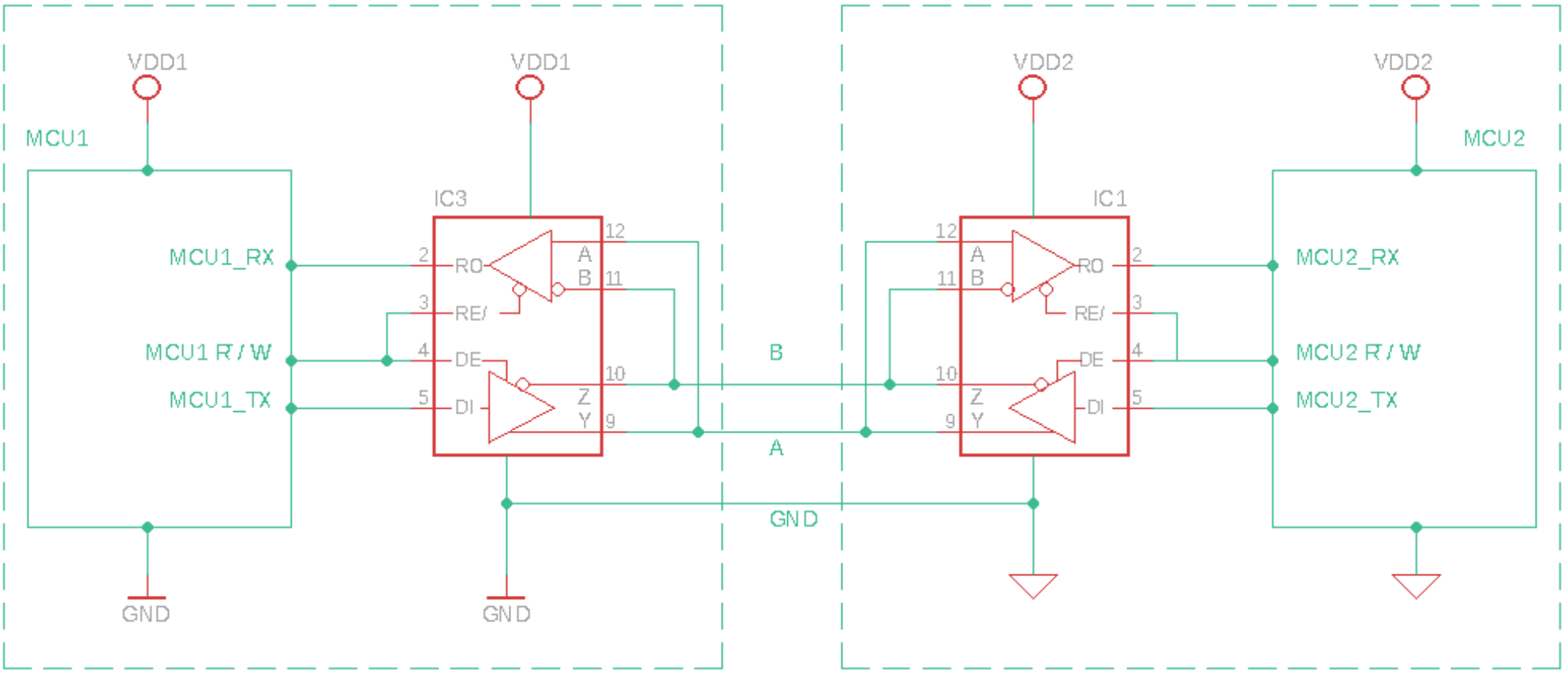
Диференциалните сигнали в повечето случаи са **еднополярни, положителни** (освен когато има отрицателно постояннотоково отместване).

Диференциалните сигнали може да са с **постояннотоково отместване -7 ÷ 12 V** спрямо маса.

Разликата на диференциалните сигнали (т.е. размахът от връх-до-връх на данните) **не трябва да е по-голяма 6 V**. Също - **не трябва да е по-малка от 1.5 V**, а в приемника може да е затихнала до **0.2 V** [2].

Интерфейс RS485

Свързване при полу-дуплекс.



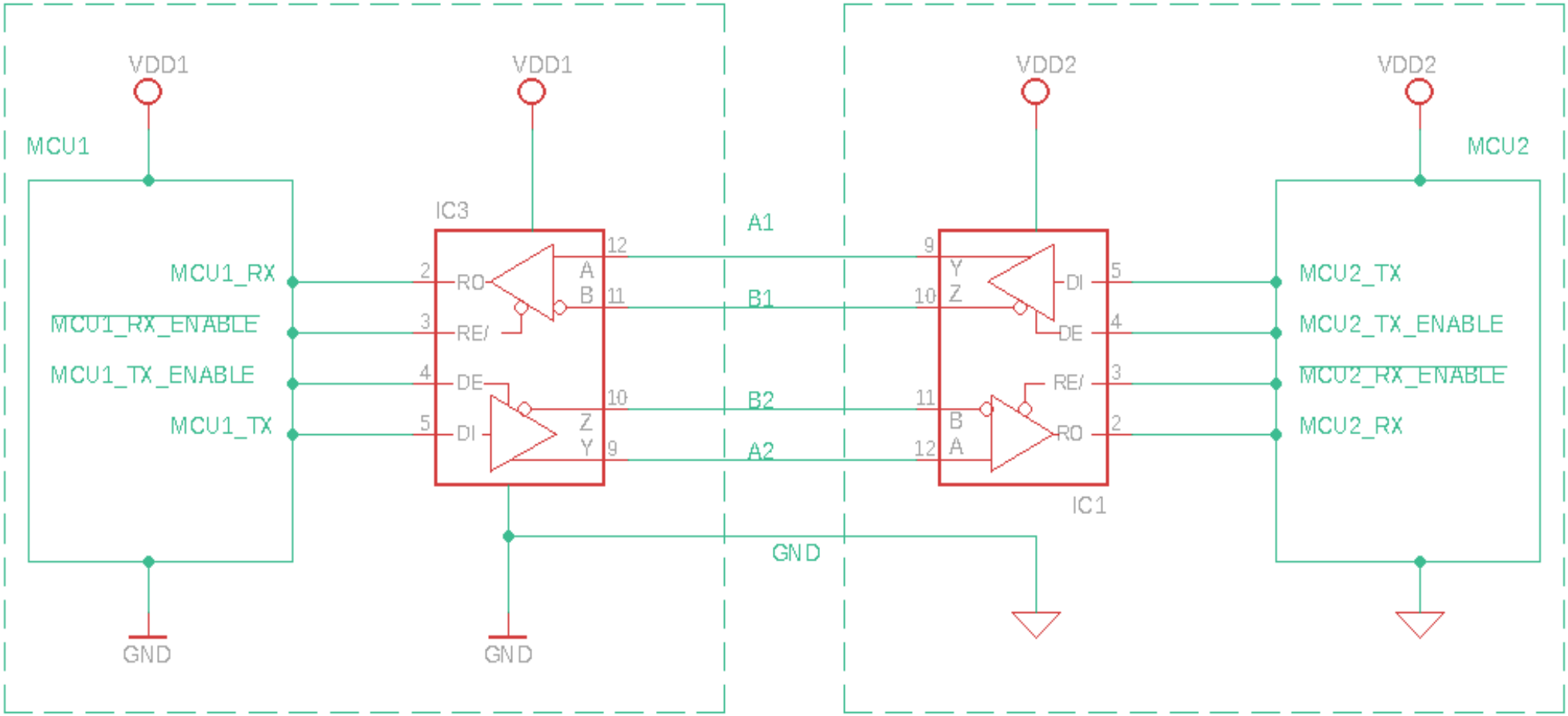
$$|V_A - V_B| \leq 6V \qquad |V_A - V_B| \geq 1.5V$$

$$V_A\text{-}V_{GND} \leq 12V \qquad V_A\text{-}V_{GND} \geq -7V$$

$$V_B\text{-}V_{GND} \leq 12V \qquad V_B\text{-}V_{GND} \geq -7V$$

Интерфейс RS485

Свързване при пълен дуплекс.



Интерфейс RS485

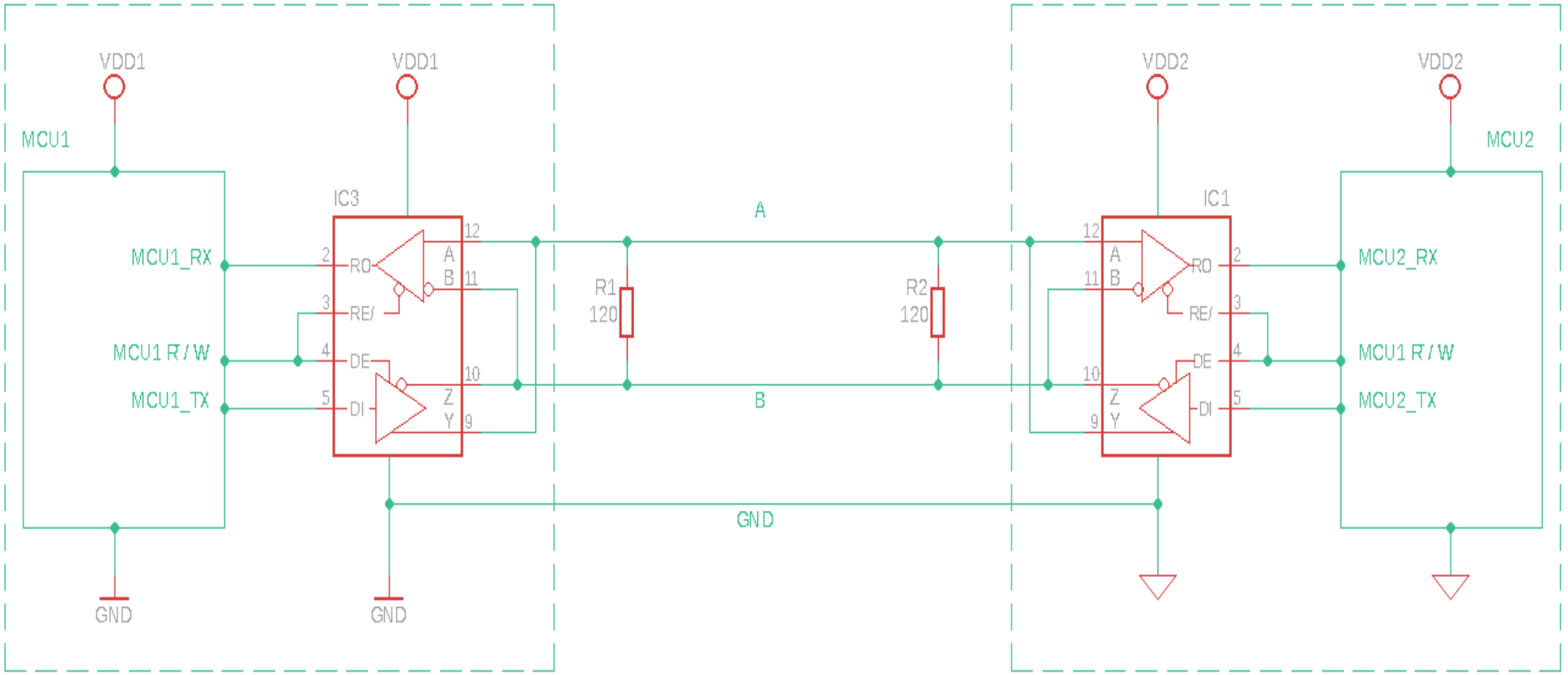
При високи скорости на обмен и неподходящо характеристично съпротивление на преносната среда (кабела) може да се появи отразяване на сигнала, което да доведе до грешки.

Тогава се прибягва до **терминиране на линията** с резистори в двата ѝ края (при полу-дуплексен режим).

Обаче - терминирането повишава консумацията на енергия от интерфейса.

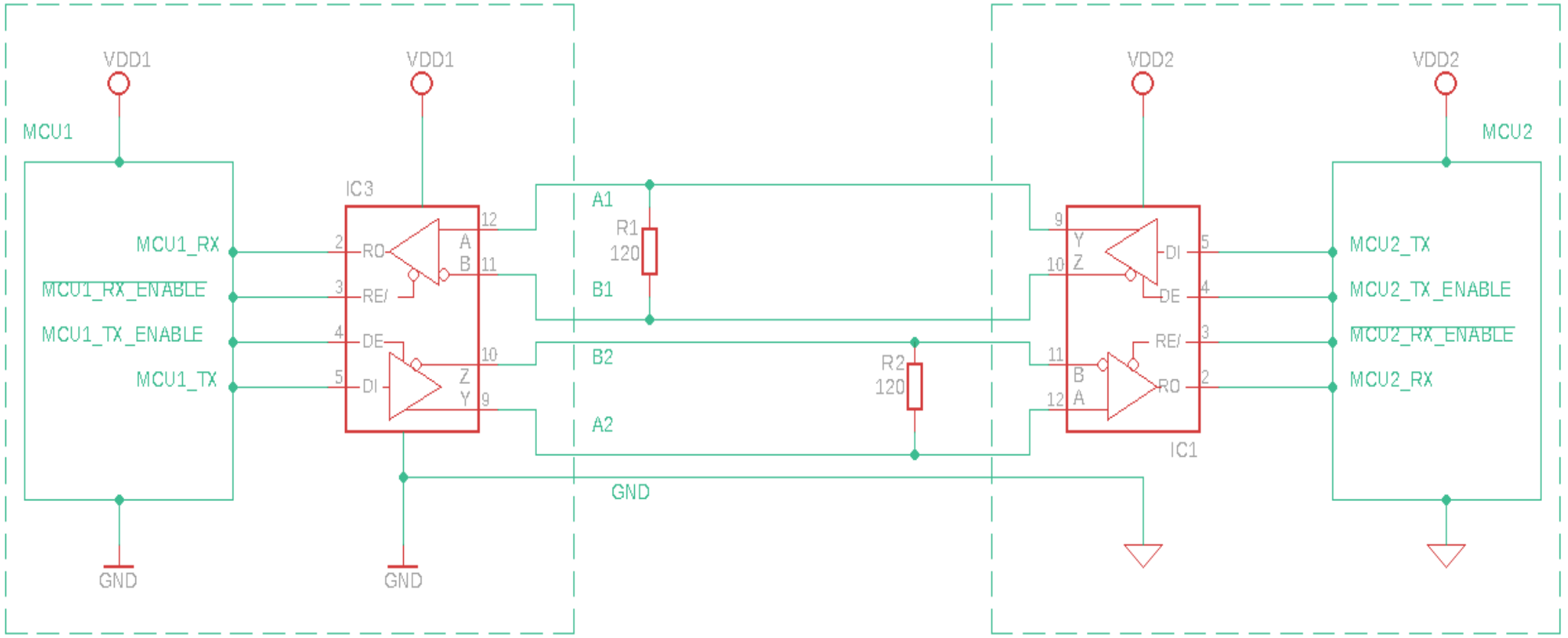
Интерфейс RS485

Терминиране при полу-дуплекс.



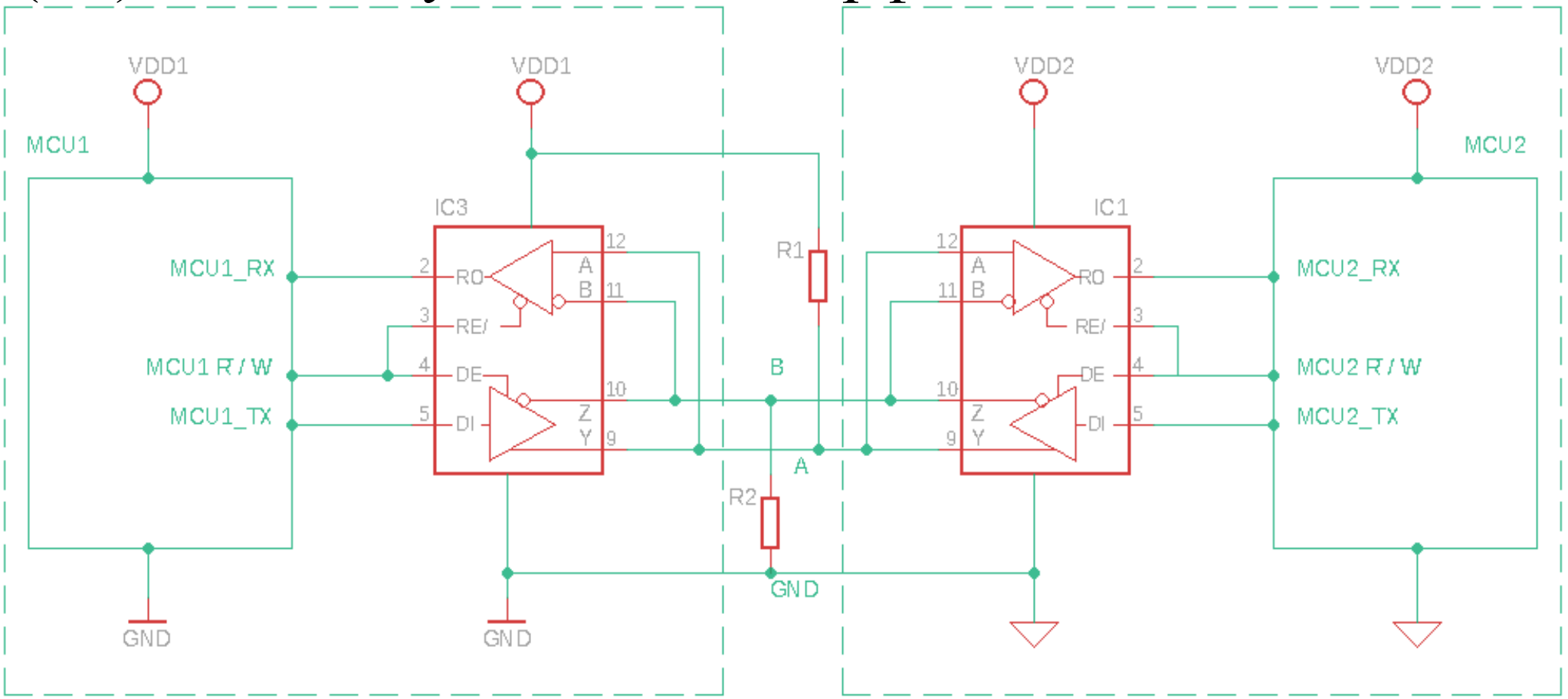
Интерфейс RS485

Терминиране при пълен дуплекс.
Резисторите се слагат при приемника.



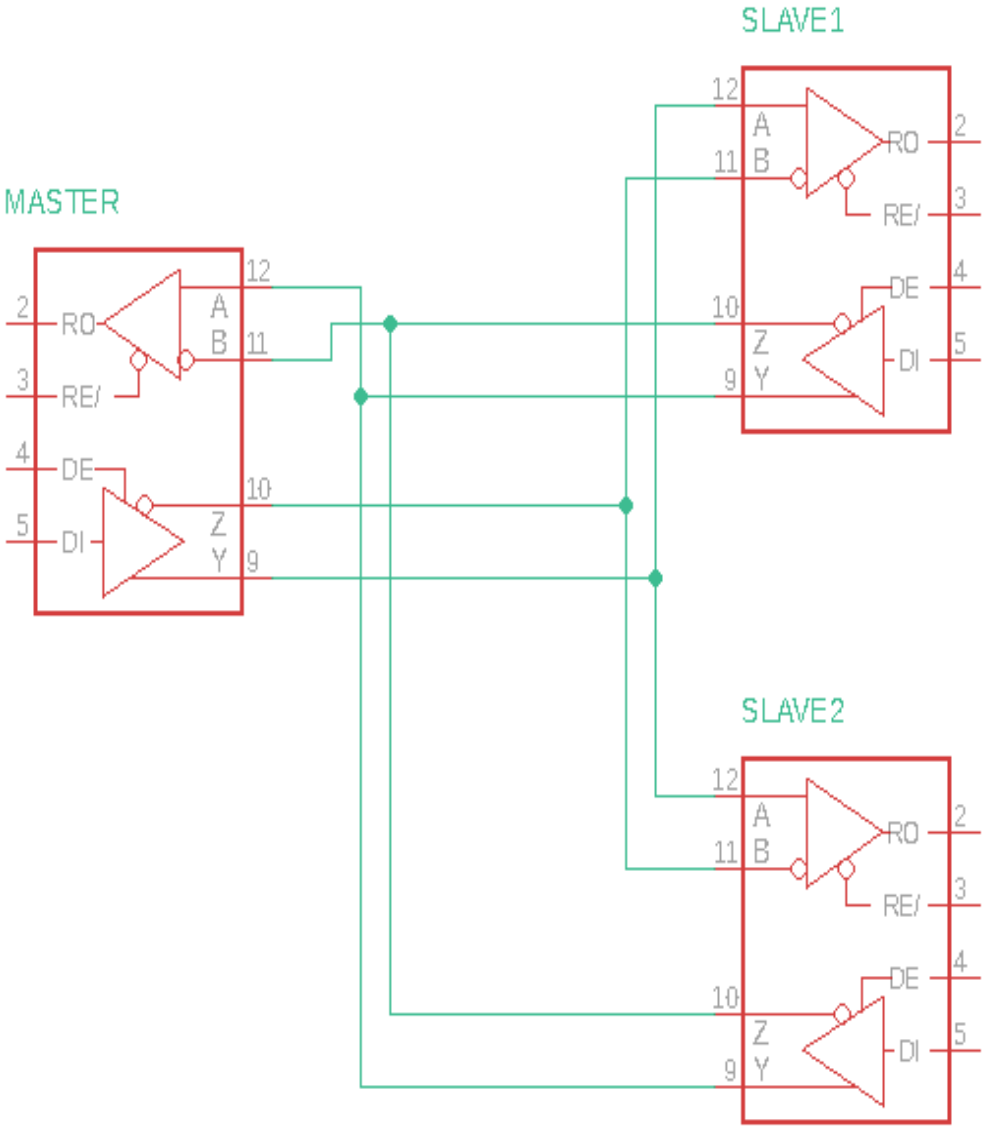
Интерфейс RS485

Когато не се предават данни и всички изходи са във високоимпедансно състояние трябва да има издърпващи резистори към захранване (R1) и маса (R2), които да установят интерфейса в логическа 1.



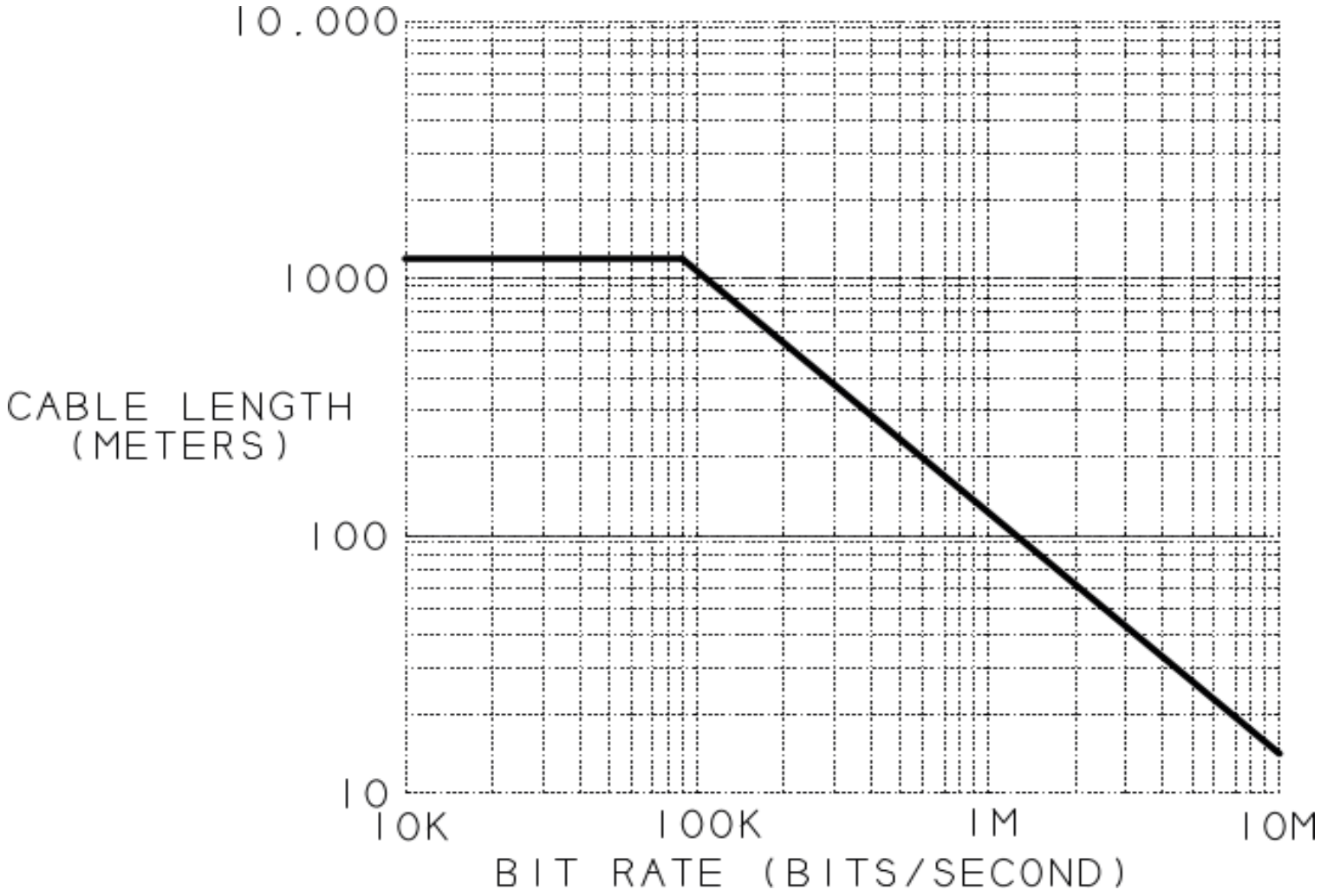
Интерфейс RS485

Когато се изграждат мрежи е допустимо да се **свържат два изхода в паралел**. Драйверните чипове имат токо-ограничение (**250 mA** по стандарт). На схемата е показан MASTER, който може да изпраща данни към SLAVE1 и 2, както и да приема данни от тях. Но SLAVE1 не може да комуникира директно със SLAVE2.



Интерфейс RS485

При RS485 **скоростта на обмен се ограничава от дължината на кабела** [2].



Интерфейс USB

USB (universal serial bus) е сериен, асинхронен, диференциален интерфейс, проектиран за връзка на вградено устройство с персонален компютър или за връзка на две вградени устройства помежду си.

Една версия на USB използва напрежителен интерфейс, а друга – токов.

Максималната дължина на проводниците не трябва да надхвърля **5 метра**.

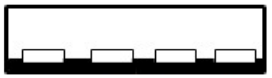
Интерфейс USB

Скоростта на обмен на информация варира в зависимост от използвания режим на входно/изходните стъпала.

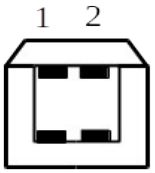
Съществуват 4 режима на работа при USB интерфейса:
***low-speed** – максимална скорост на обмен 1.5 Mbps (USB 1.0)
***full-speed** - максимална скорост на обмен 12 Mbps (USB 1.0)
***high-speed** - максимална скорост на обмен 480 Mbps (USB 2.0)
***super-speed** - максимална скорост на обмен 5 Gbps (USB 3.0)
***super-speed+** - максимална скорост на обмен 20 Gbps (USB 3.2)
***super-speed+ -** максимална скорост на обмен 40 Gbps (USB 4.0)

Интерфейс USB

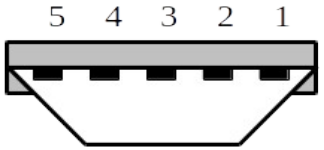
Тип A



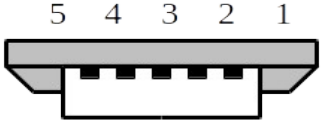
Тип B



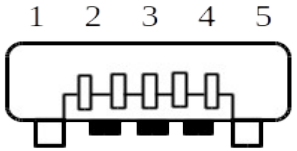
Тип Мини-A



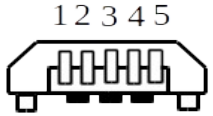
Тип Мини-B



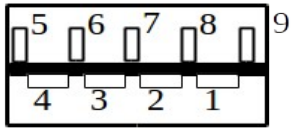
Тип Микро-A



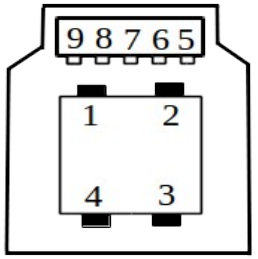
Тип Микро-B



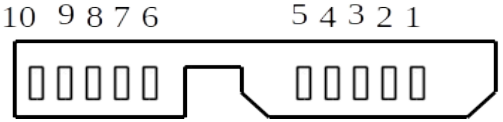
Тип A, USB 3.0



Тип B, USB 3.0



Тип Микро-B, USB 3.0



Извод	Описание
1	+5V
2	D-
3	D+
4	GND

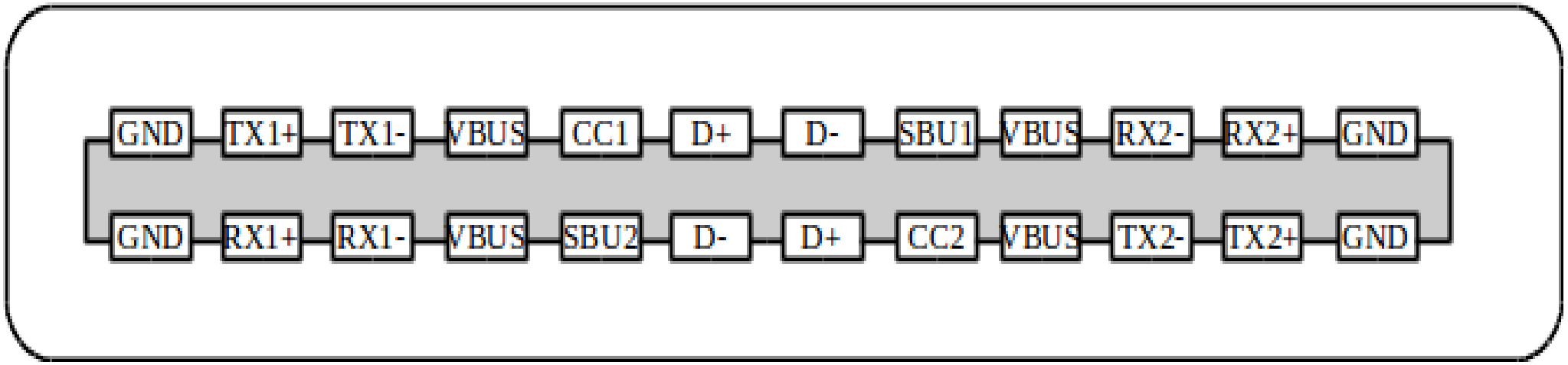
Извод	Описание
1	+5V
2	D-
3	D+
4	GND
5	RD-
6	RD+
7	RGND
8	TD-
9	TD+

Извод	Описание
1	+5V
2	D-
3	D+
4	GND
5	RD-
6	RD+
7	RGND
8	TD-
9	TD+
10	TGND

Извод	Описание
1	+5V
2	D-
3	D+
4	ID
5	GND

Интерфейс USB

Тип С стандартен



Интерфейс USB

Някои от най-важните сигнали са:
***D+** и **D-** са сигналите, с които се предава информацията. Използват се диференциални входно/изходни стъпала и затова са ни необходими два проводника.

Чрез USB (1.0, 2.0) протокола D+ и D- се ориентират или като входове, или като изходи и по този начин обменът на данни може да е двустранен. Но в даден момент информация може да се предава само в една посока (**полудуплексен режим**).

При **USB 3.0** имаме **дуплексен режим**.

Интерфейс USB

***ID** - използва се само в мини- и микро- куплунгите. Това е сигнал, указващ дали устройството, което се свързва е главно (Host) или подчинено (Device).
→ ако ID е свързан към GND → главно
→ ако ID е оставен плаващ → подчинено

***RD-, RGND, RD+** се използват в USB 3.0. Тази диференциална двойка сигнали образува допълнителен канал за приемане (R – receive) на данни.

***TD-, TGND, TD+** се използват в USB 3.0. Тази диференциална двойка сигнали образува допълнителен канал за предаване (T – transmit) на данни.

Интерфейс USB

Захранващото напрежение е +5 V, и може да отдаде ток до [4]:
***500 mA при USB 2.0**
***900 mA при USB 3.0**

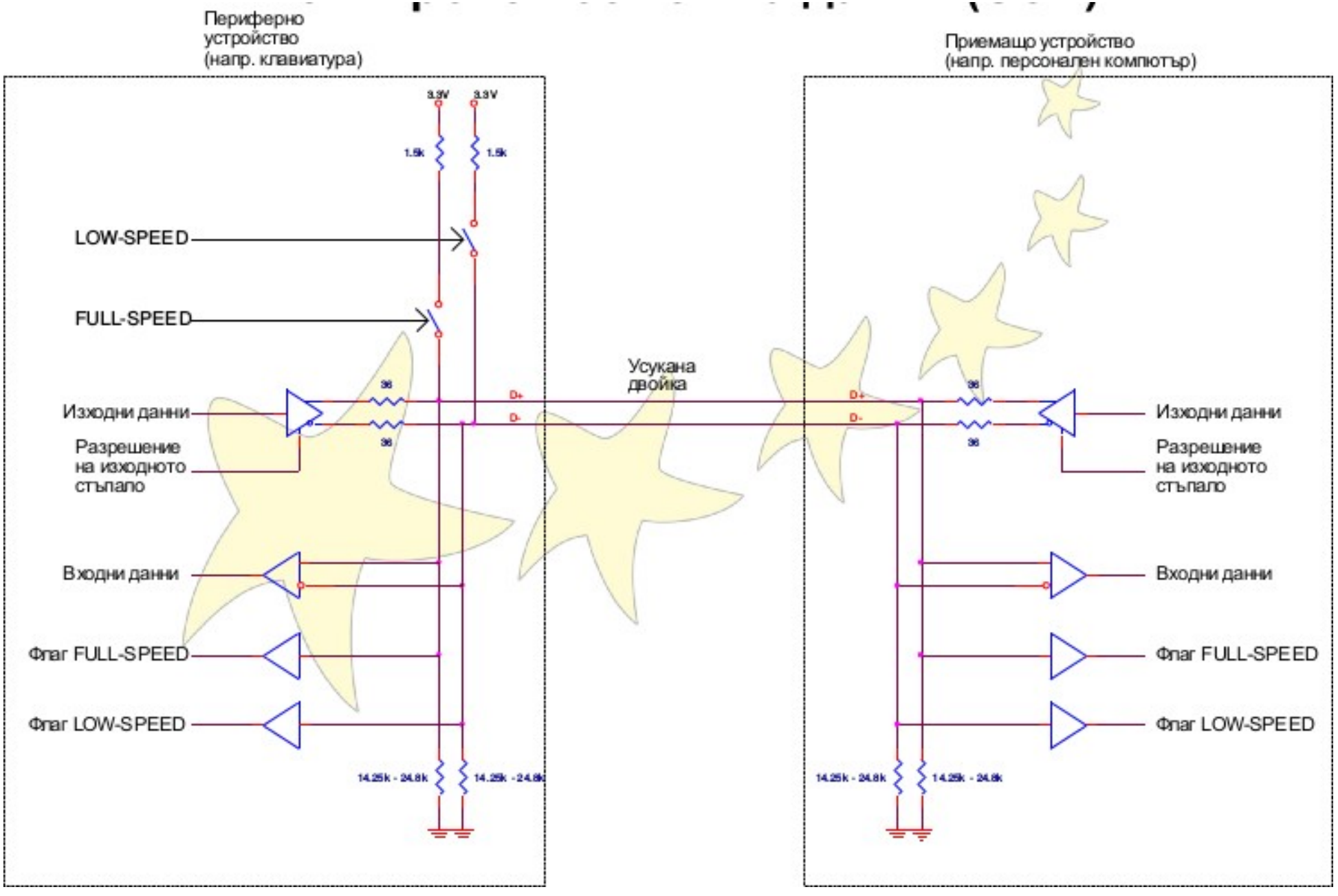
Това позволява вградената система, която ще се включва към
персонален компютър, да няма собствено захранване и да използва това на интерфейса.

Интерфейс USB

На следващия слайд е показана принципна схема на два USB трансийвъра (на едно подчинено Device и едно главно Host устройство). Вижда се, че скоростта на интерфейса се определя в зависимост от това дали D+ или D- линията е свързана с издърпващ към захранване резистор.

*Ако има **издърпващ резистор на D-**, ще се използва **low-speed USB**.
*Ако има **издърпващ резистор на D+**, ще се използва **full-speed USB**.
*Ако има **издърпващ резистор на D+ и се изпрати специален handshake пакет**, ще се използва **high-speed USB**.

Интерфейс USB



Интерфейс USB

1. При **high speed** подчиненото устройство първоначално се представя на главното като full speed устройство.

2. След това се пуска high speed handshake пакет, с който главното устройство разбира, че подчиненото е с high speed възможности.

3. Издърпващият резистор за full speed се изключва.

4. Драйверното стъпало за LS (Low-speed) и FS (Full-speed) свързва двата си изхода към маса. По този начин изходното съпротивление на това стъпало плюс терминиращият резистор 36 Ω образуват **еквивалентен терминиращ резистор от 45 Ω** (+/- 10%) за HS стъпалото. Затова HS няма отделни терминиращи резистори.

5. Включва се допълнително драйверно **токово стъпало** за HS (High-speed) **предаване**. Когато то предава, данните вървят по едната линия, а другата е дадена към маса.

Интерфейс USB

6. Включва се входно **токово стъпало** за **приемане**.

7. Включва се детектор за невалиден (по ниво) сигнал по USB интерфейса.

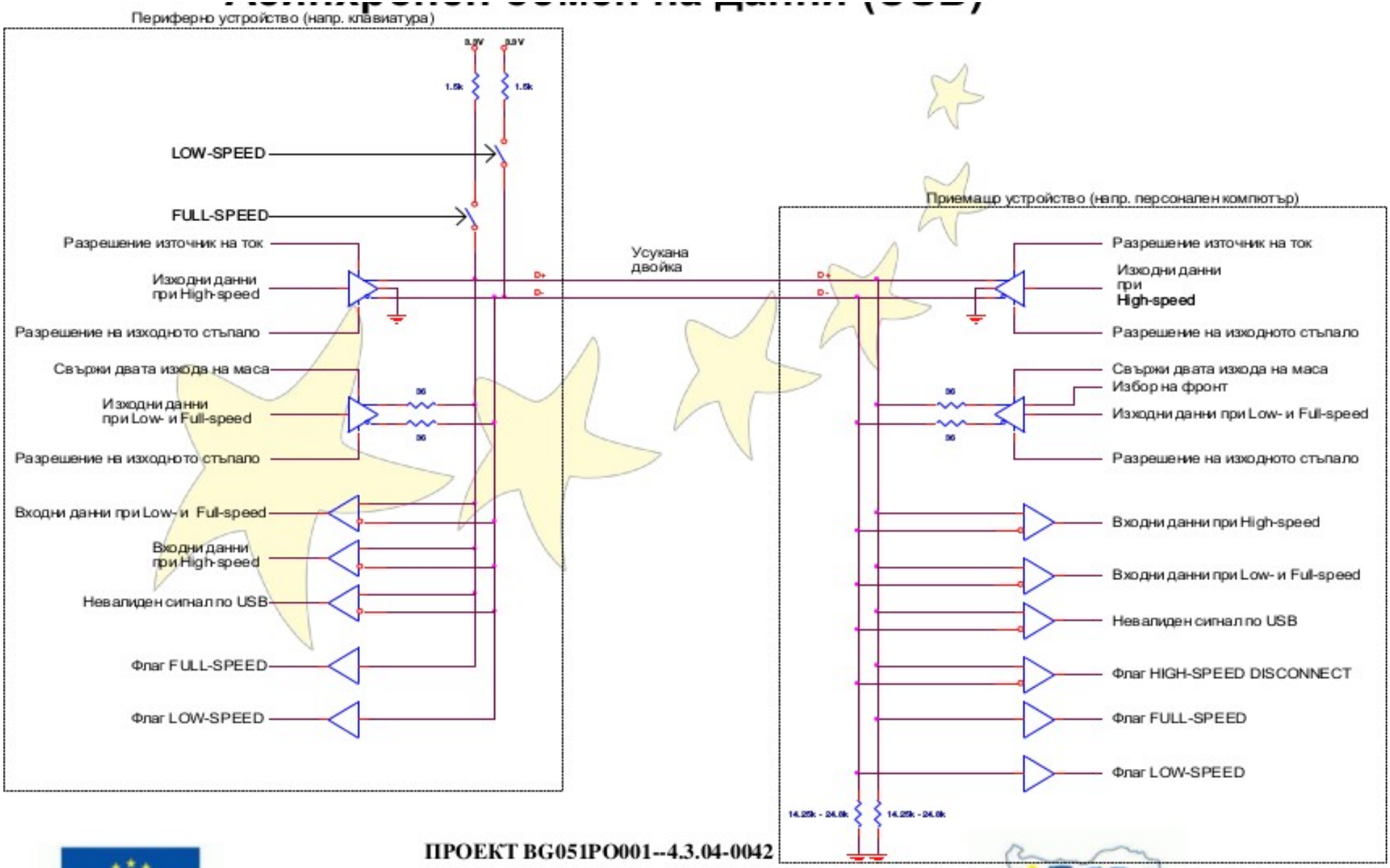
8. Включва се детектор за разкачено HS устройство.

!!!ВНИМАНИЕ!!! Стойностите на резисторите указват вид на устройството.

14.25 ÷ 24.8 kΩ – хъб

0.9 ÷ 1.575 kΩ – подчинено устройство

Интерфейс USB



ПРОЕКТ BG051PO001--4.3.04-0042

Интерфейс USB

Low и Full speed		
	Изход	**Вход**
Логическа 1	D+ = (2.8 ÷ 3.6)V D- = (0 ÷ 0.3)V	D+ ≥ 2.0V (D+ - D-) ≥ 0.2V
Логическа 0	D+ = (0 ÷ 0.3)V D- = (2.8 ÷ 3.6)V	D- ≥ 2.0V (D- - D+) ≥ 0.2V
High speed		
	Изход	**Вход**
Логическа 1	D+ = (0.36 ÷ 0.44)V D- = (-0.01 ÷ 0.01)V I_{MIN} = (0.36 – 0.01)/45=7.7 mA I_{MAX} = (0.44 + 0.01)/45=10 mA	V_{INMIN} и V_{INMAX}, продължителност на нарастващия и падащия фронт, jitter се взимат от eye диаграма
Логическа 0	D+ = (-0.01 ÷ 0.01)V D- = (0.36 ÷ 0.44)V I_{MIN} = 7.7 mA I_{MAX} = 10 mA	V_{INMIN} и V_{INMAX}, продължителност на нарастващия и падащия фронт, jitter се взимат от eye диаграма

Интерфейс USB

EYE диаграма. Вляво – аналогов ключ, който отговаря на USB high-speed спецификацията. Отдясно – ключ, който не отговаря. Повече информация в [5].

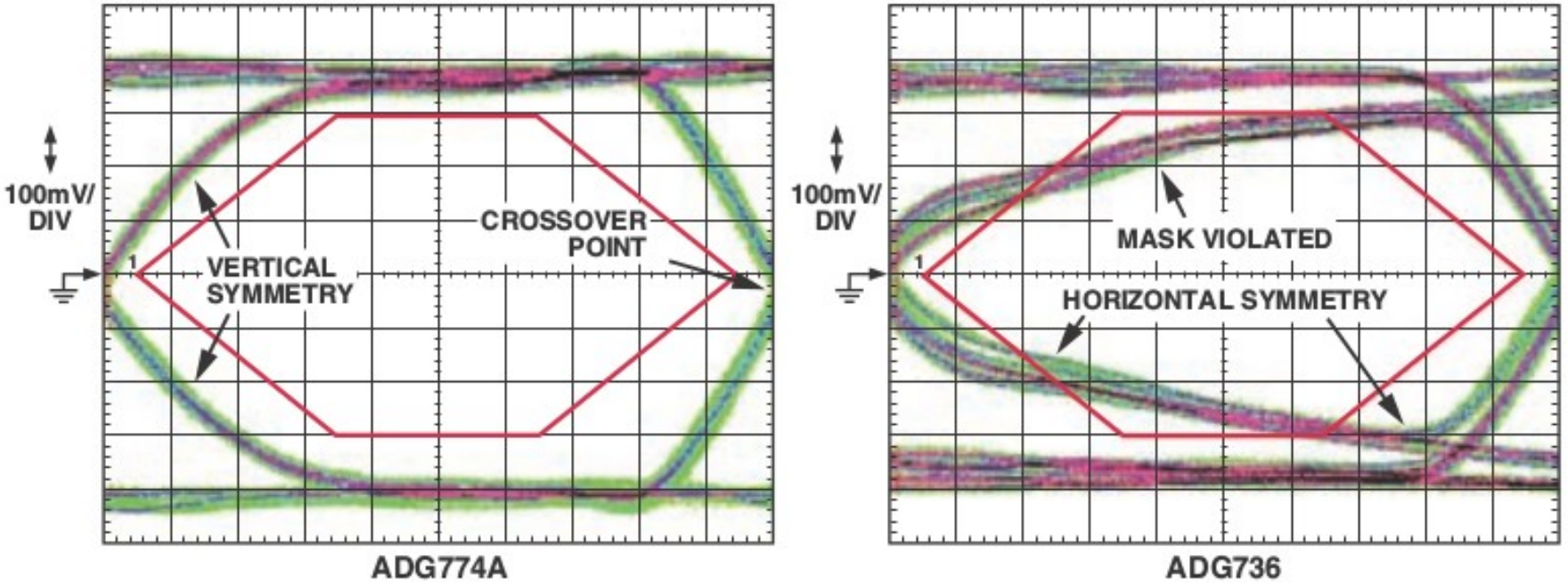


Figure 5. Comparison of the ADG774A and ADG736 at USB Hi-Speed.

Analog Dialogue 40-01, January (2006)

Интерфейс USB

USB протоколът използва 4 вида трансфери:

***Control** трансфери – използват се за **първоначална настройка** на USB интерфейса. Всички устройства трябва да поддържат control трансфери. Използват се също и за запитвания (requests) към свързаното устройство.

***Bulk** трансфери – използват се, когато времето за обмен на **голямо количество данни** не е от значение. Типично приложение на bulk трансфер е използването му във външни хард дискове, скенери, принтери, USB-UART преобразуватели и др.

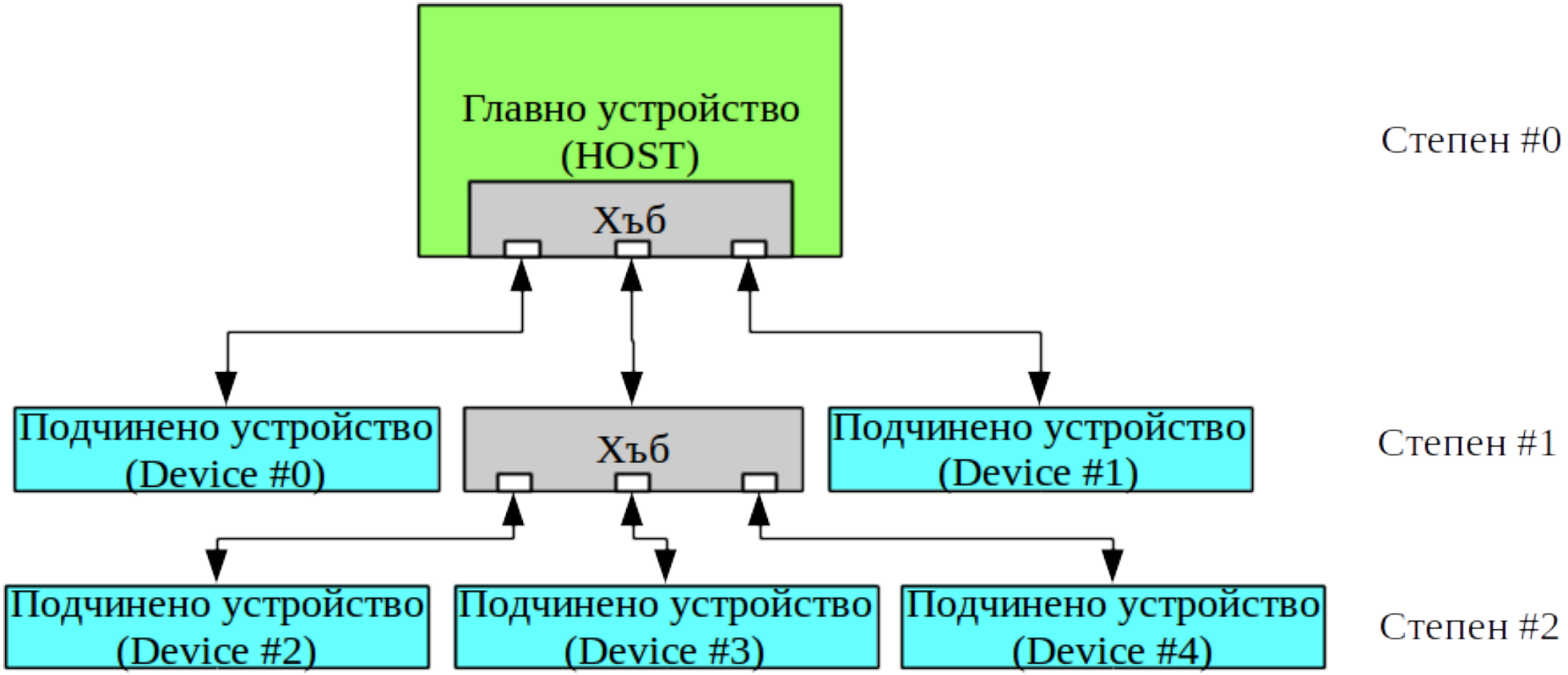
Интерфейс USB

***Interrupt** трансфери – главното устройство гарантира, че ще бъде направен опит за трансфер веднага след съответната заявка. Дали обаче трансферът ще се осъществи не е гарантирано. Това означава, че **скоростта на обмен може да варира**. Този вид трансфер се използва за **малки количества информация**, като например в мишки, клавиатури, джойстици и др.

***Isochronous** трансфери – използват се за пренос на данни, когато **точното време за пристигането им е от значение** и когато **малко грешки** при обмена могат да бъдат толерирани (няма проверка за грешки). Пример за устройства използващи isochronous трансферите са уеб камерите, предаване на звук от микрофон и други подобни “streaming” приложения.

Интерфейс USB

Свързването на устройства става в мрежа от тип многостепенна звезда.



Интерфейс USB

От програмна гледна точка, обменът на данни се осъществява чрез комуникационни буфери, наречени **крайни точки** (endpoints). Крайните точки съхраняват приетата информация или информацията, която трябва да бъде изпратена [4]. Всички

Всяка крайна точка се описва със следните параметри:

***номер** на точката

***посока** (входна/изходна)

***максимален брой байтове**, които може да приеме/предаде

Интерфейс USB

При включване на подчинено устройство, главното устройство започва процес, наречен “зачисляване” (enumeration), който се състои от 12 фази. Най-важните от тях са:

1. започва се контролен трансфер (control transfer),
2. задава се фиксиран адрес 0x00 на подчиненото устройство,
3. прочита се описателна структура от данни, **наречена главен дескриптор** (device descriptor), започвайки от крайна точка 0, адрес 0x00
4. задава се уникален адрес на подчиненото устройство,
5. прочитат се всички останали дескриптори (техният брой е известен от главния дескриптор)
6. зарежда се драйвер,
7. зарежда се дадена конфигурация на устройството (всички устройства трябва да имат поне 1 конфигурация)

Интерфейс USB

По време на зачисляването се изграждат канали (pipes) за комуникация.

Канал за комуникация е връзката между крайна точка в подчиненото устройство и софтуера на главното устройство.

Един от параметрите на каналът за комуникация е видът на трансфера (control, bulk, interrupt, isochronous).

Друг от параметрите е дали се изпращат данни или команди:
*message pipes – използват се само в контролните трансфери. Те са двупосочни.
*stream pipes – използват се в bulk, interrupt и isochronous трансфери. Те са еднопосочни.

Интерфейс USB

Дескрипторите описват възможностите за комуникация на подчиненото устройство. В тях е описано кой **клас драйвер** да бъде зареден от операционната система на главното устройство.

Интерфейс USB

Base Class	Descriptor Usage	Description	Base Class	Descriptor Usage	Description
00h	Device	Use class information in the Interface Descriptors	**0Eh**	Interface	Video
01h	Interface	Audio	**0Fh**	Interface	Personal Healthcare
02h	Both	Communications and CDC Control	**10h**	Interface	Audio/Video Devices
03h	Interface	HID (Human Interface Device)	**11h**	Device	Billboard Device Class
05h	Interface	Physical	**12h**	Interface	USB Type-C Bridge Class
06h	Interface	Image	**DCh**	Both	Diagnostic Device
07h	Interface	Printer	**E0h**	Interface	Wireless Controller
08h	Interface	Mass Storage	**EFh**	Both	Miscellaneous
09h	Device	Hub	**FEh**	Interface	Application Specific
0Ah	Interface	CDC-Data	**FFh**	Both	Vendor Specific
0Bh	Interface	Smart Card			
0Dh	Interface	Content Security			

Интерфейс USB

В Линукс на всяко едно USB устройство се присвоява по един системен файл в /dev директорията. Потребителска програма на C може да си комуникира с устройството посредством този файл и системните функции:

```
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
```

open( )
close( )
ioctl( )
write( )
read( )

На следващият слайд е показан пример с TMC клас устройство.

Интерфейс USB

```
typedef struct {
  dev_type_e dev_type; //One of the dev_type_e
  const char *dev_name; //The name of device file including path
  int device_descriptor;          //Returned from open( )
struct usbtmc_attribute attr;   //For ioctl( ), don't edit this property
    char* c_string_cmd; //Send cmd to device
    char* c_string_reply; //Read device reply to cmd
    int max_bytes; //Max. number of bytes to be read
}dev_property_t;
```

Интерфейс USB

```
int usb_tmc_open(dev_property_t *dev){
    int success = -1;

    dev->device_descriptor = open(dev->dev_name,O_RDWR);

    if(dev->device_descriptor > 0){
        dev->attr.attribute = USBTMC_ATTRIB_READ_MODE;
        dev->attr.value = USBTMC_ATTRIB_VAL_READ;
        ioctl(dev->device_descriptor, USBTMC_IOCTL_SET_ATTRIBUTE, &dev->attr);
        success = 0;
    }

    return success;
}

void usb_tmc_close(dev_property_t *dev){
    if(dev->device_descriptor){
      close(dev->device_descriptor);
      dev->device_descriptor = 0;
    }
}
```

Интерфейс USB

```
void usb_tmc_write(dev_property_t *dev){
    write(dev->device_descriptor, dev->c_string_cmd, strlen(dev->c_string_cmd));
}

void usb_tmc_read(dev_property_t *dev){
    char file_buffer[FILE_BUFF_SIZE];
    int bytes_read;

    bytes_read = read(dev->device_descriptor, file_buffer, FILE_BUFF_SIZE);

    if(bytes_read >= dev->max_bytes){
        bytes_read = dev->max_bytes-1;
    }

    file_buffer[bytes_read] = '\0';

    strcpy(dev->c_string_reply, file_buffer);
}
```

Интерфейс Ethernet

Ethernet (IEEE802.3-2002) е сериен, асинхронен, диференциален, двуполярен, напрежителен интерфейс [7].

Нивата на сигнала са +/- 2.5V.

Скоростта на предаване е стандартизирана:
*10 Mbit/s
*100 Mbit/s
*1/2.5/10/25/40/50/100/200/400/800 Gbit/s
Преносна среда:
*коаксиален кабел
*оптичен кабел
*UTP кабел

Интерфейс Ethernet

Разстоянията на предаване варират за всяка една версия на интерфейса. Типично за кабел е 100 ÷ 2000 m, а за оптика – 10 ÷ 40 km.

Видовете кодиране на информацията:

*10 Mbit/s – Манчестър кодиране

*100 Mbit/s – 4B/5B + Scramble + MLT3 + кодиране

*1000 Mbit/s – 4D-PAM5 + 8B/10B кодиране + Trellis модулация

Тип на предаване:

*полу-дуплекс (10 Mbit/s)

*пълен дуплекс (100 Mbit/s; 1000 Mbit/s)

Интерфейсът е **галванично разделен** чрез импулсен трансформатор.

Интерфейс Ethernet

Конекторът, използван от интерфейса се нарича RJ-45. Той съдържа 4 двойки кабели [8], [9].

***10/100** Mbit/s се използват само **2** диференциални двойки

***1000** Mbit/s се използват 4.

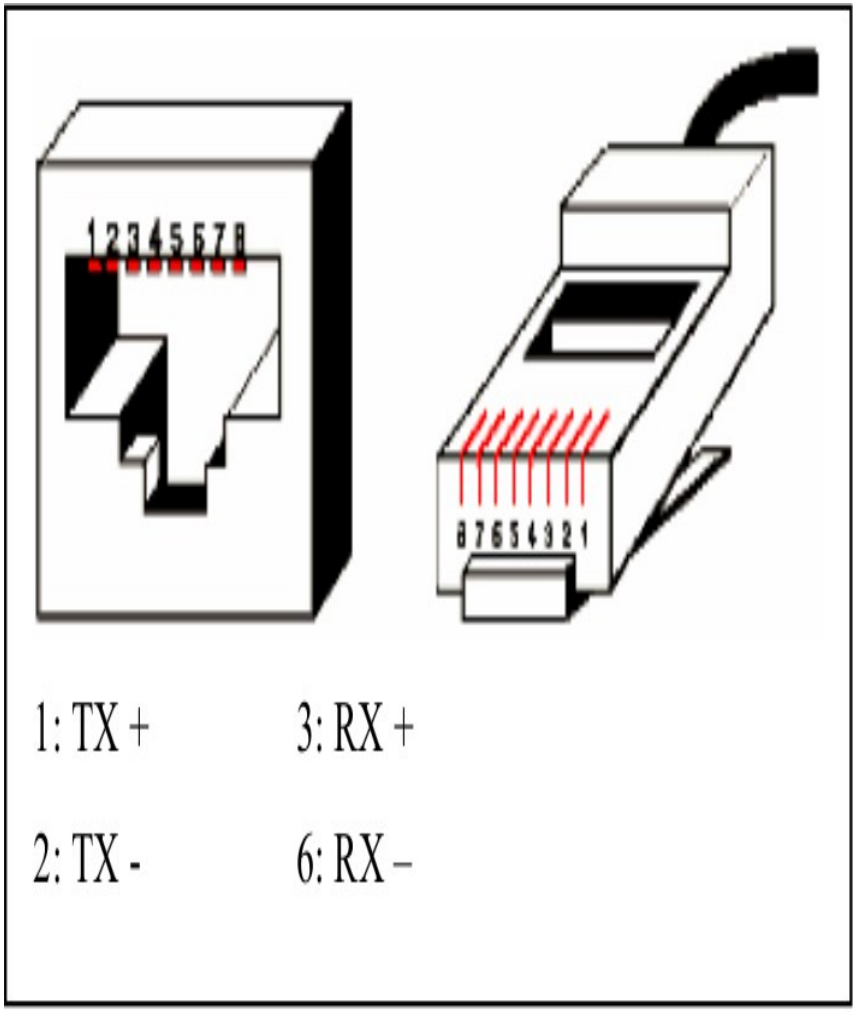
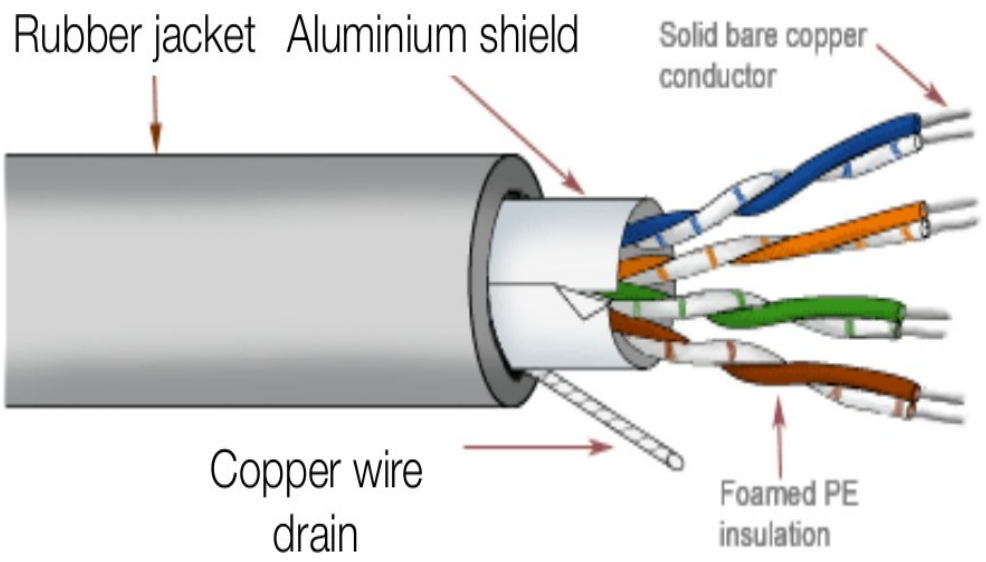


Figure 11. RJ45 Connector

Интерфейс Ethernet

10/100	
Цвят на кабела	**Име на сигнала**
Зелен-бял	+TX
Зелен	-TX
Оранжев-бял	+RX
Оранжев	-RX
1000	
Цвят на кабела	**Име на сигнала**
Зелен-бял	+A
Зелен	-A
Оранжев-бял	+B
Оранжев	-B
Син-бял	+C
Син	-C
Кафяв-бял	+D
Кафяв	-D



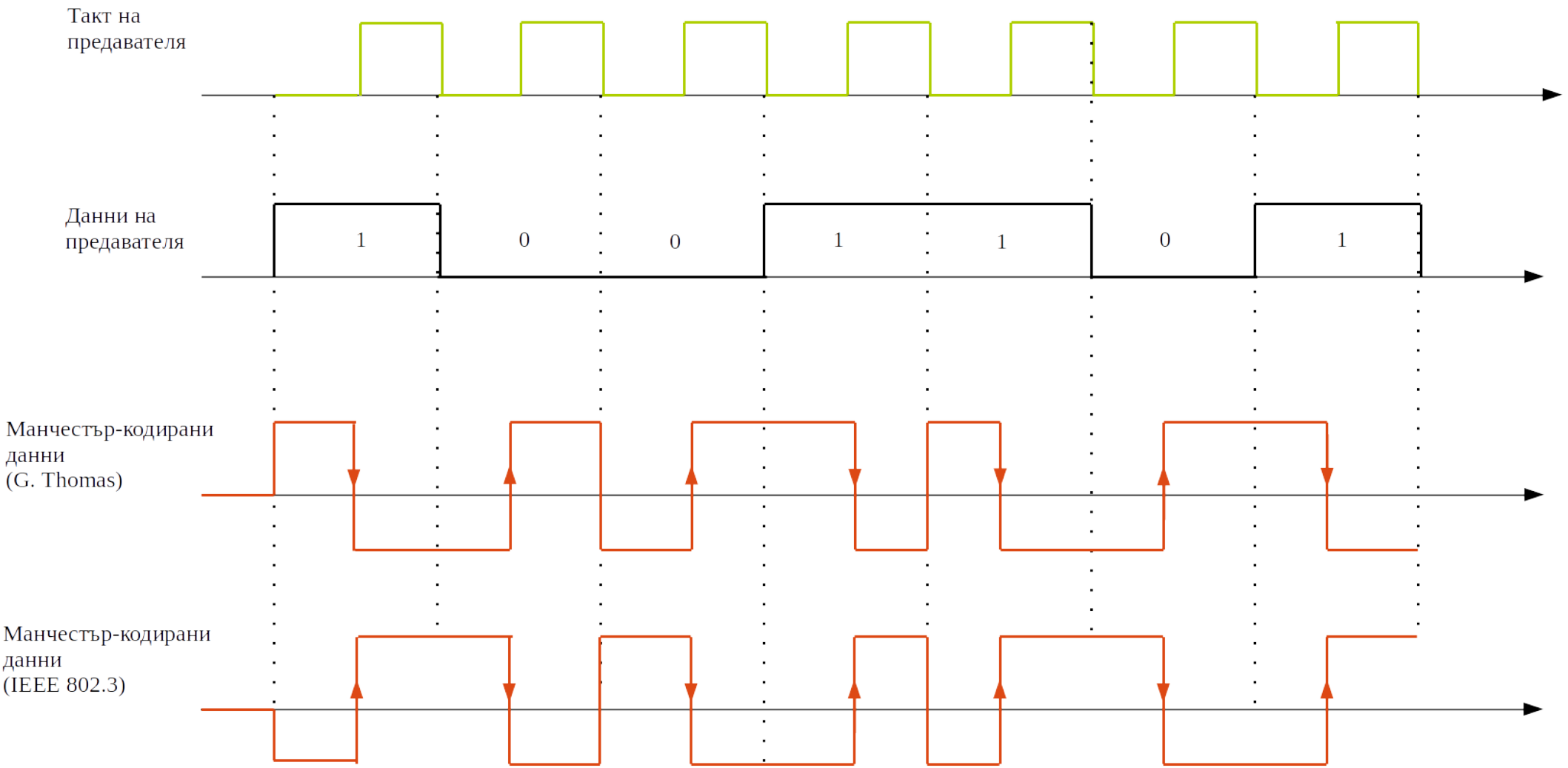
Интерфейс Ethernet

Код на Манчестър - всеки бит се представя с фронт на импулса. Промяната на фронта става в средата на периода на оригиналния сигнал (при асинхронна комуникация), което позволява чрез допълнителна схема в приемащото устройство, да бъде възстановен такта от предаващото устройство посредством данновия сигнал.

Постояннотоковата съставка на сигнала е 0 V и се използват двуполярни сигнали. Това позволява кодирането да се използва в капацитивно или индуктивно разделени трансийвъри.

Интерфейс Ethernet

Съществуват два версии на кода:

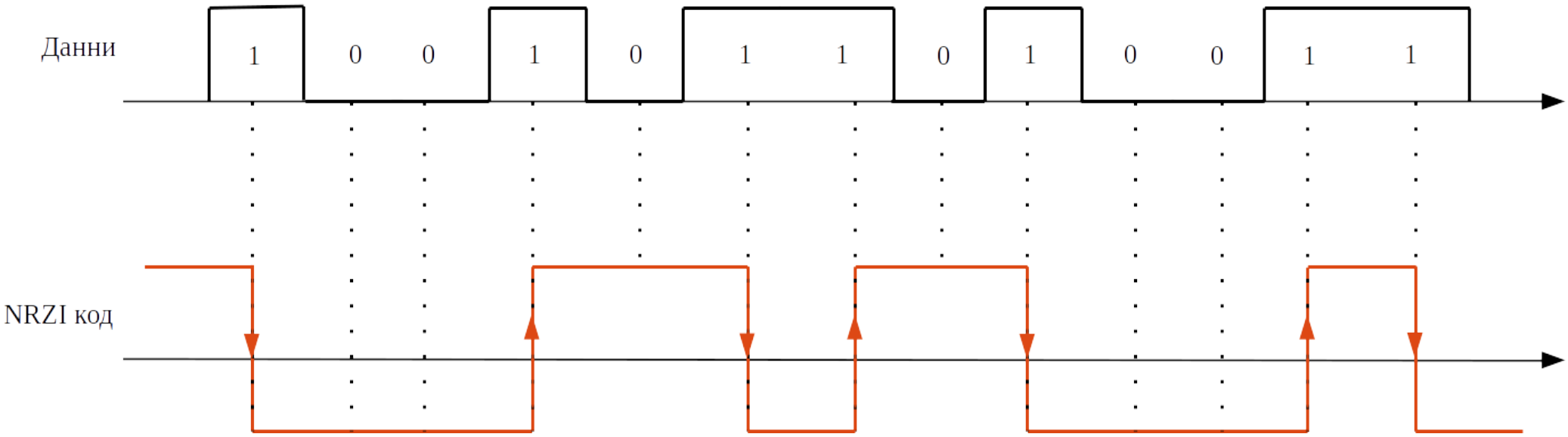


Интерфейс Ethernet

NRZI (non-return to zero, inverted) е кодиране на данни, при което логическата единица е представена с преход (1 → 0 или 0 → 1), а логическата нула – с липса на преход (запазва се предишното ниво).

Това означава, че едни и същи байтове, които се предават ще **изглеждат по различен начин**, защото ще са зависими от байтовете, предадени преди тях.

Интерфейс Ethernet



Интерфейс Ethernet

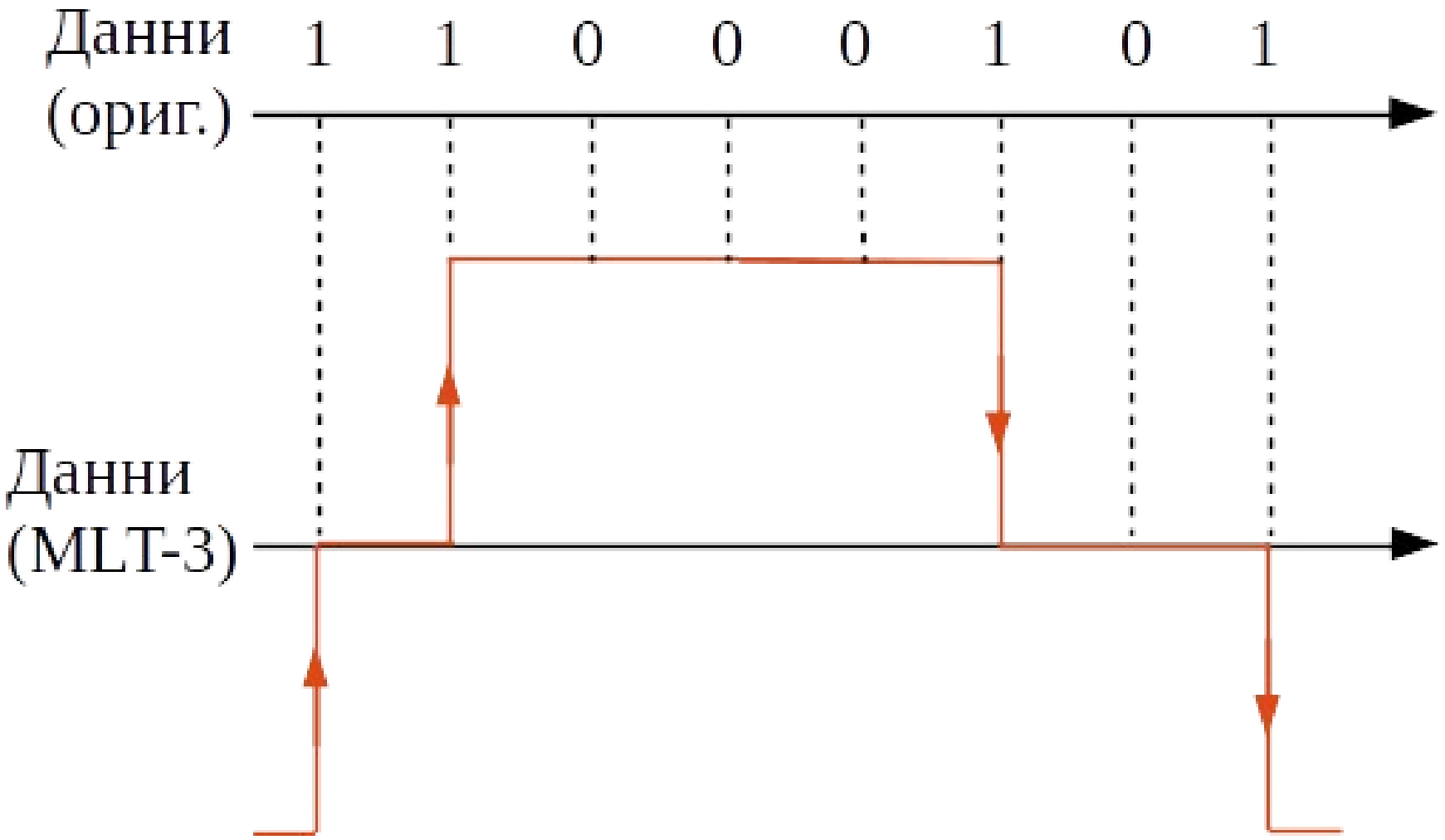
MLT-3 (Multi-Level Transmit) е кодиране на данни, при което се използват положителни (+) и отрицателни (-) напрежения, както и нула волта (0).

Логическата 1 се представя с преход.

Логическата нула се представя с липса на преход.

Аналогично на NRZI кодирането.

Интерфейс Ethernet



Интерфейс Ethernet

4B/5B (Block Coding) е кодиране на данни, при което 4 бита полезна информация се преобразуват в 5, за да може да се **добави служебна информация** и за да има **поне две логически единици** във всяко 5-битово поле.

Служебната информация се използва от протокола за обмен на данни.

Двете логически единици гарантират, че ще има повече преходи на сигнала (при NRZI+4B/5B комбинирано кодиране) за даден период, което помага за синхронизиране на приемането.

Интерфейс Ethernet

4B/5B кодиране [8]. Забележете, че комбинацията 00000 не съществува.

TABLE 3: 4B/5B ENCODING

Code	Value	Definition
0	11110	Data 0
1	01001	Data 1
2	10100	Data 2
3	10101	Data 3
4	01010	Data 4
5	01011	Data 5
6	01110	Data 6
7	01111	Data 7
8	10010	Data 8
9	10011	Data 9
A	10110	Data A
B	10111	Data B
C	11010	Data C
D	11011	Data D
E	11100	Data E
F	11101	Data F
I	11111	Idle
J	11000	SSD (Part 1)
K	10001	SSD (Part 2)
T	01101	ESD (Part 1)
R	00111	ESD (Part 2)
H	00100	Transmit Error

Интерфейс Ethernet

Добавянето на 5-ти бит означава, че 100 Mbit/s интерфейс трябва да работи на 125 Mbit/s. Това е еквивалентно на 125 MHz. При използването на **MLT-3** означава, **че фронтовете на сигнала стават 4**:

*0V → +

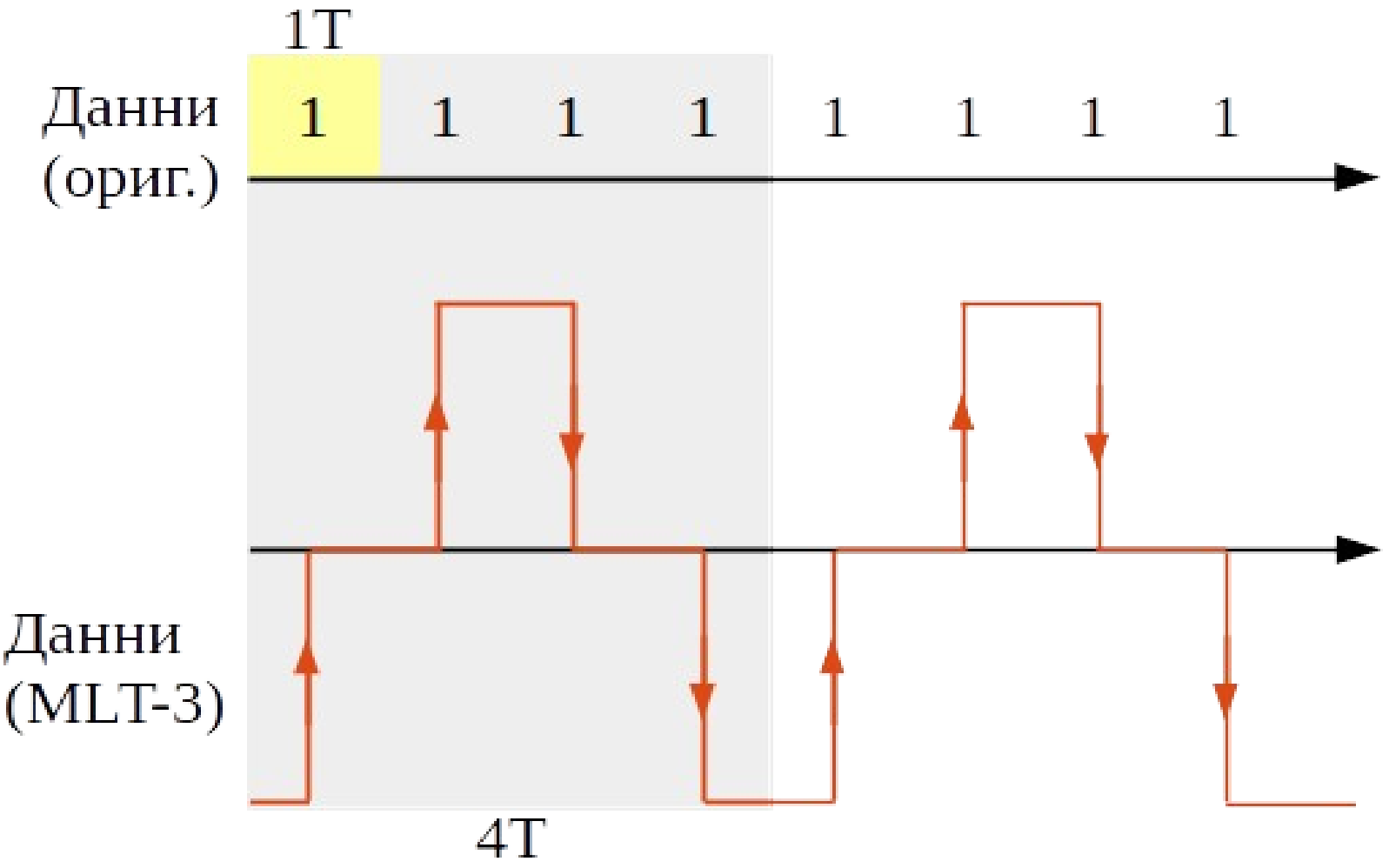
*+ → 0V

*0V→ -

*- → 0V

или с други думи можем да намалим честотата на сигнала 4 пъти и пак да предаваме със 125 Mbit/s. Оттук => **честотата при 100 Mbit/s версия е 31.25 MHz** [9].

Интерфейс Ethernet



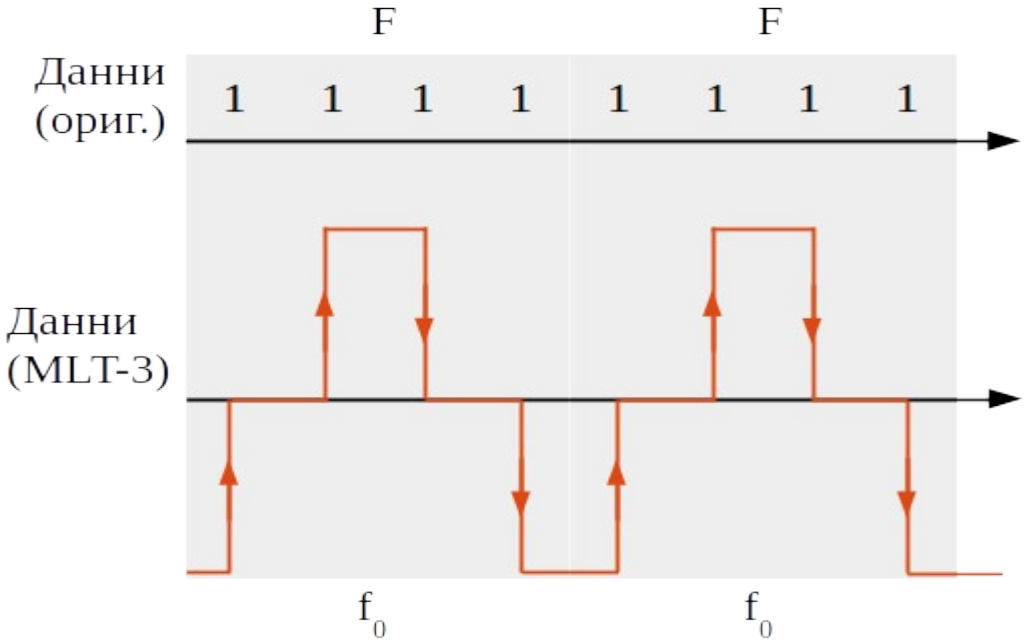
Интерфейс Ethernet

Разбъркване на данните (scrambling, spread spectrum) – данните се изменят по **псевдослучаен** закон, за да няма периодични сигнали по линията. Това помага за **шумоустойчивост** на предаването.

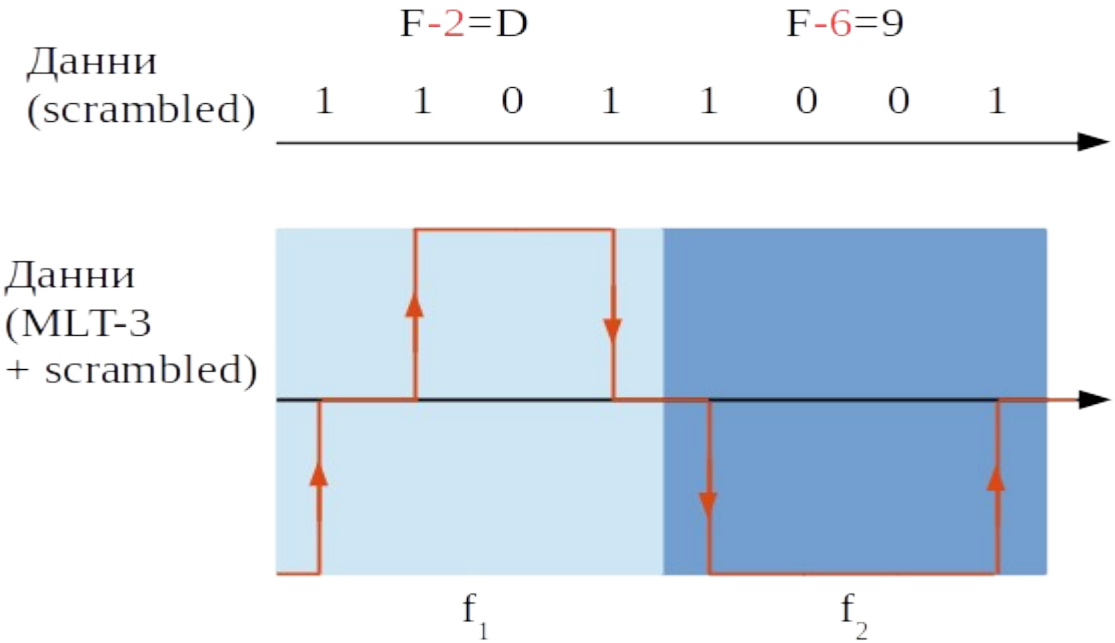
Приемникът **трябва да знае предварително** точно по какъв закон ще се изменят данните.

Интерфейс Ethernet

Пример – предаването на последователни единици при MLT-3 ще даде периодичен сигнал с честота f_0.



Ако към всяка тетрада изваждаме числата -2 и -6 това ще даде две нови честоти f_1 и f_2, различни от оригиналната.



Интерфейс Ethernet

На практика полиномите за трансформация са по-сложни. Ето как изглеждат те за 100 Mbit/s Ethernet [8]:

$$g_M(x) = 1 + x^{13} + x^{33} \quad \text{for master}$$

$$g_M(x) = 1 + x^{20} + x^{33} \quad \text{for slave}$$

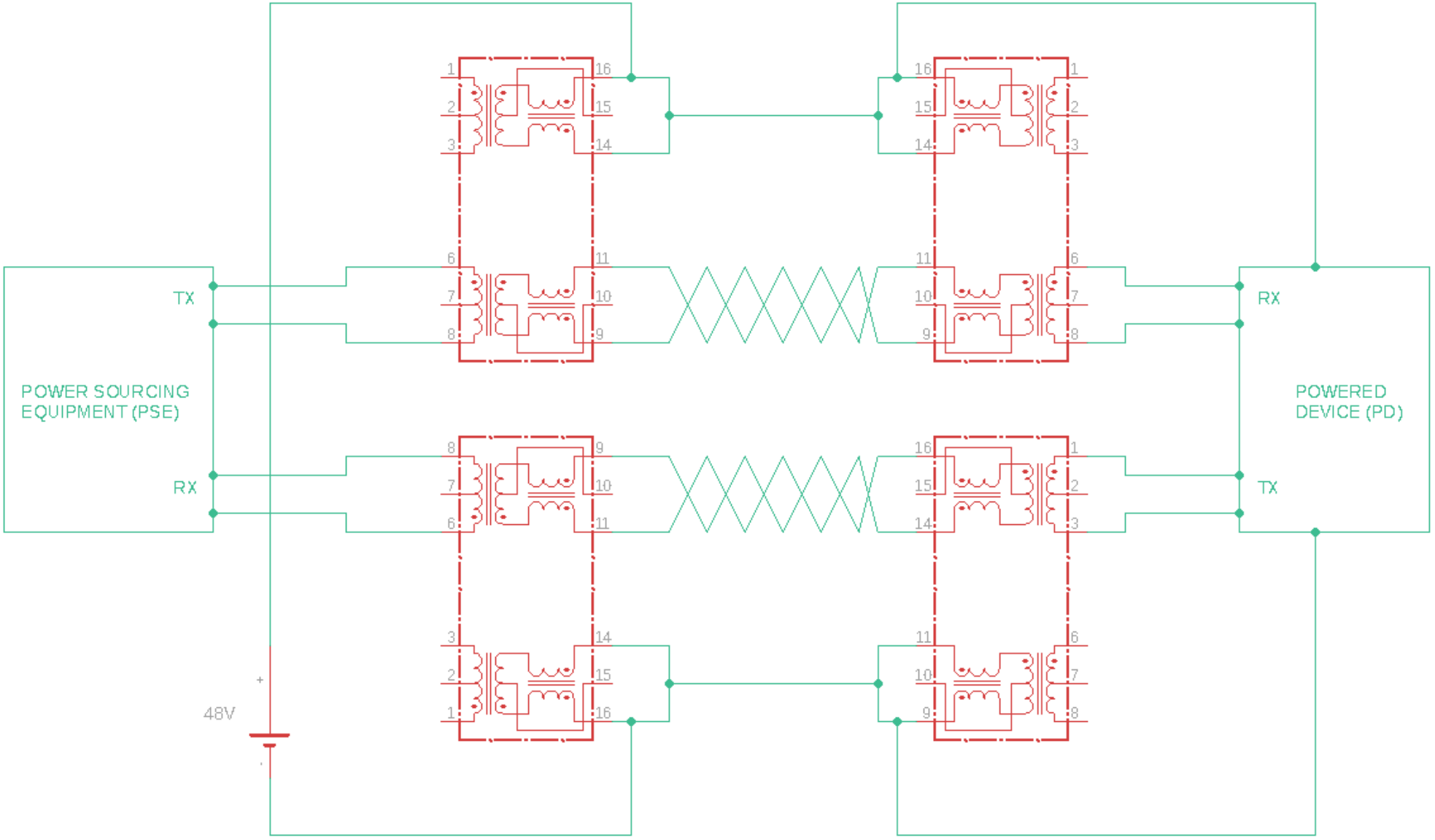
Интерфейс Ethernet

Пренос на мощност (PoE, **P**ower **O**ver **E**thernet) – по интерфейсният кабел на Ethernet могат да се захранват отдалечени устройства (напр. суичове и повторители, IP камери и т.н.).

Има два варианта това да стане:
*При 10/100 - когато се използват само 2 диференциални двойки, другите **две свободни** се окъсяват помежду си и едната е положителния извод, а другата е отрицателния на 48 V постоянно напрежение.
*При 10/100/1000 – диференциалните сигнали се отместват по постоянен ток и това **отместване се използва за захранване**.

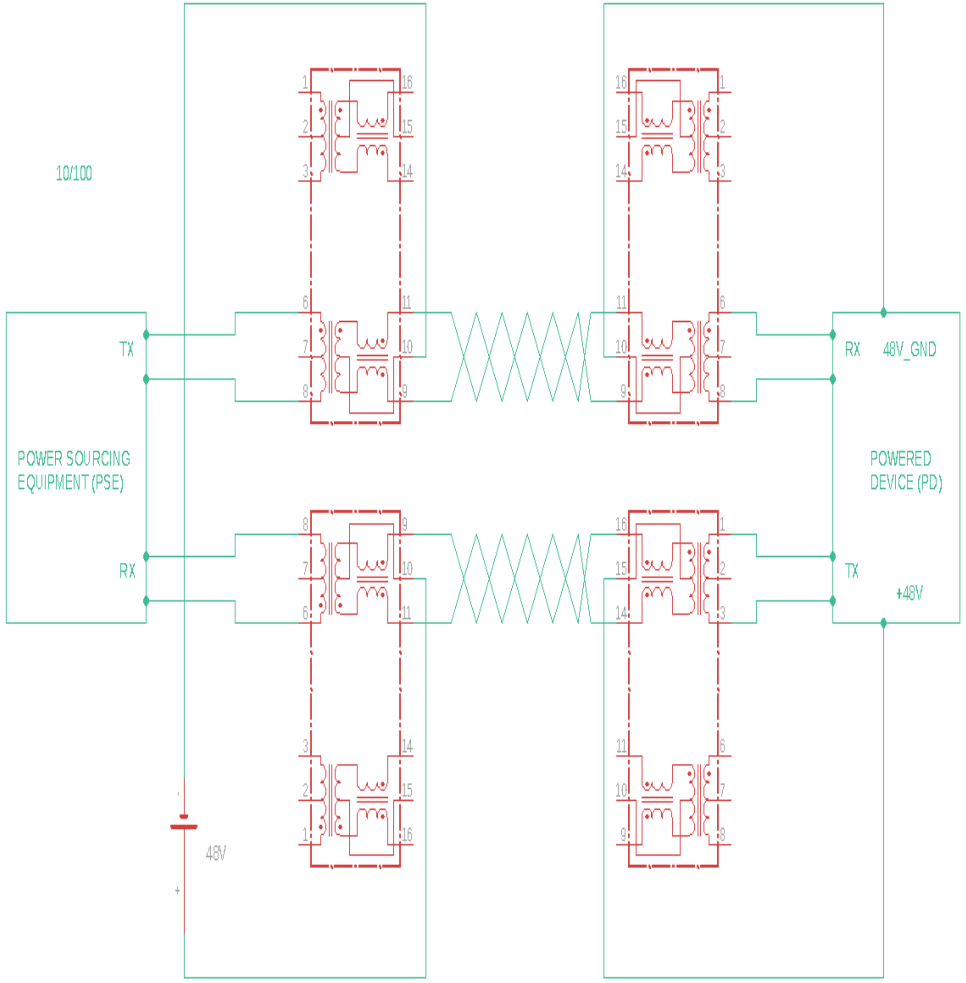
Интерфейс Ethernet

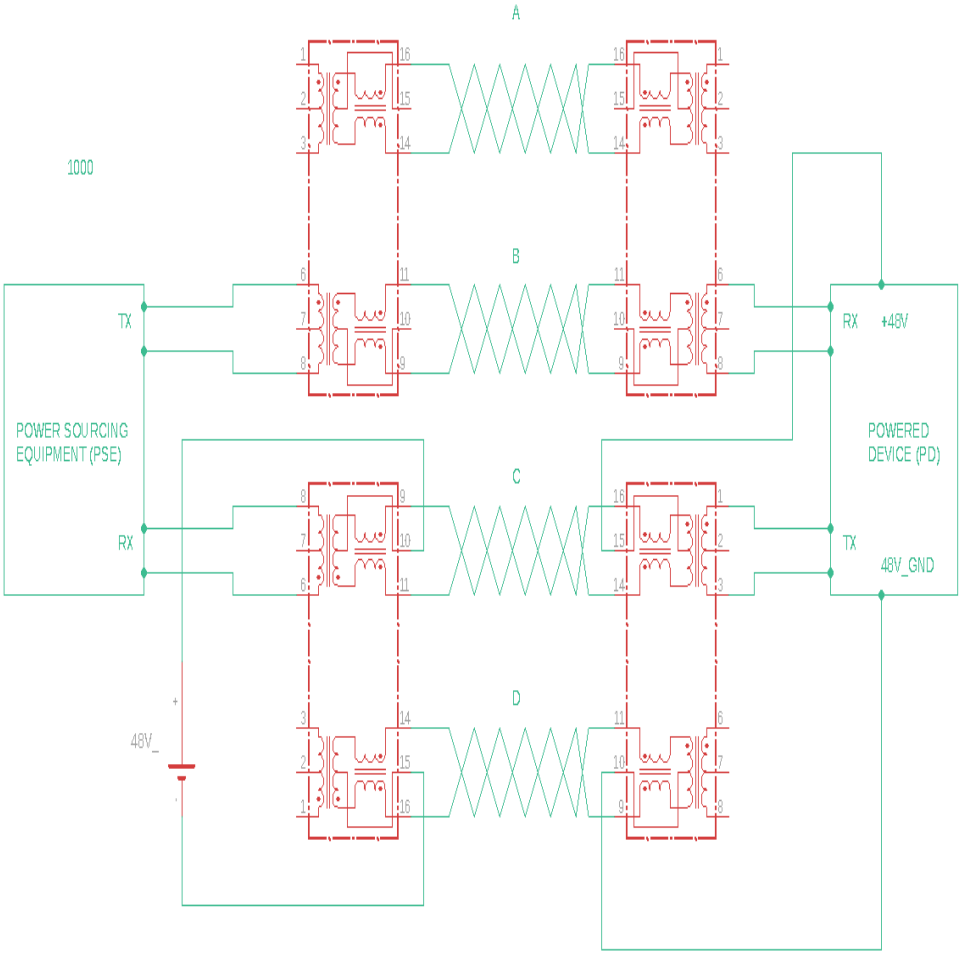
PoE по 2 свободни диференциални двойки (при 10/100):



Интерфейс Ethernet

PoE по даннови диференциални двойки (при 10/100/1000):





Интерфейс Ethernet

При едновременно предаване на захранване и данни, захранването не може да се предаде към трансийвърите, защото има импулсен трансформатор, който спира постоянния ток, а пуска “променливия” - данните.

Напреженията в източника трябва да са от 44 ÷ 57 V, а в отдалечения приемник от 37 ÷ 57 V @ 100 m разстояние.

Мощността на източника варира в различните версии от 15 ÷ 100 W (като сметнем загубите в кабела, приемникът трябва да получи 13 ÷ 71 W).

Интерфейс Ethernet

MII интерфейс (**M**edia **I**ndependent **I**nterface) – трябва да направи връзка между PHY и MAC слоевете. Това е паралелен, 4-битов, синхронен интерфейс, който се използва при 10/100.

Предаването на данни е **пълен дуплекс**, затова всъщност се използват 8 проводника + служебни.

Максимална дължина на проводниците – 50 см.

Скорост на данните по интерфейсът е ¼ от скоростта по интерфейса (2.5 Mbit/s → 10 Mbit/s, 25 Mbit/s →100 Mbit/s).

Интерфейс Ethernet

MII интерфейс (**M**edia **I**ndependent **I**nterface) – трябва да направи връзка между PHY и MAC слоевете. Това е паралелен, 4-битов, синхронен интерфейс, който се използва при 10/100.

Предаването на данни е **пълен дуплекс**, затова всъщност се използват 8 проводника + служебни.

Максимална дължина на проводниците – 50 см.

Скорост на данните по интерфейсът е ¼ от скоростта по интерфейса (2.5 Mbit/s → 10 Mbit/s, 25 Mbit/s →100 Mbit/s).

Интерфейс Ethernet

OSI модел за предаване на данни в Интернет е показан вдясно. Ethernet стандартът покрива последните два слоя и изгражда негови си три слоя:

*LLC (**L**ogical **L**ink **C**ontrol) – добавя контролна информация в даден IPv4 пакет. Софтуерен слой – драйверът за LAN картата.
*MAC (**M**edia **A**ccess **C**ontroller) – добавя адресиране, разделяне на фреймове, открива грешки. Хардуерен слой – представлява модул на μCU.
*PHY (**PHY**sical) – кодиране, разбъркване, формиране на електрически сигнали.

Интерфейс Ethernet

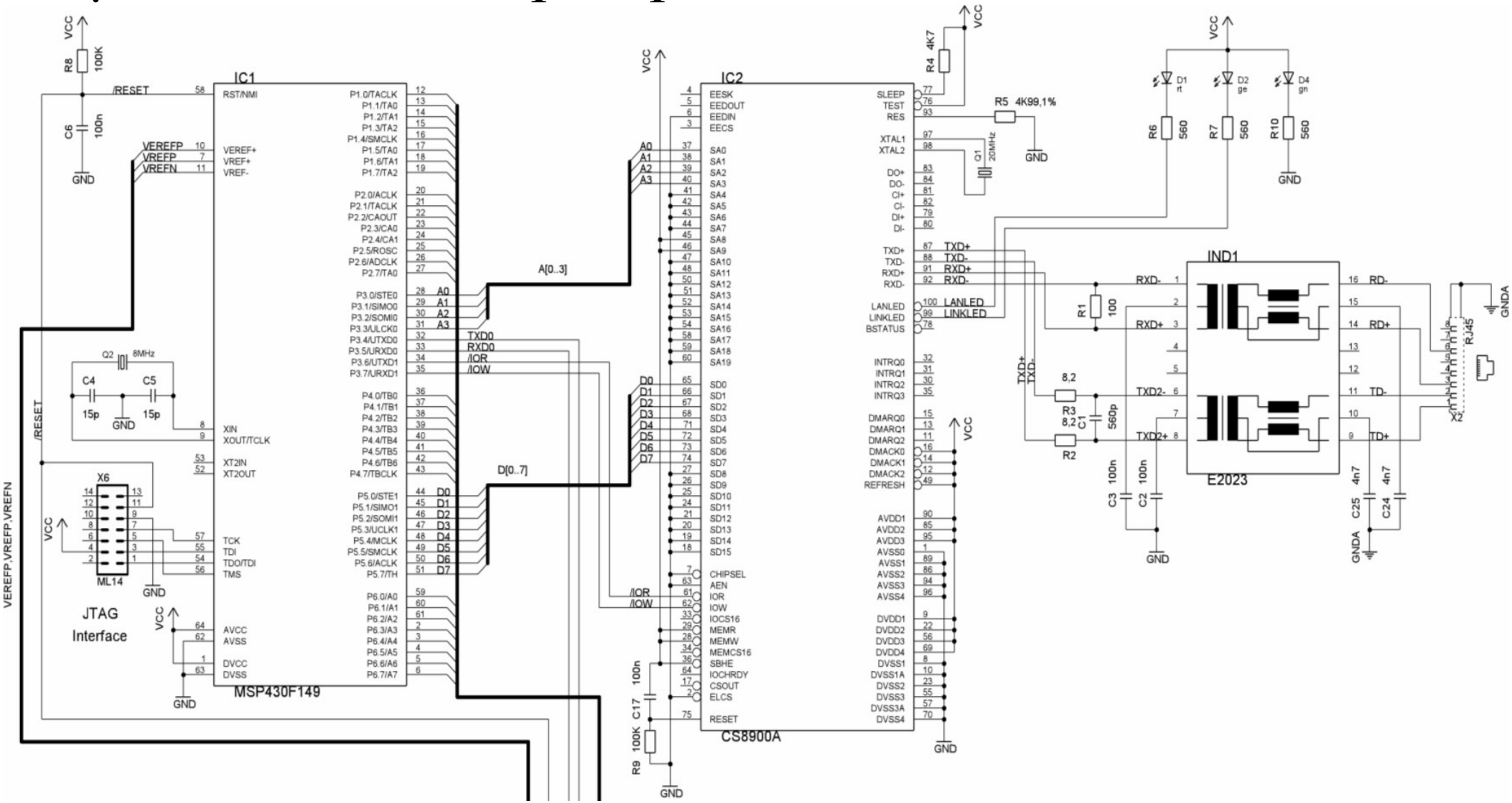
Исторически, μCU които нямаха Ethernet можеха да бъдат свързани чрез специални за целта чипове, наречени LAN контролери (напр. Ti CS8900A) + импулсен трансформатор.

След това μCU имаха MAC, но нямаха трансийвъри. Тогава се свързваха MII чипове + импулсен трансформатор.

Съвременните μCU решения имат MAC и PHY интегрирани, а външно се свързва импулсен трансформатор, обикновено интегриран с RJ45 куплунга.

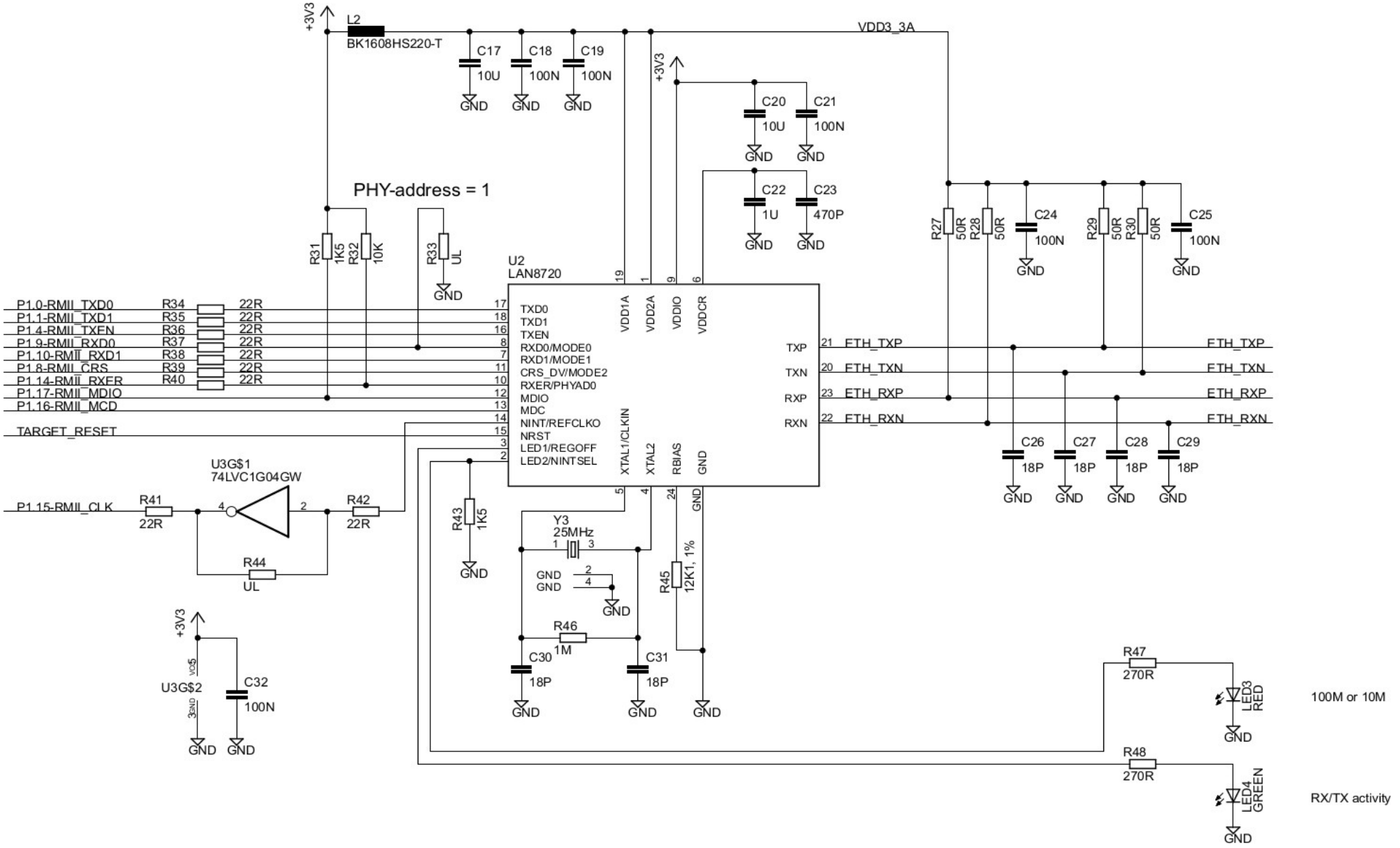
Интерфейс Ethernet

μCU + LAN контролер:



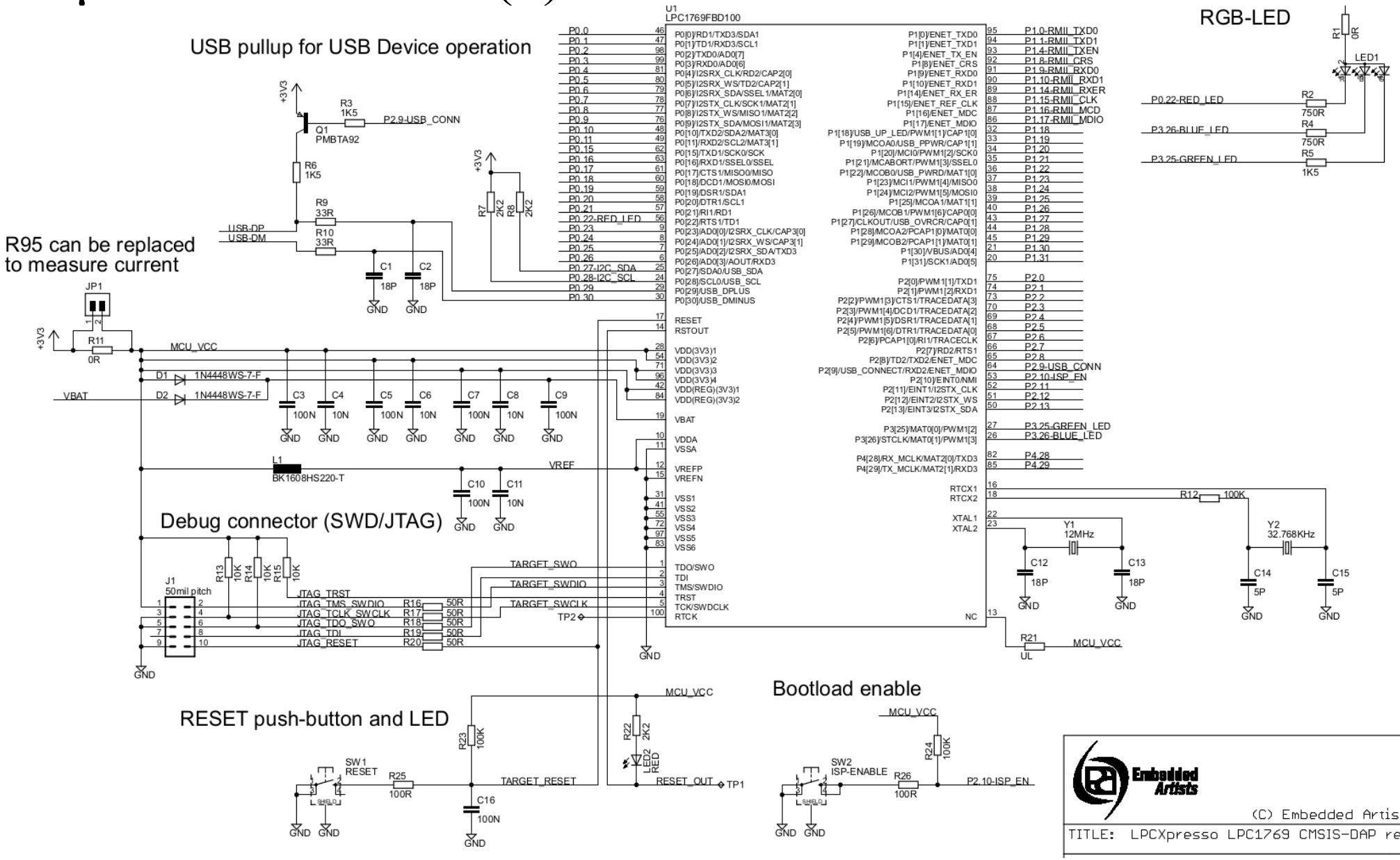
Интерфейс Ethernet

μCU + PHY чип (1):



Интерфейс Ethernet

µCU + PHY чип (2):

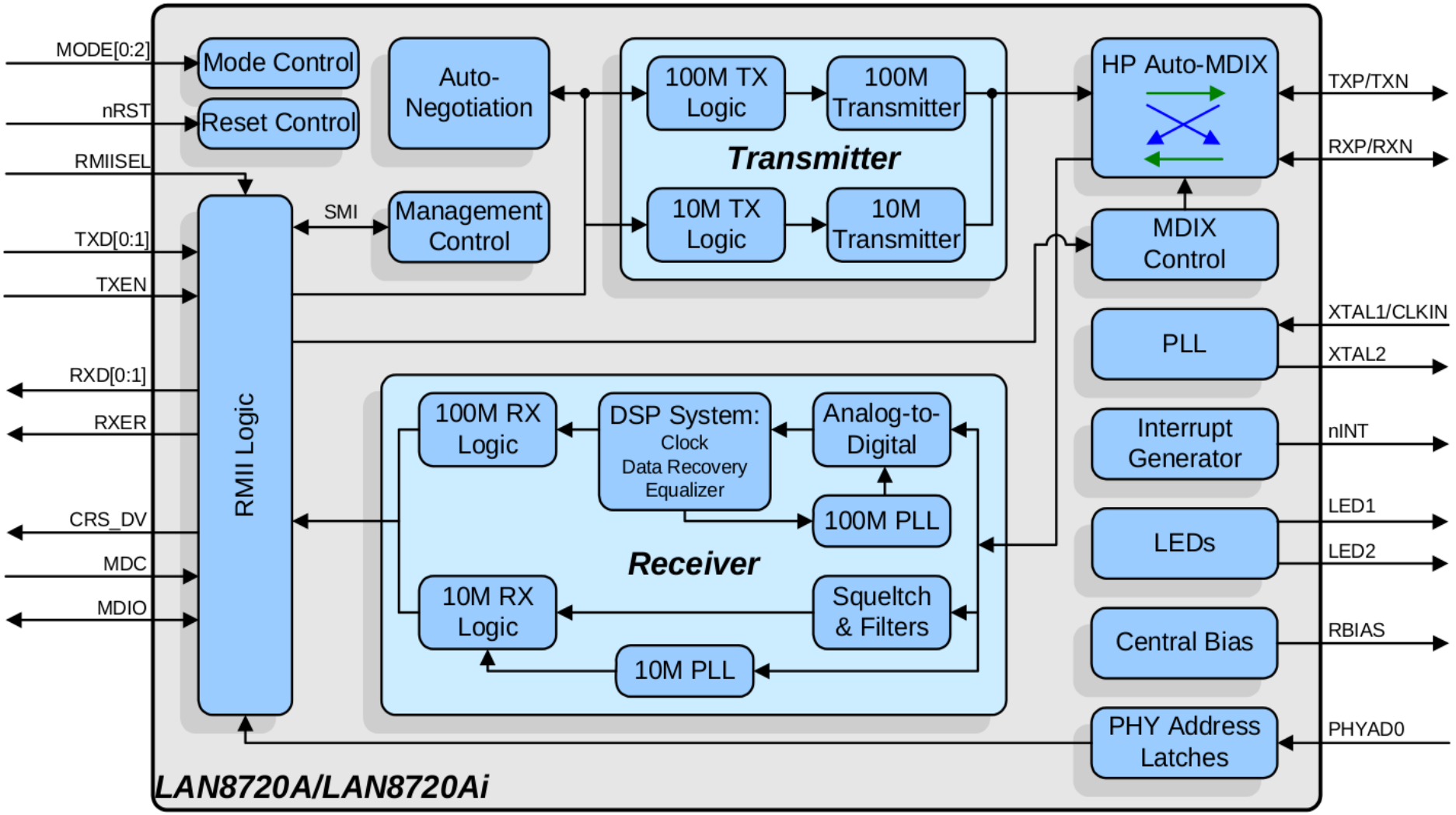


Интерфейс Ethernet

Вътрешна структура на PHY чип (LAN8720):

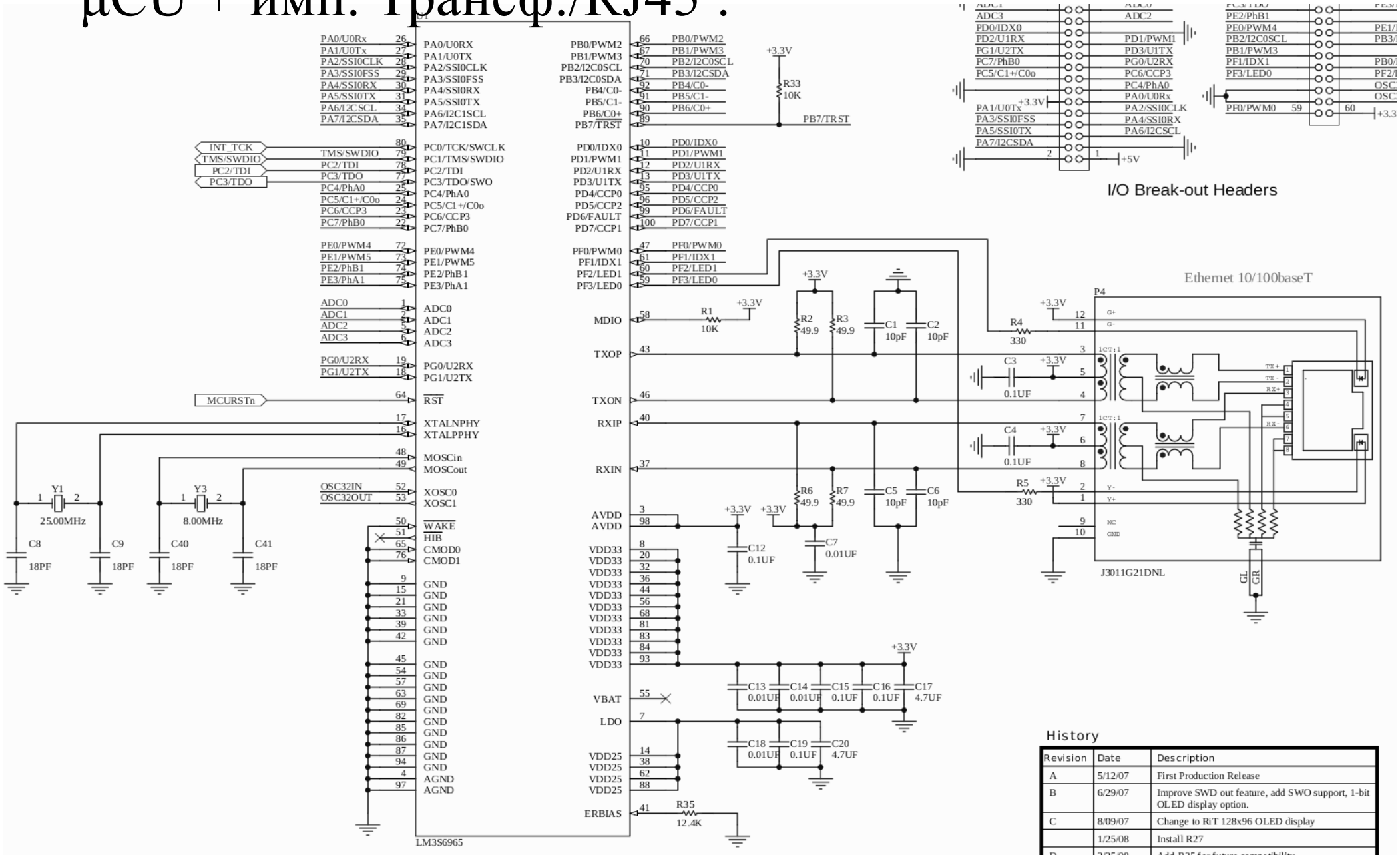
*RMII (Reduced MII) – 2-битов вариант на MII

FIGURE 1-2: ARCHITECTURAL OVERVIEW



Интерфейс Ethernet

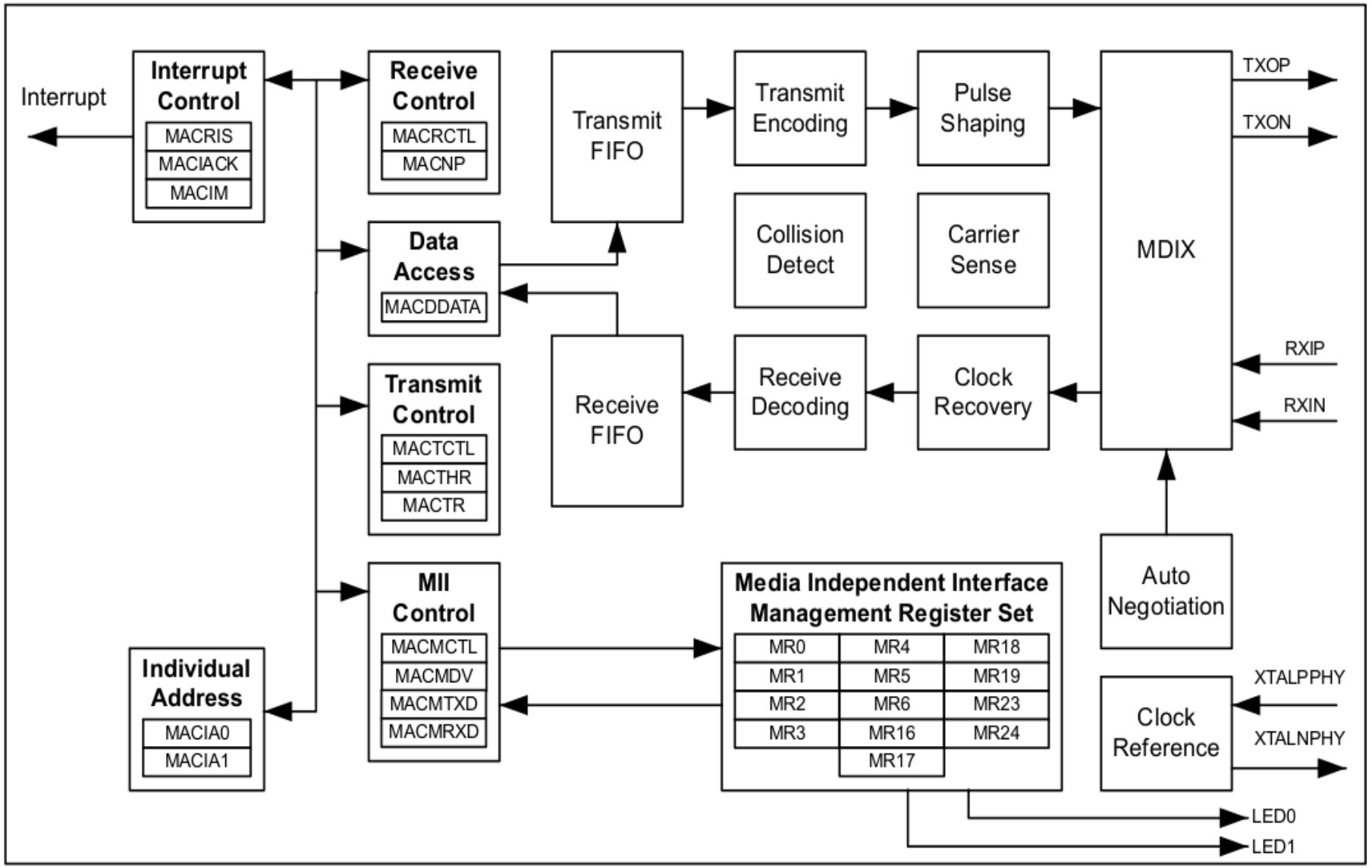
μCU + имп. Трансф./RJ45 :



Интерфейс Ethernet

Вътрешна структува на Етернет модул на LM3S6965

Figure 15-2. Ethernet Controller Block Diagram



Интерфейс Ethernet

Примерна инициализация на Texas Instruments LM3S6965 и lwIP библиотека:

```
int main(void) {
    unsigned long ulUser0, ulUser1;
    unsigned char pucMACArray[8];

    SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_8MHZ);

    // Enable and Reset the Ethernet Controller.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ETH);
    SysCtlPeripheralReset(SYSCTL_PERIPH_ETH);

    // Enable Port F for Ethernet LEDs.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeEthernetLED(GPIO_PORTF_BASE, GPIO_PIN_2 | GPIO_PIN_3);

    // Configure SysTick for a periodic interrupt.
    SysTickPeriodSet(SysCtlClockGet() / SYSTICKHZ);
    SysTickEnable();
    SysTickIntEnable();

    // Enable processor interrupts.
    IntMasterEnable();
```

Интерфейс Ethernet

```
// Configure the hardware MAC address for Ethernet Controller filtering of
// incoming packets.
// For the LM3S6965 Evaluation Kit, the MAC address will be stored in the
// non-volatile USER0 and USER1 registers.  These registers can be read
// using the FlashUserGet function, as illustrated below.

FlashUserGet(&ulUser0, &ulUser1);

if((ulUser0 == 0xffffffff) || (ulUser1 == 0xffffffff))
{
    // We should never get here.  This is an error if the MAC address has
    // not been programmed into the device.  Exit the program.
    while(1)
    {
    }
}
```

Интерфейс Ethernet

```
    // Convert the 24/24 split MAC address from NV ram into a 32/16 split MAC
    // address needed to program the hardware registers, then program the MAC
    // address into the Ethernet Controller registers.
    pucMACArray[0] = ((ulUser0 >>  0) & 0xff);
    pucMACArray[1] = ((ulUser0 >>  8) & 0xff);
    pucMACArray[2] = ((ulUser0 >> 16) & 0xff);
    pucMACArray[3] = ((ulUser1 >>  0) & 0xff);
    pucMACArray[4] = ((ulUser1 >>  8) & 0xff);
    pucMACArray[5] = ((ulUser1 >> 16) & 0xff);

    // Initialze the lwIP library, using DHCP.
    lwIPInit(pucMACArray, 0, 0, 0, IPADDR_USE_DHCP);

    // Initialize a sample httpd server.
    httpd_init();

    // Set the interrupt priorities.  We set the SysTick interrupt to a higher
    // priority than the Ethernet interrupt to ensure that the file system
    // tick is processed if SysTick occurs while the Ethernet handler is being
    // processed.  This is very likely since all the TCP/IP and HTTP work is
    // done in the context of the Ethernet interrupt.
    IntPriorityGroupingSet(4);
    IntPrioritySet(INT_ETH, ETHERNET_INT_PRIORITY);
    IntPrioritySet(FAULT_SYSTICK, SYSTICK_INT_PRIORITY);

    while(1){  }
}
```

Интерфейс Ethernet

Библиотеката lwIP използва драйверът на Texas Instruments (StellarisWare) за достъп до хардуера на конкретния микроконтролер (макросът HWREG достъпва регистър от Ethernet модула, т.е. volatile unsigned long *reg = (volatile unsigned long *)0x10003824):

```
long EthernetPacketPut
(unsigned long ulBase, unsigned char *pucBuf, long lBufLen){
    // Check the arguments.
    ASSERT(ulBase == ETH_BASE);
    ASSERT(pucBuf != 0);
    ASSERT(lBufLen > 0);

    // Wait for current packet (if any) to complete.
    while(HWREG(ulBase + MAC_O_TR) & MAC_TR_NEWTX)
    {
    }

    // Send the packet and return.
    return(EthernetPacketPutInternal(ulBase, pucBuf, lBufLen));
}
```

Интерфейс Ethernet

```
long EthernetPacketGet
(unsigned long ulBase, unsigned char *pucBuf, long lBufLen){
    // Check the arguments.
    ASSERT(ulBase == ETH_BASE);
    ASSERT(pucBuf != 0);
    ASSERT(lBufLen > 0);

    // Wait for a packet to become available
    while((HWREG(ulBase + MAC_O_NP) & MAC_NP_NPR_M) == 0)
    {
    }

    // Read the packet
    return(EthernetPacketGetInternal(ulBase, pucBuf, lBufLen));
}
```

Литература

[1] Г. Михов, “Цифрова схемотехника”, ТУ-София, 1999.
[2] J. Axelson, “Serial Port Complete”, Lakeview Research, 2007.
[3]P. Gaspar, A. Santo, B. Ribeiro, H. Santos, “Communications USCI Module”, MSP430 Teaching Materials, presentation, Texas Instruments, 2009.
[4] J.Axelson, “USB Complete”, Lakeview Research, 2009.
[5] E. Murphy, P. Fitzgerald, “Switching in USB Consumer Applications”, Analog Dialogue 40-01, Analog Devices, 2006.
[6] М. Митев, “Микропроцесорна схемотехника”, записки на лекции, 2021.
[7] AN1120, “Ethernet Theorgy of Operation”, Microchip application note, 2008.
[8] R. Neuhaus, “A Beginner's Guide to Ethernet 802.3”, Analog Devices, Engineer-to-Engineer Note EE-269, 2005.
[9]https://courses.cs.washington.edu/courses/cse461/17au/lectures/ethernet.pdf