2004-2006m. Bendrojo programavimo dokumento 2 prioriteto 5 priemonė
„Žmogiškųjų išteklių kokybės gerinimas mokslinių tyrimų ir inovacijų srityje“

Projektas

Fizinių mokslų II ir III studijų pakopų pertvarka, jas pritaikant prioritetinių MTEP sričių vystymui

Projekto numeris BPD2004-ESF-2.5.0-03-05/0012

Mikrovaldikliai elektroninėse grandinėse

Mokymo priemonė

Parengė: Vytautas Jonkus

Vilnius 2006 – 2008 m.

Projektas “Fizinių mokslų II ir III studijų pakopų pertvarka, jas pritaikant prioritetinių MTEP sričių vystymui”, remiamas ES Struktūrinio fondo lėšomis, įgyvendinant 2004-2006 metų bendrojo programavimo dokumento 2.5 priemonę „Žmogiškųjų išteklių kokybės gerinimas mokslinių tyrimų ir inovacijų srityje“.

Turinys

Sutartiniai žymėjimai

Žymėjimai tekste:

registrų ir bitų vardai: **PORTA**, **T1CON**, **RB0**, **RB1**, ...

sutrumpinimai ir kiti vardai: ***PIC18***, ***DSP***, ***Timer1***, ...

programos tekstas, C ar asemblerio instrukcijos: `TBLRD*`, `for(;;)`, ...

Žymėjimai paveiksluose. Skirtingos spalvos reiškia:

registrų vardai: **PORTA**, **T1CON**, **FSR0H**, ...

bitų vardai: **RB0**, **RB1**, **T1ON**, **ADON**, ...

mikrovaldiklio išvadų vardai: **RA0**, **RA1**, ..., **RB0**, **RB1**, ..., **RC0**, **RC1**, ...

Programų tekstų ypatumai:

skaičiai dešimtainėje sistemoje: 12, 56, ...

skaičiai šešioliktainėje sistemoje (C kalbos standartas): 0xA3, 0x78, ...

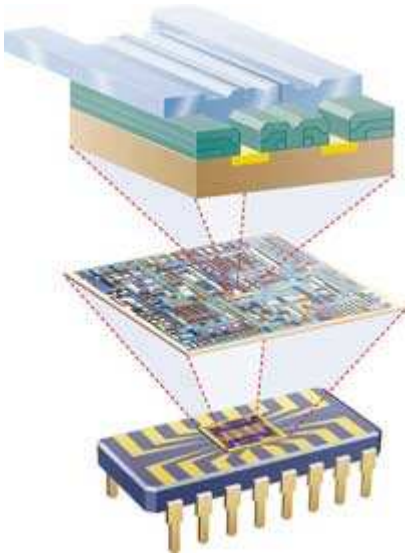
skaičiai dvejetainėje sistemoje (tinka tik Microchip C kompiliatoriui): 0b10101100, ...

Įvadas

Mokymo priemonė yra skirta VU fizikos fakulteto magistratūros studijų studentams, klausantiems to paties pavadinimo kurso. Laikoma, kad studentai turi elektronikos ir programavimo pagrindus. Dėstomas kursas yra skirtas studentus supažindinti su mikrovaldiklių praktinio taikymo ypatybėmis, studentai turi atlikti daug praktinių užduočių, todėl ir mokymo priemonė yra orentuota į tai, ką reikia žinoti, kad reali elektroninė schema su mikrovaldikliu sėkmingai dirbtų. Tikiuosi, kad mokymo priemonėje pateikti programų pavyzdžiai pagelbės studentams atliekant praktines užduotis.

Dėstomas kursas yra paremtas praktinėmis užduotimis, o praktinės užduotys yra atliekamos su maketais ir konkrečiais mikrovaldiklių modeliais. Šio kurso praktinės užduotys ir mokymo priemonė yra orientuoti į Microchip ***PIC18*** šeimos mikrovaldiklius, programavimo kalba – ***C***. Tokį pasirinkimą lėmė šių mikrovaldiklių populiarumas, dokumentacijos aiškumas (norint parašyti kad ir paprastą programą mikrovaldikliui, tenka nemažai perskaityti dokumentacijos) ir nemokamas ***C*** kompiliatorius. Mokymo priemonėje aiškinant mikrovaldiklio vidinių elektroninių grandinių veikimą, remtasi ***PIC18*** šeimos mikrovaldikliais, tačiau tarp skirtingų mikrovaldiklių yra daug panašumų, išsiaiškinus vieno mikrovaldiklio veikimą nesunku perprasti kitą.

Apie mikroprocesorius, mikrovaldiklius ir DSP. Įžengus žmonijai į XXI amžių kasdieninė buitis be gudrių elektroninių įrenginių yra visiškai neįsivaizduojama, nors tas jų „gudrumas“ kartais atrodo kvailai. Turint minutėlę laisvo laiko galime patyrinėti kur tas „gudrumas“ slepiasi. Paimkime kurį nors „gudruolį“ kuris yra po ranką, o pas visuos tikrai yra mobilusis telefonas, ir ilgą atsuktuvą. Daug ką pamatysite viduje, tame tarpe ir didelį „blyną“ su daugybe mažų kojyčių. Tai jis kartais sumano per pokalbio vidurį pailsėti ir tenka gražiai paprašyti, kad vėl imtųsi darbo. Tai – specialios paskirties mikroprocesorius.



Jei yra specialios paskirties mikroprocesoriai, kuo tuomet jie skiriasi nuo „ne specialių“? Pagrindinė mikroprocesoriaus užduotis – vykdyti komandas: paimti komandą iš atminties, atlikti veiksmus ir rezultatą vėl padėti į atmintį. Be šios pagrindinės funkcijos mikroprocesoriuje gali būti elektroninės grandinės, kurių funkcijos nesusijusios su pagrindiniu mikroprocesoriaus darbu – vykdyti komandas. Šiais laikais, kai galima už pažastų kiekvieną atomą paimti, nunešti ir padėti ten kur reikia, nesudaro sunkumų į tą patį kristalą, kuriame yra pagrindinės mikroprocesoriaus grandinės, integruoti ir analoginį-skaitmeninį keitiklį ar dar ką nors. Tik ar to reikia?

Galima įžiūrėti tris pagrindinius reikalavimus mikroprocesoriui: jis turi būti spartus, jame turi būti integruotos visiems gyvenimo atvejams reikalingos grandinės ir jis turi būti pigus. Tai – vienas kitam prieštaraujantys reikalavimai. Tenka ieškoti kompromiso: jei mikroprocesorius bus labai spartus, tai jo kaina bus didelė arba jei integruosime daug papildomų grandinių taps technologiškai sunku išgauti didelę spartą ir t.t. Vienas polius – ne itin spartūs mikroprocesoriai, tačiau turintys daugybę integruotų papildomų grandinių, norint panaudoti juos tam tikrai užduočiai reikalingas minimalus papildomų elementų skaičius. Tai – mikrovaldikliai (angl. microcontroller). Kitas polius – labai greiti, gerai pritaikyti atlikti skaičiavimus procesoriai, tačiau nėra turtingi papildomomis elektroninėmis grandinėmis, tokį mikroprocesorių jungiant į elektroninę grandine tenka schemoje panaudoti daug išorinių

elementų. Tai – ***DSP*** (angl. Digital Signal Processor) procesoriai. Egzistuoja daugybė mikroprocesorių gamintojų ir dar didesnė mikroprocesorių modelių įvairovė, kurių galimybės užima tarpinę padėti tarp dviejų minėtų kraštutinumų.

Yra sukurta daug mikrovaldiklių archtektūrų ir daygybė gamintojų, Ši įvairovė, kad ir neišsamiai, atsispindi 1 lentelėje. Dalis mikrovaldiklių archtektūrų yra įdiegta tik į tam tikro gamintojo mikrovaldiklių šeimą. Yra architektūrų, kurios buvo pripažintos daugelio gamintojų ir, nusipirkę licenziją, gamintojai gamina ir siūlo savo mikrovaldiklius, pagamintus pagal pripažintą architektūrą. Kelių architektūrų įsigalėjimas mikrovaldiklių gamintojų tarpe yra naudingas reiškinys: tos pačios architektūros mikrovaldikliams bus ta pati komandų sistema, o tai reiškia, kad tiks tos pačios programavimo priemonės (kompiliatoriai ir kt.); bus mažiau klaidų procesoriaus branduolyje, nes, kurdamas naują naują architektūrą, gamintojas įvelia gana daug klaidų elektroninėje dalyje. Iš tokių populiarumą sulaukusių mikrovaldiklių architektūrų reiktų išskirti ARM (angl. Advanced RISC Machine) architektūrą. Iš visų 32 bitų mikrovaldiklių, kurie yra pagaminami, 75% iš jų turi ARM architektūrą. Ją sukūrusi bendrovė Acorn Computers, vėliau įsteigusi dukterinę ARM bendrovę, pati iš viso negamina mikrovaldiklių, tačiau įvairiems gamintojams parduoda licenzijas gaminti mikrovaldiklius pagal šią perspektyvią architektūrą.

1 lentelė. Mikrovaldiklių architektūrų ir gamintojų įvairovė.

Architektūra	Mikrovaldiklis	Gamintojas
ARM (ARM7, ARM7TDMI, StrongARM, ARM9TDMI, ARM9E, Cortex, ...)	LPC2xxx, AT83xxx, AT87xxx, CC2xxx, ML67xxx, S3C4xxx, LH754xx, ST10Fxxx, STR7xx, STR9xx	Analog Devices, Cirrus Logic, Atmel, Freescale, Fujitsu, Infineon, Philips, STMicroelectronics, OKI, Samsung, Sharp, ...
AVR	ATMega8xxx, ...	Atmel
C166, XC166, XE166	C16x, XC16x, XE16x	Infineon
ColdFire	MCF5xxx, ...	Freescale
HCS12	68HC9xx, MC9S12xxx	Freescale
H8S, H8/300H		Renesas
M16C, M8C, M32C, R32C		Renesas
M Core	MC68xxx, MMC2xxx	Freescale
MIPS32		AMD, Microchip, NEC, LSI Logic, Sunplus, Broadcom, Toshiba
MK5		Sharp
PIC10, PIC12, PIC16, PIC18, PIC24	PIC16Fxxx, PIC18Fxxx, ...	Microchip
SuperH		Hitachi
SAM8	S3Cxxxx, S3Fxxxx	Samsung
V850		NEC
78K0	QB78K0Sxxx, ...	NEC
8051, 8052		Analog Devices, Atmel, Chipcon, Cypress, Honeywell, Infineon, Intel, Philips, OKI, STMicroelectronics, Sharp, Triscend, ...
...	**...**	**...**

Apie *PIC18* mikrovaldiklių šeimą. Šios šeimos mikrovaldiklius gamina Microchip kompaniją. ***PIC18*** šeima „išaugo“ iš ankstesnės ***PIC16*** šeimos, buvo patobulinta, tačiau paveldėjo ir blogų architektūros savybių. ***PIC18*** mikrovaldiklių šeima – tai 8 bitų

mikroprocesorius, duomenų ir programos atmintis yra atskira, pasižymi ypač taupiu elektros energijos vartojimu, pilnai atitinka principą, kad viskas ko gali prireikti, turi būti integruota mikrovaldiklio viduje. Microchip kompanijos mikrovaldiklių dokumentacija yra labai išsami ir aiški (tai ne toks jau dažnas reiškinys mikrovaldiklių gamintojų tarpe), pateikia daug pavyzdžių (ypač naudingi sudėtingi atvejai: ***USB*** sąsajos ir ***TCP/IP*** bibliotekos) ir nemokamą ***C*** kalbos kompiliatorių (žinoma, su kai kuriais apribojimais). Ne paslaptis, kad mikrovaldiklių gamintojai projektuodami įvelia klaidų mikrovaldiklio elektroninėse grandinėse, tuomet jos veikia ne taip kaip tikimasi arba iš viso neveikia. Save gerbiantis gamintojas viešai skelbia šias klaidas ir būdus kaip jas apeiti. Ne išimtis ir ***PIC18*** šeimos mikrovaldikliai, – čia klaidų yra nemažai, tačiau informaciją apie jas galima rasti gamintojo puslapyje. ***PIC18*** šeimos mikrovaldikliai yra gana dažnai atnaujinami, net tiksliai to paties pavadinimo mikrokrovaldikliai gali skirtis branduolio versija (mikrovaldiklio branduolio versiją galima nuskaityti su programatoriumi).

Artimiausias ***PIC18*** šeimos konkurentas pagal populiarumą ir galimybes yra Atmel kompanijos gaminama ***AVR*** mikrovaldiklių šeima. Šių dviejų gamintojų mikrovaldikliai dažnai vadinami „studentiškais“, nes yra itin populiarūs elektronikos mėgėjų tarpe ir yra labai tinkami susipažinimui su mikrovaldiklių architektūra ir taikymais. Tradiciškai Microchip mikrovaldikliai laikomi geriau tinkamais iš baterijų maitinamiems prietaisams ir pasižymintys didesniu atsparumu elektriniams trukdžiams schemoje, o Atmel mikrovaldikliai turi geresnę atminties architektūrą ir yra šiek tiek spartesni.

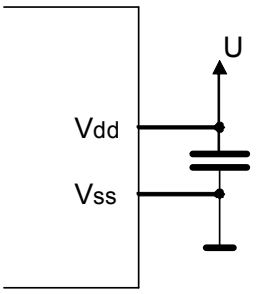
PIC18 šeima turi daug modelių. Jų vardų sudarymo principas yra toks: su vienkartinio programavimo atmintimi modeliai žymimi ***PIC18Cxxx*** arba ***PIC18Cxxxx*** (kur ***xxx*** yra skaičius), o su elektriškai perprogramuojama atmintimi (***EEPROM***) žymimi ***PIC18Fxxx*** arba ***PIC18Fxxxx***, jų maitinimo įtampa yra 5V. Jei mikrovaldiklio maitinimo įtampų diapazonas yra nuo 2.7V iki 5V, tai jų pavadinimas būtų ***PIC18LFxxx*** arba ***PIC18LFxxxx***. Trys skaitmenys po ***F*** raidės buvo naudojami senesniuose modeliuose, keturi skaitmenys yra naudojami naujesniuose. ***PIC18*** šeimos mikrovaldiklių taktinis dažnis gali būti 0 ... 40 MHz, išimtis yra modeliai su integruota ***USB*** sąsaja, jų taktinis dažnis gali būti iki 48 MHz. Viena komanda yra įvykdoma per 4 taktus, tik valdymo perdavimo, paprogramės iškvietimo ir grįžimo iš paprogramės komandos yra vykdomos 8 taktus. Mikrovaldikliai turi ***EEPROM*** atmintį programai, atskirą ***EEPROM*** atmintį – duomenis, ***RAM*** atmintį ir konfigūracinių bitų ***EEPROM*** atmintį. Konfigūraciniai bitai yra įrašomi programavimo su programatoriumi metu, jie nustato koks bus taktinių impulsų šaltinis ir kitus gyvybiškai svarbius mikrovaldiklio darbui parametrus. Programa į mikrovaldiklius yra įrašoma su specialiu programatoriumi, programavimo metu mikrovaldikliui taktinių impulsų generatorius nėra reikalingas. Į programavimo būseną mikrovaldiklis pervedamas prijungus 12 V įtampą prie ***MCLR*** išvado, o programa perduodama per ***RB6*** ir ***RB7*** išvadus. 12 V šaltinio naudojimas programavimui gali būti nepatogus, todėl yra numatytas kitas būdas mikrovaldikliui pervesti į programavimo būseną: panaudojant ***RB5*** išvadą,– prie jo prijungus 5 V, mikrovaldiklis yra pervedamas į programavimo būseną. Žinoma, jei ***RB5*** yra naudojamas programavimui, jo kitiems tikslams negalima panaudoti, tačiau konfigūraciniuose bituose galima išjungti šią programavimo galimybę ir tuomet ***RB5*** tampa standartinės paskirties išvadu. Ką tik iš fabriko atvykusiuose „šviežiuose“ mikrovaldikliuose ***RB5*** paskirtis yra skirta programavimui.

1. Mikrovaldiklio jungimas į elektroninę schemą

1.1 Maitinimas

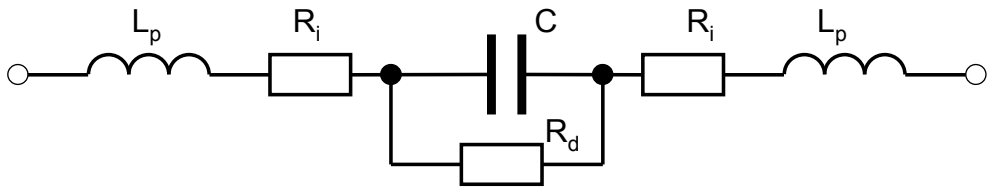
Mikrovaldiklio kaip ir bet kurio procesoriaus pagrindas yra sinchroniškai persijunginėjantys loginiai elementai, trigeriai ir t.t., o kurie savo ruožtu yra sudaryti iš tranzistorių, dirbančių kaip elektroninio rakto veikoje (tiesa, yra skaitmeninių technologijų, kur tranzistoriai dirba tiesinėje veikoje), t.y. tranzistorius yra įjungtas arba išjungtas. Tokios tranzistorių sistemos maitinimo ypatybė yra padidėjęs srovės poreikis tranzistoriams keičiant būseną, o mikroprocesoriuje visi tranzistoriai keičia būseną sinchroniškai. Jei mūsų mikrovaliklis dirba 40 MHz dažniu, tai kas 25 ns yra žymus suvartojamos srovės padidėjimas, o pati srovės padidėjimo trukmė yra trumpesnė. Jei maitinimo šaltinis nesugeba kompensuoti padidėjusios srovės poeikio, krenta maitinimo įtampa. O jei maitinimo įtampa nukrenta žemiau leistino lygio kad ir labai trumpam laiko tarpui, mikrovaldiklio darbas sutrika. Išvada – maitinimo šaltinis turi sugebėti kompensuoti trumpus srovės poreikio padidėjimus, o realybė – maitinimo šaltiniui ši užduotis yra neįmanoma. Maitinimo šaltinis sugeba kompensuoti žymiai mažesniu dažniu atsirandantį srovės suvartojimo pokytį. Visų pirma, vienas maitinimo šaltinis maitiną visą schemą, nuo šaltinio yra nuvesti ilgi takeliai, tiekiantys srovę daugeliui elektroninių lustų schemoje. Maitinimo grandinės didelis induktyvumas ir baigtinė signalo sklidimo trukmė nuo maitinimo šaltinio iki srovės vartotojo sąlygoja maitinimo šaltinio neįgalumą tokio poreikio akivaizdoje. Realūs impulsiniai maitinimo šaltiniai dirba apie 100 kHz dažniu ir yra pajėgūs kompensuoti periodinių srovės suvartojimo poreikių pokyčius geriausiu atveju iki 10 kHz.

Kondensatoriai maitinimo grandinėje. Mikrovaldiklis turi bent vieną, o dažniausiai ir kelis išvadus, prie kurių jungiama maitinimo įtampa. Kondensatoriai prie maitinimo grandinės turi būti jungiami kuo arčiau mikrovaldiklio išvadų. Šie kondensatoriai atlieka dvejopą funkciją: filtruoja kintamos įtampos svyravimus, ateinančius iš maitinimo šaltinio, ir veikia kaip elektrinio krūvio talpyklos, iš kurių yra kompensuojamas staigus srovės poreikis. Jau minėtame pavyzdyje, periodiškai kas 25 ns atsiranda srovės suvartojimo pikas, kurio trukmė dar trumpesnė. Esant tokiems trumpiems laiko intervalams tampa svarbūs kondensatoriaus paskirstyti parametrai (induktyvumas, ekvivalentinė varža), atstumas iki mikrovaldiklio išvadų, takelių spausdintinėje plokštėje išsidėstymas (takelių varža ir induktyvumas gali riboti krūvio iš kondensatoriaus atidavimo laiką). Kad paaiškėtų kaip ir kokius kondesatorius jungti prie maitinimo grandinės, tenka panagrinėti paskirstytus kondensatoriaus parametrus.



1 pav. Kondensatoriaus jungimas prie mikrovaldiklio maitinimo išvadų.

Realus kondensatoriaus. Kondensatorius aukšto dažnio grandinėje pasižymi ne vien talpa, bet ir induktyvumu, varža. Kaip seka iš kondensatoriaus ekvivalentinės schemos 2 paveiksle, kondensatorius turi induktyvumą L_p, kurį, pagrinde, sąlygoja kontaktų induktyvumas, nuoseklią varžą R_i, nuotekio varžą R_d, kurią salygoja nuostoliai dielektrike (priedo, R_d priklauso nuo įtampos pastoviojo sando). Kondensatorių gamintojai pateikia visų varžų suminę vertę, vadinamą ekvivalentine kondensatoriaus varža (angl. ESR – Equivalent Series Resistance). Kondensatoriui greitai atiduoti sukauptą krūvį trukdo varža R_i ir induktyvumas L_p. Varža R_i priklauso nuo kondensatoriaus elektrodų tipo ir gamybos technologijos, induktyvumą L_p daugiausia lemia kondensatoriaus išvadų geometriniai matmenys. Mažą ekvivalentinę važą turi keraminiai kondensatoriai, tačiau jie gaminami mažos talpos, nes didelės talpos keraminis kondensatorius užimtų didelį tūrį. Elektrolitiniai kondensatoriai su aliuminio ir tantalo elektrodais pasižymi didele talpa ir užima mažą tūrį, bet jų ekvivalentinė varža yra didesnė. Didesnę ekvivalentinę varžą turi aliuminio elektrolitiniai kondensatoriai, negana to, jų ekvivalentinė varža laikui bėgant stipriai didėja dėl elektrolito pokyčių.

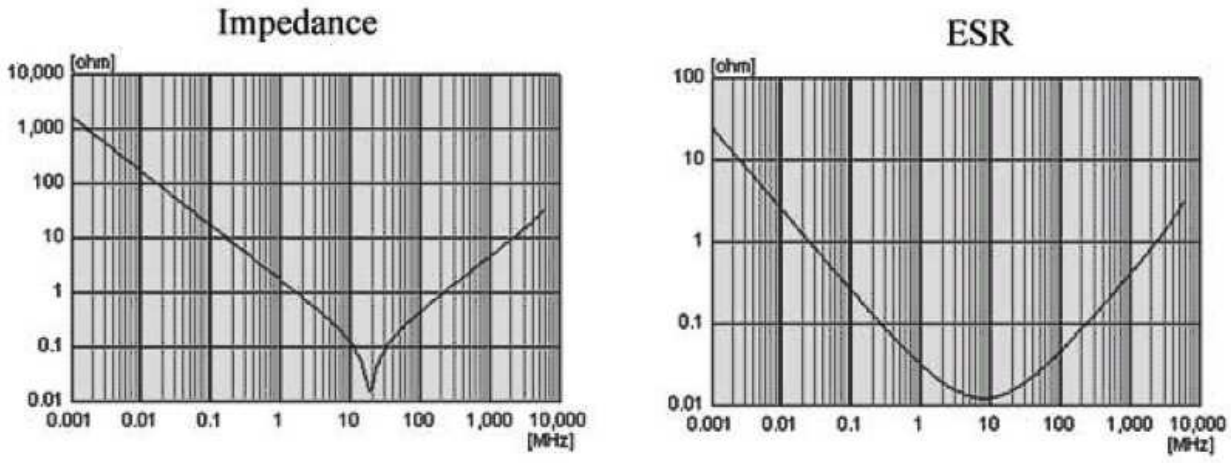


2 pav. Ekvivalentinė realaus kondensatoriaus schema.

Patyrinėję ekvivalentinę kondensatoriaus schemą, pamatysime, kad kondensatorius kintamos įtampos grandinėje elgiasi kaip nuoseklus rezonansinis kontūras, kurio rezonansinis dažnis yra

$$f_r = \frac{1}{2\pi\sqrt{LC}} \,. \qquad \textbf{(1)}$$

Esant mažesniam dažniui už f_r, realaus kondensatoriaus dažninė charakteristika yra talpinio pobūdžio, o didesniuose dažniuose pradeda dominuoti induktyvumas. Dažniausiai kondensatoriaus induktyvumo pasireiškimas yra nepageidautinas, tuomet, esant tam pačiam dažniui, yra dvi galimybės: mažinti kondensatoriaus induktyvumą arba talpą. Antrasis variantas rodo, kad didelės talpos kondensatoriai aukšto dažnio grandinėse yra nenaudingi. Mažinti induktyvumą galima renkantis kondensatorių su kitu korpusu, dažniausiai mažesniu.

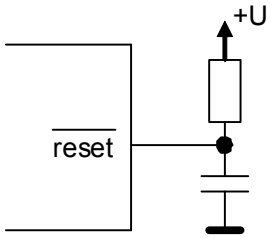


3 pav. Kondesariaus impedanso ir ekvivalentinės varžos priklausomybė nuo dažnio. Kondesatoriaus talpa 0.1 µF, korpusas 0805, korpuso induktyvumas 0.73 nH.

Kaip matyti iš 3 paveikslo, tiek kondensatoriaus impedansas, tiek ekvivalentinė varža turi minimumą. O jeigu tam tikram dažniui yra impedanso ir ekvivalentinės varžos minimumas, tai kondensatorius, esant tam dažniui, efektyviausiai atiduoda sukauptą krūvį. Iš šios kondensatoriaus elgsenos seka tokios išvados (arba patarimai): prie mikrovaldiklio išvado geriau jungti ne vieną didesnės talpos kondensatorių, o kelis kondensatorius tam tikros talpos, suskaičiuotos atsižvelgiant į kondensatoriaus korpuso induktyvumą ir dažnį. Periodiško mikrovaldiklio srovės suvartojimo dažnis ar periodas visada yra žinomas.

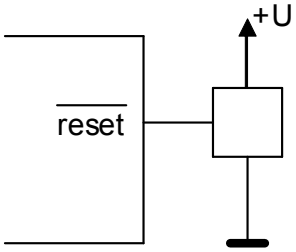
1.2 „Reset“ signalas

Įjungus mikrovaldikliui maitinimą, visų jo registrų vertės yra neapibrėžtos. Reikalingas elektrinis signalas, kuris atliktų pradinius visų registrų nustatymus, tai – „reset“ signalas. Jam būna išskirta atskira procesoriaus koja, tačiau mikrovaldikliuose, taupant resursus (išvadus), dažnai „reset“ signalo grandinė yra integruojama į vidų. Taigi „reset“ signalo užduotis – neleisti pradėti mikrovaldikliui dirbti, kol nėra tam sąlygų (maitinimo įtampa turi pakilti iki darbinės vertės, turi startuoti taktinių impulsų generatorius), o visiems parametrams esant normos ribose, nustatyti teisingas pradines registrų vertes ir leisti dirbti.



4 pav. RC grandinės panaudojimas „reset“ signalui formuoti.

Paprastai, mikrovaldiklio „reset“ signalo išvadas turi invertuotą veikimo logiką: aktyvus yra žemas lygis. Atsižvelgiant į minėtas „reset“ signalo funkcijas, parasčiausiu atveju galima galvoti taip: labai tikėtina, kad mikrovaldiklio maitinimo įtampa pakils iki reikiamo lygio ir taktinių impulsų generatorius startuos visada per maždaug vienodą laiko tarpą, tuomet reikia neleisti mikrovaldikliui pradėti dirbti fiksuotą laiko tarpą. Šias funkcijas gali atlikti paprasčiausia RC grandinė, prijungta prie mikrovaldiklio „reset“ išvado: įjungus maitinimą, įtampa ant kondensatoriaus pakils per tam tikrą laiko tarpą ir leis mikrovaldikliui pradėti darbą. Deja, RC grandinės panaudojimas „reset“ signalui formuoti turi rimtų trūkumų: įtampos ant kondensatoriaus kilimo trukmė priklauso nuo temperatūros (nes mikrovaldiklio išvado įėjimo srovės ir kondensatoriaus parametrai priklauso nuo temperatūros); jei maitinimo įtampa kils per daug lėtai, RC grandinė per daug greitai paleis mikrovaldiklį (dar turi startuoti taktinių impulsų generatorius); darbo metu sumažėjusi iki neleistino lygio maitinimo įtampa sutrikdys mikrovaldiklio darbą, o ši RC „reset“ signalo formavimo grandinė nesuformuos „reset“ signalo.



5 pav. Lusto, skirto „reset“ signalui formuoti, jungimas prie mikrovaldiklio.

Visus minėtus trūkumus, kuriuos turi RC „reset“ signalo formavimo grandinė, ištaiso sudėtingesnės schemos. „Reset“ signalui formuoti yra gaminami specialūs lustai, kurie

jungiami prie mikrovaldiklio taip, kaip parodyta **Error! Reference source not found.** paveiksle. Jie yra skirti tam tikrai maitinimo įtampai *U*, seka, kad maitinimo įtampa neišeitų už ±5% ribų (priklauso nuo konkretaus lusto), priešingu atveju aktyvuojamas „reset“ signalas, kuris lieka aktyvus dar 100 ms (priklauso nuo konkretaus lusto), kai maitinimo įtampa sugrįžta į leistiną maitinimo įtampų diapazoną.

„Reset“ išvadas yra labai svarbus mikrovaldiklio darbui, nes bet koks įtampos sumažėjimas ant šio išvado privers mikrovaldiklį pradėtį darbą iš pradžių. Įtampos sumažėjimo priežastis gali būti įvairūs trukdžiai, atsirandantys schemoje dėl netinkamo lustų įjungimo į schemą, netinkamos maitinimo grandinės ar tiesiog schemą su mikrovaldikliu eksplotuojant elektromagnetiniu požiūriu triukšmingoje aplinkoje. Žinodami šį pavojų, mikrovaldiklių gamintojai integruoja specialias schemas, kurios leidžia „reset“ išvadui būti nejautriu trumpiems impulsams. Tuomet, norint aktyvuoti „reset“ signalą, ant „reset“ išvado įtampa turi būti žemo lygio ne trumpiau kaip tam tikras laiko tarpas.

„Reset“ grandinė *PIC18* šeimos mikrovaldikliuose. ***PIC18*** šeimos mikrovaldikliuose „reset“ signalui skirtas išvadas vadinamas ***MCLR***, jis turi dar vieną paskirtį – pervesti mikrovaldiklį į programavimo būseną. Kai ant ***MCLR*** išvado įtampa tampa lygi 12V<*U*<14V, mikrovaldiklis pereina į programavimo būseną, tuomet į jo ***EEPROM*** atmintį su programatoriumi galima įrašyti programą. ***PIC18*** mikrovaldikliai turi integruotas grandines, kurios seka, kad maitinimo įtampos neišeitų iš leistino diapazono, o taip atsitikus formuoja „reset“ signalą, turi grandinę, kuri formuoja 72 ms trukmės intervalą. Kai kuriuose modeliuose galima pakeisti ***MCLR*** išvado paskirtį, tuomet jis tampa eiliniu išvadu, išlieka tik jo antroji funkcija – mikrovaldiklio pervedimas į programavimo būseną. Minėtos ***PIC18*** mikrovaldiklių grandinės gali būti įjungtos ar išjungtos programavimo su specialiu programatoriumi metu.

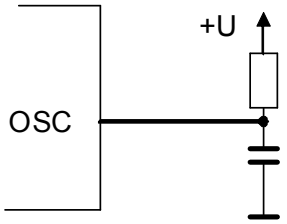
1.3 Taktinių impulsų šaltinis

Mikrovaldiklio kaip ir bet kurio procesoriaus darbui yra būtinas taktinių impulsų šaltinis, šie impulsai sinchronizuoja visų mikrovaldiklio grandinių darbą. Mikrovaldikliuose būna numatyti įvairūs taktinių impulsų šaltiniai, konkretus pasirinkimas priklauso nuo elektroninės grandinės su mikrovaldikliu taikymo. Reikalavimai taktinių impulsų šaltiniui gali būti įvairūs (jie irgi priklauso nuo elektroninės grandinės taikymo srities): kuo tikslesnis taktinių impulsų dažnis (pvz., svarbu kai naudojama į mikrovaldiklį integruota ***USB*** sąsaja), kuo pigesnė savikaina (svarbu masinei gamybai), kuo mažesnis energijos suvartojimas (svarbu iš baterijos maitinamiems įrenginiams) ir t.t. Šie skirtingi reikalavimai schemai sąlygoja skirtingų mikrovaldiklio taktinių impulsų šaltinių poreikį, į tai atsižvelgdami, mikrovaldiklių gamintojai numato įvairių taktinių impulsų šaltinių prijungimo galimybę.

Vidinis RC generatorius. Daugelis mikrovaldiklių gamintojų integruoja vidinį RC generatorių. Pats generatoriaus pavadinimas sako, kad taktinių impulsų laikines charakteristikas sąlygoja rezistorius ir kondensatorius, todėl šio tipo generatorių yra nesunku integruoti į mikrovaldiklį. Deja, šių elementų nominalai gamybos metu gaunami išsibarstę, o paties vidinio RC taktinio generatoriaus dažnis būna nukrypęs iki 4 % nuo nominalios vertės. Jei gamintojas gamybos metu atlieka generatoriaus kalibravimą, gaunamas 2 % tikslumas. Vidinį RC generatorių galima panaudoti kaip taktinių impulsų šaltinį tuomet, kai mikrovaldiklis neprivalo užtikrinti tikslių laikinių charakteristikų. Pavyzdžiui, toks taktinių

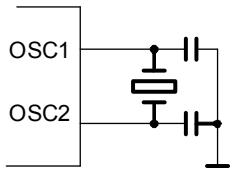
impulsų šaltinis tiktų, jei būtų naudojama į mikrovaldiklį integruota ***RS232*** sąsaja, bet netiktų, jei būtų naudojama ***CAN*** ar ***USB*** sąsajos. Kai kuriuose mikrovaldikliuose po „reset“ signalo kaip taktinių impulsų šaltinis mikrovaldi,kliui automatiškai įjungiamas vidinis RC generatorius, o perjungimas į kitą taktinių impulsų šaltinį vyksta programiniu būdu.

Išorinis RC generatorius. Rezistorius ir kondensatorius yra atskirai į schemą jungiami elementai, o likusi generatoriaus grandinė yra integruota į mikrovaldiklį. Tokio taktinių impulsų generatoriaus trūkumai: netikslus taktinių impulsų dažnis, kuris priklauso ir nuo maitinimo įtampos ir nuo temperatūros. Privalumai: paprastumas ir pigumas. Tokių generatorių dažnis paprastai būna iki 4 MHz.



6 pav. Išorinės RC grandinės jungimas prie mikrovaldiklio.

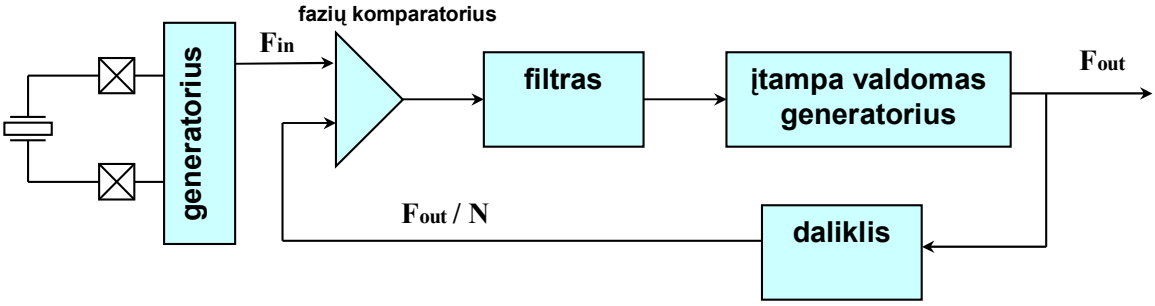
Kvarcinis arba keraminis rezonatorius. Kvarcinis arba keraminis rezonatorius prie likusios generatoriaus grandinės, kuri yra integruota mikrovaldiklyje, jungiamas taip kaip parodyta **Error! Reference source not found.** pav. Kvarcinio rezonatoriaus atveju yra jungiami apie 20 pF talpos kondensatoriai, kurie reikalingi generuojamų impulsų stabilumui, tačiau padidina srovės suvartojimą. Keraminiam rezonatoriui šie kondensatoriai nereikalingi. Tokio generatoriaus dažnis paprastai būna nuo 1 MHz iki 20 MHz, o didesnis taktinių impulsų dažnis gaunamas naudojant dažnio dauginimo grandines, kurios taip pat yra integruojamos į mikrovaldiklius. Taktinių impusų generatorius su kvarciniu arba keraminiu rezonatoriumi generuoja itin stabilaus dažnio taktinius impulsus (keraminis rezonatorius šiek tiek prasčiau) su itin maža temperatūrine dažnio priklausomybe. Trūkumas – didesnė kaina.



7 pav. Rezonatoriaus jungimas prie mikrovaldiklio

Išorinis taktinių impulsų generatorius. Visada yra galimybė prie mikrovaldiklio jungti išorinę grandinę, kuri pati generuoja taktinius impulsus (aukščiau nagrinėtuose variantuose buvo naudojamas mikrovaldiklyje integruotas generatorius, o prie išvadų jungiama tik dažnį užduodanti grandinė). Poreikis panaudoti atskirą elektroninės schemos elementą taktiniams impulsams generatoruo gali atsirasti tuomet, kai schemoje be mikrovaldiklio yra kiti elementai, kuriems reikalingi to paties dažnio taktiniai impulsai arba taktinių impulsų šaltiniui yra keliami didesni reikalavimai (pvz., dažnio stabilumui ar patikimumui), nei gali užtikrinti vidinis mikrovaldiklio generatorius.

Dažnio dauginimo grandinės. Paprastai, taktiniams mikrovaldiklio impulsams išorinės grandinės naudojamos iki 20 MHz dažnio. Tačiau, jei yra reikalingas žymiai didesnis taktinis dažnis, yra naudojamos mikrovaldiklyje integruotos dažnio dauginimo grandinės. Jų veikimo principą iliustruoja 8 paveikslas.



8 pav. Dažnio dauginimo grandinės blokinė schema

Mikrovaldiklyje integruotas generatorius generuoja taktinius impulsus tokio dažnio, koks yra prijungtas mikrovaldiklio išvadų kvarcinis rezonatorius. Iš generatoriaus F_{in} dažnio signalas patenka į dažnio dauginimo grandinę, kurios veikimo principas yra toks: įtampa valdomas generatorius generuoja $F_{out} = N \cdot F_{in}$ dažnio impulsus, generatoriaus dažnis yra paderinamas lyginant F_{in} su F_{out}/N signalus, kur F_{out}/N dažnio signalas gaunamas padalinus su dalikliu generatoriaus išėjimo signalą F_{out}. Šis F_{out} signalas gali būti mikrovaldiklio taktinių impulsų šaltinis, jis yra N kartų didesnio dažnio nei kvarcinio rezonatoriaus dažnis.

2. *PIC18* mikrovaldiklių vidinės elektroninės grandinės

1.4 Atmintis

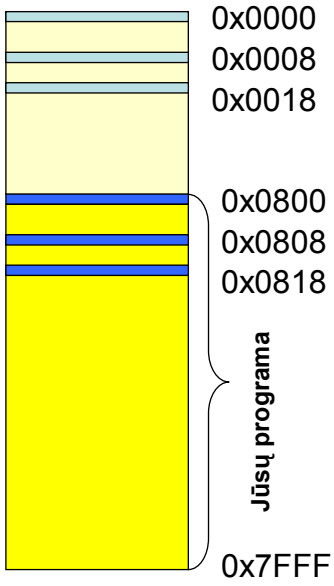
PIC18 šeimos mikrovaldikliuose yra realizuota Harvardo architekūra, t.y. atmintis programai ir atmintis duomenims yra atskirta: programa negali būti vykdoma iš duomenų atminties, tačiau duomenys į programos atmintį galį būti patalpinti. Pagal elektrines savybes, atmintis yra dviejų rūšių: ***RAM*** atmintis (atminties turinys išnyksta išjungus maitinimą) ir elektriškai perprogramuojama ***EEPROM*** atmintis (atminties turinys neišnyksta išjungus maitinimą). Kaip ir visuose procesoriuose, taip ir ***PIC18*** mikrovaldikliuose yra 31 lygio stekas, kuris yra realizuotas atskirai nuo duomenų atminties, ir vidiniai registrai, tiksliau, tokių yra tik vienas – **w** darbinis registras. ***PIC18*** mikrovaldikliuose naudojama ***RISC*** komandų sistema, t.y. supaprastinta komandų sistema: pvz. nėra komandų, kurios duomenis siųstų iš vienos atminties ląstelės į kitą, reikės su viena komanda iš atminties ląstelės vertę patalpinti į darbinį registrą **w**, o su kita komanda iš darbinio registro perrašyti ten kur norėjote.

Taigi, atmintis ***PIC18*** mikrovaldikliuo yra tokia:

- atmintis duomenims:
  - ***RAM*** atmintis;
  - ***EEPROM*** atmintis su padidintu trynimo/rašymo ciklų skaičiumi;
- atmintis programai:
  - ***EEPROM*** atmintis;
- stekas:
  - ***RAM*** atmintis, 31 lygis;
- darbinis registras **w**:
  - ***RAM*** atmintis, 1 baitas;
- konfigūraciniai bitai:
  - ***EEPROM*** atmintis.

1.4.1. Atmintis programai

PIC18 šeimos mikrovaldikliuose viena komanda užima du baitus atmintyje: jei mikrovaldiklyje yra 32 kB atminties programai, tai joje telpa 16 k komandų. Komandas mikrovaldiklis pradeda vykdyti nuo pirmosios atminties ląstelės, kurios adresas yra 0. Yra dar dvi ypatingos ląstelės programos atmintyje: įvykus trūkiui, mikrovaldiklis nutraukia normalų komandų vykdymą ir vykdymas perduodamas į 0x08 arba 0x18 programos atminties ląstelę. Detaliau apie trūkius bus pasakojama kitame skyriuje.



9 pav. Atmintis programai PIC18 mikrovaldiklyje.

Programa yra įrašoma su specialiu programatoriumi. Tačiau maketuose, su kuriais studentai dirba praktinių užsiėmimų metu, yra naudojamas kitas būdas. Į mikrovaldiklio programai skirtos atminties pradžią yra įrašytas užkroviklis. Jis iš kompiuterio per ***RS232*** arba per ***USB*** sąsaja gali atsisiųsti programą ir įrašyti į laisvą atmintį. Mikrovaldiklio atmintis tampa padalinta į dvi dalis: atminties pradžia yra niekada nekeičiama, nes joje yra užkroviklis, o likusi atminties dalis yra programuotojo žinioje. Šitaip vykdant programos užkrovimą, specialus programatorius yra nereikalingas, tačiau programuotojas turi kompiliuoti programą su specialiais nustatymais. Visų pirma, dabar programa prasideda ne nuo 0 ląstelės, o nuo 0x800 ląstelės,– tai reikia pasakyti saistyklė (angl. linker). Trūkio vektoriai arba atminties ląstelės, į kurias valdymas perduodamas, kai įvyksta trūkis, yra užkroviklio srityje ir negali būti keičiamos. Tačiau 0x08 ląstelėje yra įrašyta komanda `goto 0x808`, o 0x18 ląstelėje – `goto 0x818`. Taigi, įvykus trūkiui, mikrovaldiklis puola vykdyti 0x08 arba 0x18 ląstelę, o paskui valdymas perduodamas į programuotojo adresų sritį – 0x808 arba 0x818 ląstelę. Kompiliatoriui turi būti nurodyta, kad trūkių vektoriai yra ne įprastoje vietoje, o 0x808 ir 0x818 ląstelėse.

Kaip programuotojas gali nuskaityti ar modifikuoti programos atmintį? Visų pirma, programos atminties modifikavimas vyksta ne po vieną baitą, bet po 64 baitus – trynimas, 8 baitus – rašymas, 1 baitą – skaitymas. Prieš rašant į atmintį reikia ją ištrinti: ištrynimo procedūra įjungia visus bitus baite, t.y. visuose ištrintuose baituose yra 0xFF vertė. Rašymo

procedūra pakeičia bitų vertę iš 1 į 0, bet ne atvirkščiai, todėl prieš rašant būtina ištrinti atmintį. Programos atminties ląstelės adresas yra 3 baitų ilgio, jam nustatyti yra naudojami **TBLPTRU**, **TBLPTRH** ir **TBLPTRL** registrai, nuskaityta vertė talpinama į **TABLAT** registrą.

Bet kurios programos atminties ląstelės skaitymas:

- įrašyti ląstelės adresą į **TBLPTRU**, **TBLPTRH** ir **TBLPTRL** registrus,
- įvykdyti asemblerinę komandą `TBLRD*` (mikrovaldiklio komandų sistemoje yra ir daugiau šios paskirties komandų, bet papildomai atliekančių dar ir adreso padinimą ar sumažinimą),
- paimti rezultatą iš **TABLAT** registro.

Programos atminties bloko trynimas:

- trynimas vyksta 64 baitų blokais, reikia įrašyti bloko adresą į **TBLPTRU**, **TBLPTRH** ir **TBLPTRL** registrus, 0-5 bitai **TBLPTR** registre yra ignoruojami,
- įjungti **EEPGD**, **FREE** ir **WREN** bitus **EECON1** registre,
- patartina uždrausti trūkius,
- įrašyti 0x55 į **EECON2** registrą,
- įrašyti 0xAA į **EECON2** registrą,
- įjungti **WR** bitą **EECON1** registre, trynimo metu mikrovaldiklio branduolys sustabdomas iki kol baigsis trynimas, tačiau kitos mikrovaldiklio elektroninės grandinės dirba be pertrūkio,
- vykdyti asemblerio `NOP` instrukciją (reikalingą procesoriaus branduoliui grįžtant vėl į darbinę būseną),
- leisti trūkius, ištrinti **WREN** bitą **EECON1** registre.

Progrmos atminties bloko rašymas:

- rašymas vyksta 8 baitų blokais,
- reikia įrašyti baito adresą į **TBLPTRU**, **TBLPTRH** ir **TBLPTRL** registrus,
- reikia įrašyti vertę į `TBLAT` registrą,
- įvykdyti asemblerio `TBLWR*` procedūrą,
- kartojant aukščiau išvardintus veiksmus, įrašyti 8 baitus. Realiai, šie baitai dar nėra įrašyti į programos atmintį, jie saugomi vidiniuose mikrovaldiklio registruose,
- įjungti **EEPGD** ir **WREN** bitus **EECON1** registre,
- patartina uždrausti trūkius,
- įrašyti 0x55 į **EECON2** registrą,
- įrašyti 0xAA į **EECON2** registrą,
- įjungti **WR** bitą **EECON1** registre, rašymo metu mikrovaldiklio branduolys sustabdomas iki kol baigsis trynimas,
- vykdyti asemblerio `NOP` instrukciją (reikalingą procesoriaus branduoliui grįžtant vėl į darbinę būseną),
- leisti trūkius, ištrinti **WREN** bitą **EECON1** registre.

1.4.2. *RAM* atmintis duomenims

RAM atmintis duomenims yra suskirstyta bankais nuo 0 iki 15, t.y. viso yra 16 bankų po 256 baitus. Didžiausias ***RAM*** atminties kiekis ***PIC18*** mikrovaldikliuose galėtų būti 4096 baitai, tačiau dalį 0 ir 15 banko užima specialios atminties ląstelės, kurios kontroliuoja mikrovaldiklio elektroninių grandinių darbą. Todėl programuotojo žinioje būna ne daugiau 3840 baitų atminties, priklausomai nuo konkretaus mikrovaldiklio modelio.

Specialūs registrai, valdantys mikrovaldiklio elektroninių grandinių darbą, kaip jau minėta, yra 0 ir 15 banke, iš viso jie užima 256 baitus, t.y. vieną banką. Komandų sistemoje komandos, kurios yra skirtos kreiptis į atmintį, turi parametrą, nurodantį, kad kreipiamasi į minėtą atmintį ir nereikia nustatyti banko (banko nustatymui būtų reigaliga dar mažiausiai viena instrukcija), t.y. rašymas/skaitymas vykdomas su viena instrukcija (viena asemblerio komada).

Kreipiantis į vartotojui skirtą ***RAM*** atmintį, papildomai reikia nurodyti banką, o tai pailgina skaitymą/rašymą iki dviejų asemblerio komandų. Galima naudoti tiesioginę ir netiesioginę adresaciją. Pirmuoju atveju į `BSR` registrą įrašomas banko numeris ir kreipiamasi į atminties ląstelę. Antruoju atveju atminties ląstelės adresas (12 bitų) yra įrašomas į `FSR0H`, `FSR0L` arba `FSR1H`, `FSR1L` arba `FSR2H`, `FSR2L`, o vertė skaitoma/rašoma į `INDF0`, `INDF1` arba `INDF2` registrą. Antrasis būdas dažniausiai naudojamas prieiti prie masyvo elementų.

Programuojant aukšto lygio kalba, pavyzdžiui ***C***, apie ***RAM*** atminties adresavimo būdus galima užmiršti, – kompiliatorius automatiškai generuos kodą su vienu iš adresavimo būdų. Minėtus registrus modifikuoti, programuojant ***C*** kalba, reikia labai atsargiai, nes lengvai sutrikdysite normalų programos vykdymą.

10 pav. RAM atmintis. Paryškintos 0 ir 15 banko dalys vaizduoja sritį, kurioje yra specialūs registrai.

1.4.3. ***EEPROM*** *duomenų atmintis*

PIC18 šeimos mikrovaldikliuose ***EEPROM*** duomenų atmintis būna 256, 1024 arba 2048 baitų, pasižyminti padidintu trynimo/rašymo ciklų skaičiumi (ben jau 1 mln.). Ši atmintis – kietojo disko personaliniuose kompiuteriuose analogas, išjungus maitinimo įtampą duomenys iš jos nedingsta. Šios duomenų atminties skaitymas/rašymas vykdomas naudojant **EEDATA**, **EEADR**, **EECON1** ir **EECON2** registrus. Skaitymas/rašymas vyksta po vieną baitą, trynimo kaip atskiros procedūros nėra, duomenų atminties ląstelė ištrinima automatiškai, kai įjungiamas rašymas.

Skaitymo procedūra:

- įrašyti adresą į **EEADR** registrą,
- **EECON1** registre įjungti **EEPGD** ir **CFGS** bitus,
- **EECON1** registre įjungti **RD** bitą,
- paimti rezultatą iš **EEDATA** registro, rezultatą galima paimti sekančioje komandoje po **RD** bito įjungimo.

Rašymo procedūra:

- įrašyti adrsą į **EEADR** registrą,
- įrašyti vertę į **EEDATA** registrą,
- **EECON1** registre įjungti **EEPGD** ir **CFGS** bitus,
- **EECON1** registre įjungti **WREN** bitą,
- įrašyti skaičių 0x55 į **EECON2** registrą,
- įrašyti skaičių 0xAA į **EECON2** registrą,
- įjungti **WR** bitą **EECON1** registre,
- **WR** bitas išsijungs, kai baigsis rašymo ciklas, kol baigsis rašymo ciklas reiktų laukti, jei paskui iš karto seka kito baito rašymas į duomenų EEPROM atmintį,
- išjunti **WREN** bitą **EECON1** registre.

1.5 Trūkiai

Trūkių paskirtis – greita mikrovaldiklio reakcija į įvykį (elektrinį signalą). Įsivaizduokite prie mikrovaldiklio išvado prijungtą mygtuką, kurio būseną turi sekti programa mikrovaldiklyje. Be to, programa vykdo kitas užduotis ir negali pastoviai skanuoti išvado būsenos. Pirmas variantas: programa tam tikrais laiko tarpais patikrina išvado būseną, bet tada programa pavėluotai sureaguos į mygtuko paspaudimą. Antras variantas: mygtuko paspaudimas ir įtampos pokytis ant mikrovaldiklio išvado aktyvuoja trūkį, tuomet, mikrovaldiklis nutraukia pagrindinės programos vykdymą ir pradeda vykdyti trūkio paprogramę. Tiek pirmas, tiek antras variantai yra priimtini, tačiau toliau šiame skyriuje bus nagrinėjama, kaip panaudoti trūkius.

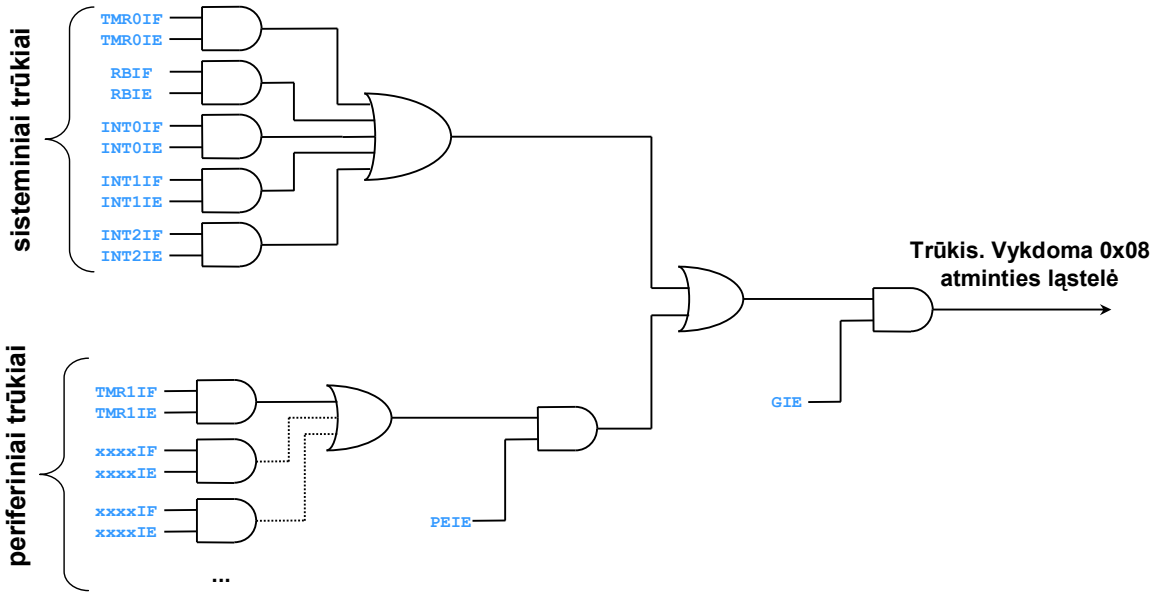
Įvykus trūkiui, procesorius nutraukia pagrindinės programos vykdymą ir pradeda vykdyti trūkio paprogramę. Trūkio šaltinis gali būti skaitiklio persipildymas, baito priėmimas iš ***RS232*** sąsajos, analoginio-skaitmeninio keitiklio įvykdytas įtampos matavimas, įtampos pokytis ant išvado ir dar daugelis kitų. ***PIC18*** šeimos mikrovaldikliuose ši paprogramė turi prasidėti tam tikroje programos atminties ląstelėje:

- 0x08 ląstelėje, jei nėra įjungti trūkių prioritetai,
- 0x08 (žemo prioriteto) arba 0x18 (aukšto prioriteto) ląstelėse, jei yra įjungti trūkių prioritetai.

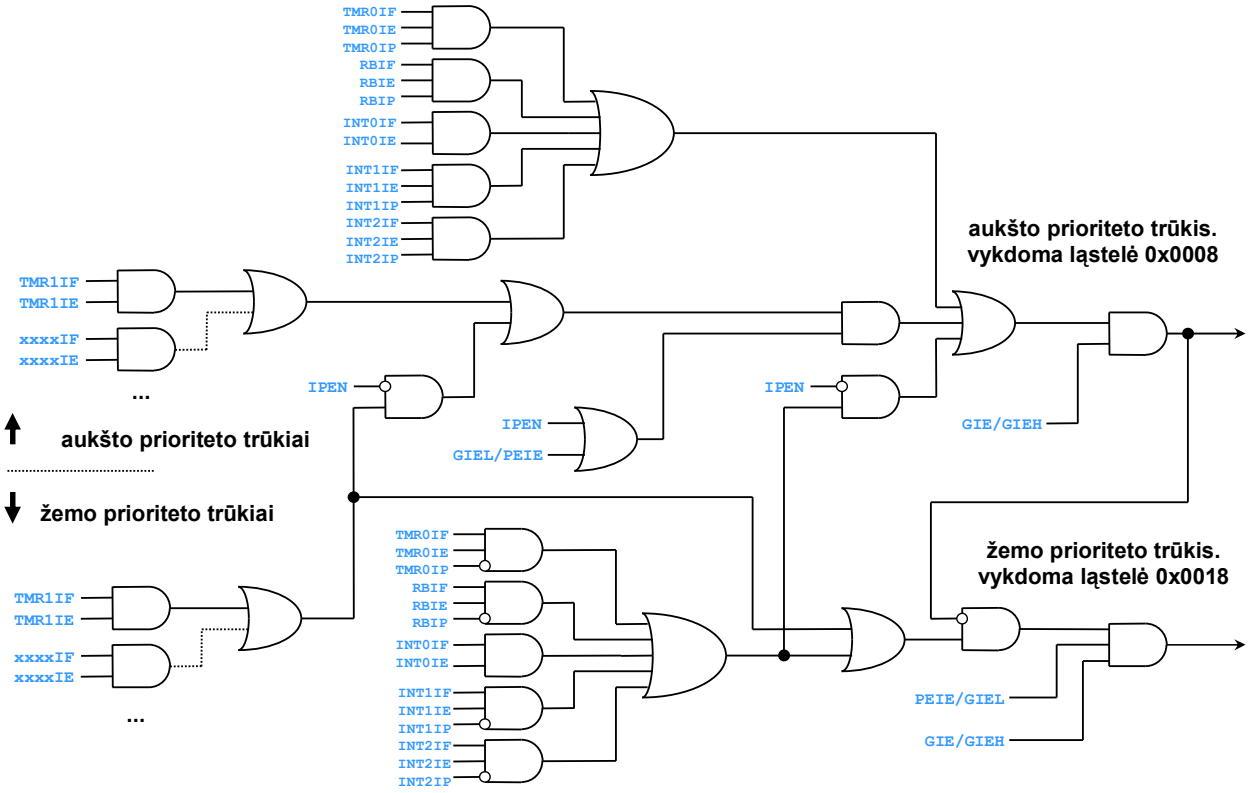
Po „reset“ signalo, mikrovaldiklyje trūkių prioritetai nėra įjungti. Trūkių prioritetus įjungia **IPEN** bitas, esantis **RCON** registre. Kiekvieną trūkio šaltinį kontroliuoja trys bitai, kurių vardai yra **xxxIE**, **xxxIF** ir **xxxIP**:

- **xxxIF** bitą įjungia elektroninės mikrovaldiklio grandinės, esant tam tikroms sąlygoms, pvz.: skaitikliui ***TIMER0*** persipildžius, įjungiamas **TMR0IF**=1 bitas. **xxxIF** bitas automatiškai neišsijungia (išimtis **RCIF** ir **TXIF**), tai gali padaryti programa.
- **xxxIE** bitas leidžia įvykti trūkiui, jei **xxxIE**=1 ir atsiranda trūkio sąlyga **xxxIF**=1, tuomet procesorius pradeda vydyti trūkio paprogramę;
- **xxxIP** bitas nustato kokio prioriteto yra trūkis. Jei **xxxIP**=1 – aukšto prioriteto trūkis, tuomet, įvykus trūkiui, vykdoma 0x18 ląstelė (vykdoma kaip paprogramė, t.y. grįžimo į pagrindinę programą adresas talpinamas į steką, paprogramė baigiasi specialia grįžimo komanda `RETIE`), jei **xxxIP**=0 – žemo prioriteto trūkis, vykdoma nuo 0x08 ląstelė. Jei trūkių prioritetų galimybė nėra įjungta (**IPEN**=0), tuomet **xxxIP** bitas įtakos neturi.

Trūkiai ***PIC18*** mikrovaldikliuose skirstomi į sisteminius ir periferinius (žr. 11, 12 pav.). Jei nėra įjungti trūkių prioritetai, galima visiems trūkiams įvykti leisti su **GIE** bitu, o atskirai periferinius trūkius leisti/drausti su **PEIE** bitu. Jei yra leisti trūkių prioritetai, tai **GIE** bitas draudžia/leidžia visus trūkius, o **PEIE/GIEL** bitas leidžia/draudžia tik žemo prioriteto trūkius.



11 pav. Trūkiai be prioriteto. Trūkio aktyvavimo loginė grandinė.



12 pav. Trūkiai su prioritetu. Trūkio aktyvavimo loginė grandinė.

Rašant trūkių paprogrames ***C*** kalboje, reikia atsižvelgti į tokius momentus: a) kompiliatoriui (tiksliau, vienai iš jo dalių – saistyklei (angl. linker) turi būti nurodyta, kad trūkio paprogramė prasideda ne bet kur atmintyje, o nuo 0x08 arba 0x18 ląstelės; b) įvykus trūkiui, pagrindinės programos vykdymas yra nutraukiamas ir vykdoma trūkio paprogramė. Ar nesugadins trūkio paprogramė pagrindinės programos registrų ir duomenų? Laimei, kompiliatorius šio pavojaus leidžia išvengti be didesnių programuotojo pastangų. Kompiliatorius automatiškai įterpia į trūkio paprogramės pradžią kodą, kuris išsaugo gyvybiškai svarbius mikrovaldiklio registrus. Šie registrai yra : darbinis **W** registras, banko išrinkimo registras **BSR**, operacijų statuso registras **STATUS**, netiesioginės adresacijos registrai **FSR0**, **FSR1**, **FSR2** (jei reikia), daugybos registrai **PRODH**, **PRODL** ir kompiliatoriaus tarpinių veiksmų registrai.

Trūkio paprogramei negali būti perduoti duomenys ir ji negali gražinti rezultato taip, kaip įprasta apibrėžiant funkciją ***C*** kalboje. Duomenims perduoti ar priimti iš trūkio paprogramės yra naudojami gloablūs kintamieji. Pagrindinė programos pažiūriu šių kintamųjų vertė gali pakisti bet kuriuo programos vykdymo metu (paprindinė programa „nežino“ kada įvyks trūkis), todėl tokie kintamieji turi būti apibrėžti su kintamojo tipo modifikatoriumi **volatile**. Modifikatorius **volatile** uždraudžia kompiliatoriui taikyti optimizaciją šių kintamųjų atžvilgiu ir informuoja, kad šių kintamųjų vertė gali pakisti bet kuriuo programos vykdymo metu; kompiliatorius specialiu būdu kompiliuoja tą programos dalį, kur yra naudojami šie kintamieji.

Pavyzdys. Parašyti trūkio paprogramę, trūkio šaltinis – ***TIMER1*** skaitiklio persipildymas. Programa pritaikyta Microchip ***C*** kompiliatoriui, skirtumas nuo kitų kompiliatorių yra trūkio paprogramės sintaksė: kompiliatoriui reikia nurodyti, kad trūkio paprogramę reikia kompiliuoti ypatingu būdu, Microchip kompiliatoriui tai nustatoma **#pragma** komandomis.

```
volatile char skaicius;

void main(void)
{
    TMR1H = 0x12;      // pradinė vertė
    TMR1L = 0x23;
    T1CON = 0x01;      // daliklis 1:1, paleidžiam skaitiklį
    PIR1bits.TMR1IF = 0;  // ištrinam TMR1IF bitą
    PIE1bits.TMR1IE = 1;  // leidžiam TIMER1 trūkį
    INTCONbits.PEIE = 1;  // leidžiam periferinius trūkius
    INTCONbits.GIE  = 1;  // leidžiam visus trūkius
    for(;;)
    {
      … // galima atlikti veiksmus su kintamuoju "skaicius"
        // šio kintamojo vertė padidinama 1 kas tam tikrą
        // laiko tarpą, kurį generuoja TIMER1 skaitiklis
    }
}

#pragma code
#pragma interrup int_isr
void int_isr (void)        // trūkio paprogramė yra patalpinta
{                          // bet kurioje atminties vietoje
    T1CON = 0;             // sustabdom skaitiklį
    TMR1H = 0x12;          // vėl užkraunam pradinę vertę
    TMR1L = 0x23;
    T1CON = 0x01;          // paleidžiam skaitiklį
    skaicius++;            // padininam kintamojo vertę
    PIR1bits.TMR1IF = 0;   // būtina ištrinti TMR1IF bitą
}

#pragma code low_vector=0x808       // nurodom, kad funkcija
void interrupt_at_low_vector(void)  // turi prasidėti nuo
{                                   // 0x808 ląstelės
    _asm
    goto int_isr     // 0x808 ląstelėje bus tik goto komanda
    _endasm
}
```

1.6 Išvadai

Mikrovaldiklių išvadų paskirtis. Dalis mikrovaldiklių išvadų yra svarbūs mikrovaldiklio normaliam darbui ir programa negali jų valdyti (pvz., maitinimo išvadai), kiti išvadai yra palikti visiškai programuotojo nuožiūrai ir atsakomybei. Išvadai, kurių programuotojas negalima valdyti: maitinimo išvadai, ***MCLR*** išvadas, išvadai taktinių impulsų įvedimui arba kvarcinio rezonatoriaus prijungimo išvadai, ***USB*** įtampos stabilizatoriaus išėjimo išvadas. Tiesa, kai kurių šių išvadų paskirtį galima pakeisti programavimo su programatoriumi metu, nustatant atitinkmus konfigūracinius bitus.

Išvadai, kuriuos pilnai valdo programuotojas, yra suskirstyti į grupes po 8 išvadus. Šios grupės, sekant dokumentacijoje įvestus vardus, yra **PORTA**, **PORTB**, **PORTC** ir t.t. Kiekvienas išvadas turi savo vardą: ***RA0***, ***RA1***, ..., ***RB0***, ***RB1*** ir t.t. Kiekvienas išvadas gali būti skaitmeninis įėjimas, skaitmeninis išėjimas arba turėti trečią paskirtį. Ta trečia paskirtis yra susijusi su mikrovaldiklyje integruotomis elektroninėmis grandinėmis, kurios gali valdyti išvadus, prie kurių jos yra prijungtos. Startavus mikrovaldikliui, visi bendros paskirties išvadai yra sukonfigūruoti kaip įėjimai: dalis kaip analoginiai įėjimai, kiti – skaitmeniniai įėjimai.

Išvadus valdantys registrai:

- **TRISA**, **TRISB**, ... – valdo išvado kryptį: bito vertė 0 – skaitmeninis išėjimas, 1 – skaitmeninis įėjimas;
- **PORTA**, **PORTB**, ... – nuskaito arba išstato įtampą ant mikrovaldiklio išvado.

Pavyzdžiui, norint į ***RA5*** išvadą paduoti +5V įtampą, reikia **TRISA** registre 5-ą bitą padaryti lygų 0, o **PORTA** registro 5-ą bitą įrašyti 1. Norint nuskaityti koks yra įtampos lygis, aukštas ar žemas, ant ***RA5*** išvado, reikia **TRISA** registre 5-ą bitą padaryti lygų 1 ir nuskaityti kokia yra **PORTA** 5-o bito vertė.

Dar su išvadų valdymu yra susieti **LATA**, **LATB**, ... registrai, šie registrai – tai trigeriai, kurie saugoja į išvadus išvedamos įtampos logines vertes („0“ ar „1“). Skirtumas nuo **PORTA**, **PORTB**, ... registrų grupės yra tik skaitant jų turinį. Rašant į **PORTA**, **PORTB**, ... arba į **LATA**, **LATB**, ... skaičius patenka į trigerius ir, jei išvadas yra sukonfigūruotas kai skaitmeninis išėjimas, trigerio išėjimas sąlygoja loginio „0“ arba „1“ įtampą ant išvado. Skaitant **PORTA**, **PORTB**, ... registrų grupę, nuskaitysite įtampos lygį, kuris realiai yra ant išvado, skaitant **LATA**, **LATB**, ... registrų grupę nuskaitysite, tai ką norite išvesti, t.y. trigerių vertę. Įsivaizduokite, kad išvadas yra sukonfigūruotas kaip skaitmeninis įėjimas, prie jo prijungta loginio „0“ įtampa. Jūs rašote „1“ į **PORTA**, **PORTB**, ... arba **LATA**, **LATB**, ... atitinkamo registro atitinkamą bitą, kuris yra susijęs su nagrinėjamu išvadu. Kadangi išvadas yra skaitmeninis įėjimas, jūs įtampos ant išvado nekontroliuojate. Dabar nuskaitykite išvado įtampos lygį: skaitydami iš **PORTA**, **PORTB**, ... registrų, gausite loginį „0“ , t.y. realios įtampos ant išvado lygis, skaitydami bito vertę iš **LATA**, **LATB**, ... registrų, gausite logini „1“, t.y. tai ką įrašėte į išvedimo trigerius.

Yra dar viena situacija, kuomet gausite skirtingas vertes skaitydami bitą iš **PORTA**, **PORTB**, ... ir iš **LATA**, **LATB**, ... registrų grupės. Įsivaizduokite, kad mikrovaldiklio išvadas yra sukonfigūruotas kaip skaitmeninis išėjimas, įtampa ant jo yra loginis „0“ ir prie jo prijungta talpinio pobūdžio apkrova. Įrašote loginį „1“ į trigerį, valdantį išvado įtampą, įtampa ant mikrovaldiklio išvado pradeda didėti nuo loginio „0“ iki loginio „1“ lygio. Kadangi prie išvado yra prijungta talpinė apkrova, kuri šuntuoja išvadą, įtampa didėja pakankamai lėtai, kad vykdant kelias sekančias komandas dar nėra pasiekusi loginio „1“ lygio. Tuomet, skaitydami išvado įtampos lygį iš **PORTA**, **PORTB**, ... registrų grupės gausite realios įtampos lygį – loginį „0“, o skaitydami iš **LATA**, **LATB**, ... registrų grupės gausite loginį „1“.

Programos pavyzdys. Išvedame +5V įtampą į ***RA5*** išvadą:

```
void main(void)
{
      ...
      TRISA = 0b11011111; // 5-ą bitą konfigūruojame
                          // kaip skaitmeninį išėjimą
      PORTA = 0b00100000; // įrašome loginį "1"
      ...
}
```

Programos pavyzdys. Tikrinama kokia įtampa yra ant ***RA5*** išvado:

```
void main(void)
{
      ...
      TRISA = 0b11111111; // 5-ą bitą konfigūruojame
                          // kaip skaitmeninį įėjimą
      if(PORTA & 0b00100000)
      {
            ... // aukštas lygis
      }
      else
      {
            ... // žemas lygis
      }
      ...
}
```

1.7 Skaitikliai *PIC18* šeimos mikrovaldikliuose

Skaitiklių, kurie yra integruoti į mikrovaldiklį, paskirtis yra tokia pati kaip ir skaitiklių diskretiniame luste (jei yra poreikis, toks lustas gali būti įjungtas į schemą) – skaičiuoti impulsus. Skaitiklis mikrovaldiklyje yra atskirai nuo programos veikianti elektroninė grandinė: programa veikia sau, o skaitiklis skaičiuoja impulsus. Žinoma, modifikuodama skaitiklio darbą valdančius registrus, programa įtakoja skaitiklio darbą. Iš viso ***PIC18*** šeimos mikrovaldikliuose yra 4 skaitikliai, turintys skirtingas galimybes; jų vardai, taip kaip yra vadinami dokumentacijoje: ***Timer0***, ***Timer1***, ***Timer2*** ir ***Timer3***.

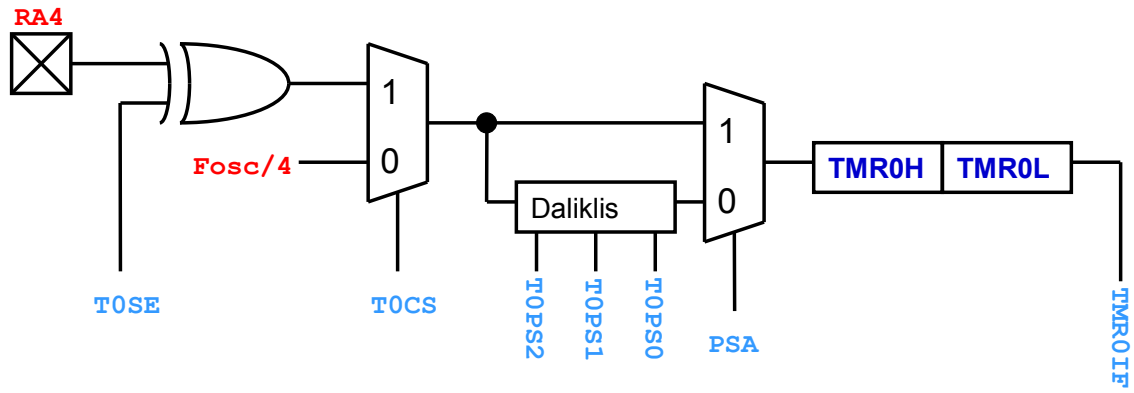
Skaitiklis ***PIC18*** šeimos mikrovaldikliuose yra 8 arba 16 bitų registras, kurio turinys, atėjus implusui, padidėja. Skaitikliai ***PIC18*** šeimos mikrovaldikliuose neturi galimybės skaičiuoti impulsus mažėjančia kryptimi. Impulsų šaltinis skaitikliui gali būti procesoriaus taktiniai impulsai arba impulsai iš išorinės grandinės. Be to, prieš skaitiklį yra impulsų daliklis, kurio dalinimo koeficientas yra keičiamas programiškai.

Impulsams ateinant į skaitiklį, jo turinys didėja, kol galų gale, pasiekęs maksimalią vertę, pereina į 0. Persipildymo metu tam tikrame registre įjungiamas bitas, informuojantis apie skaitiklio persipildymą. Tokiu būdu, programa laikas nuo laiko gali sekti ar jau įvyko skaitiklio persipildymas, t.y. ar jau praėjo tam tikras laiko intervalas. Taipogi, šis persipildymo bitas gali aktyvuoti trūkį.

1.8 *Timer0* skaitiklis

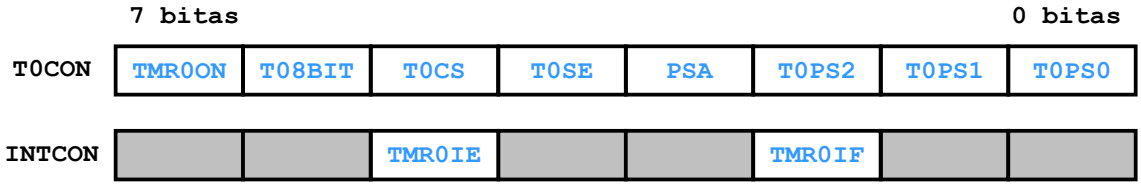
Timer0 skaitiklis gali būti 8 arba 16 bitų (pasirenkama konfigūruojant skaitiklį valdantį registrą). Skaitiklis turi daliklį kurio dalinimo koeficientas gali būti nuo 1:1 iki 1:256. Impulsų šaltinis skaitikliui gali būti taktiniai impulsai ir iš išorės į ***RA4*** išvadą ateinantys impulsai. Šiame skyriuje nagrinėsime skaitiklio darbą kai jis yra 16 bitų.

Skaitiklio funkcinė schema yra pavaizduota 13 paveiksle. Matome, kad impulsai iš ***RA4*** išvado arba taktiniai impulsai $F_{osc}/4$ patenka į komutavimo elementą, kuris valdomas **TOCS** bituku. Tolimesnis komutavimo elementas, iš kurio impulsai patenka tiesiai į skaitiklį yra valdomas **PSA** bitu. Su **PSA** bitu galima pasirinkti ar impulsai eis ar neis pro daliklį, t.y. ar dalinimo koeficientas bus 1:1 ar toks, koks bus nustatytas su **TOPSO**, **TOPS1** ir **TOPS2** bitais. Skaitikliui persipildžius, įsijungia **TMROIF** bitas, kuris gali aktyvuoti trūkį, jei yra leisti trūkiai. Skaitiklio vertė nuskaityta arba įrašyta į **TMROH** ir **TMROL** registrus.



13 pav. Skaitiklio Timer0 funkcinė schema.

Skaitiklio darbą valdantis registras **T0CON** ir bitų išsidėstymas jame yra pateiktas 14 paveiksle. Bitas **TMR0ON** įjungia maitinimą skaitiklio grandinei ir skaitiklis pradeda veikti. Bitas **T08BIT** nustato ar skaitiklis yra 8 ar 16 bitų, žemiau pateiktas pavyzdys bus kai ***Timer0*** dirba kaip 16 bitų skaitiklis t.y. kai **T08BIT=0**. Bitas **T0SE** kontroliuoja kuriam impulso, ateinančio į ***RA4*** išvadą, frontui esant vyksta skaitiklio turinio didėjimas: jei **T0SE=0** – esant kylančiam frontui, jei **T0SE=1** – esant krintančiam frontui.



14 pav. Timer0 skaitiklio darbą valdantys registrai.

Kai ***Timer0*** veikia kaip 16 bitų skaitiklis, skaitymas ir rašymas į **TMR0H** ir **TMR0L** registrus vyksta per vidinius tarpinius registrus (t.y. daromos kopijos), todėl yra svarbi skaitymo rašymo tvarka: kuris registras yra skaitomas ar rašomas pirma. Jei yra skaitomi registrai, pirma turi būti skaitomas **TMR0L** registras, skaitymo komandos vykdymo metu padaroma **TMR0H** kopija, toliau nuskaitomas **TMR0H** registras (tiksliau, jo kopija). Jei yra rašoma į šiuos registrus, pirma turi būti rašoma į **TMR0H** registrą (realiai ne įrašoma, o tik padaroma kopija tos vertės, kurią norit rašyti), paskui modifikuojamas **TMR0L** registras, tuo pačiu yra modifikuojama ir **TMR0H** registras pagal padarytą kopiją.

Jei skaitiklis yra naudojamas laiko intervalams generuoti, tuomet jį reiktų naudoti tokiu būdu: pasirinkti taktinį generatorių impusų šaltiniu, nustatyti pradinę vertę, modifikuojant **TMR0H**, **TMR0L** registrus, nustatyti jei reikia daliklio vertę, paleisti skaitiklį ir tikrinti, kada persipildys. Laiko tarpas tarp skaitiklio paleidimo ir jo persipildymo nusakomas formule:

$$T = \frac{4}{F_{osc}} D\left(65535 - TMR0_{pr}\right), \qquad \textbf{(2)}$$

čia F_{osc} yra taktinis dažnis, D – daliklio vertė (sveikas skaičius 1, 2, 4, 8, ...), $TMR0_{pr}$ – pradinė skaitiklio registrų **TMROH**, **TMROL** vertė.

Pavyzdys. Reikia parašyti paprogramę, kuri vėlintų programą 400 ms intervalui, procesoriaus taktinis dažnis *F*osc = 40 MHz. Pasirenkame daliklio vertę 1:128 , t.y. $D = 128$, surandame iš formulės $TMR0_{pr}$:

$$400\,\text{ms} = \frac{4}{40000\,\text{kHz}} \cdot 128 \cdot \left(65535 - TMR0_{pr}\right), \tag{3}$$

iš čia dydis $TMR0_{pr}$ dešimtainiame ir šešioliktainiame formate yra:

$$TMR0_{pr} = 34285 = 0\,\text{x}\,85\,\text{ED}\ . \tag{4}$$

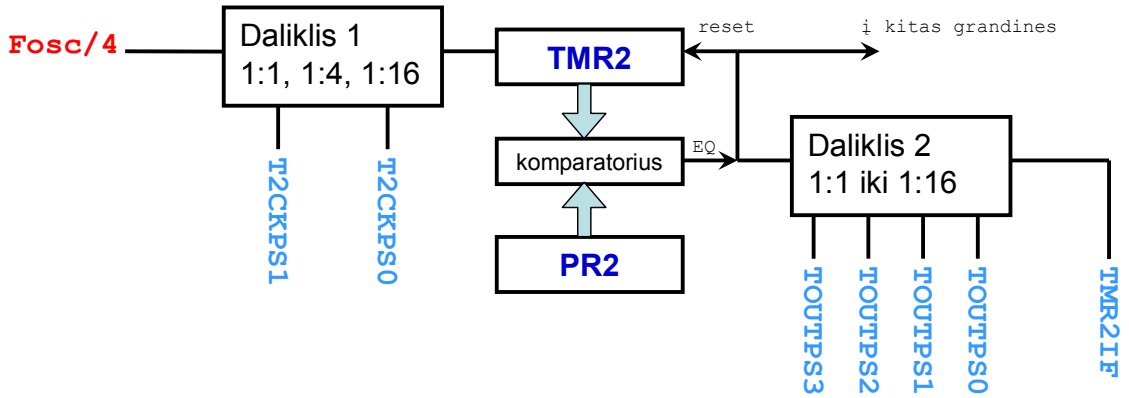
Šią vertę reikia įrašyti į skaitiklį, prieš jį paleidžiant: **TMROH** = 0x85, **TMROL** = 0xED.

Programa:

```
void laukti(void);

void main(void)
{
      ...
      laukti(); // programa sustoja 400 ms laiko intervalui
      ...
}

void laukti(void)
{
      INTCONbits.TMR0IF = 0;  // ištriname persipildymo požymį
      TMR0H = 0x85;  // pirma rašoma į TMR0H registrą!
      TMR0L = 0xED;
      T0CON = 0b10000110; // paleidžiame skaitiklį
      while(INTCONbits.TMR0IF == 0); // laukiame kol persipildys
      T0CON = 0; // išjungiam skaitiklį
}
```

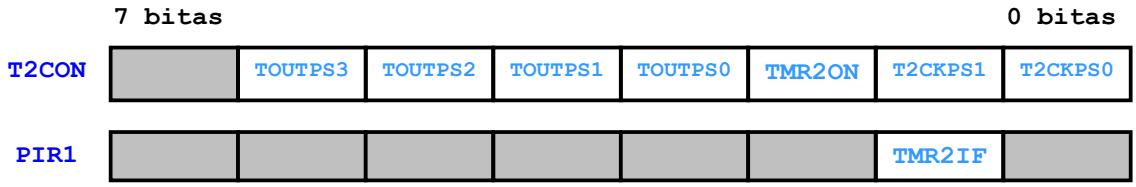
1.9 *Timer2* skaitiklis

Timer2 skaitiklio elektroninė grandinė ir galimybės ženkliai skiriasi nuo ***Timer0***. Visų pirma, ***Timer2*** skaitiklis yra 8 bitų, signalas iš jo išėjimo gali būti nuvestas į kitas mikrovaldiklio elektronines grandines. Skaitiklio darbo principai turėtų paaiškėti iš 15 paveikslo.



15 pav. Timer2 skaitiklio funkcinė schema.

Impulsų šaltinis ***Timer2*** skaitikliui gali būti tik mikrovaldiklio taktinių impulsų generatorius. Šie impulsai patenka į 1-ajį daliklį, kuriam galima nustatyti tris skirtingas dalinimo vertes. Iš daliklio impulsai patenka į 8 bitų skaitiklį, kurio vertė yra **TMR2** registras. Skaitiklis skačiuoja impulsus didėjančia kryptimi iki vertės, esančios **PR2** registre. Skaitiklio persipildymo impulsas patenka į 2-ajį daliklį ir kitas mikrovaldiklio grandines (pavyzdžiui, į ***CCP*** modulį, jei jis yra įjungtas). Signalas po 2-ojo daliklio įjungia **TMR2IF** bitą ir gali būti panaudotas trūkio aktyvavimui. Skirtingai nuo ***Timer0*** skaitiklio, nereikia rašyti pradinės vertės į **TMR2** skaitiklio registrą (nors ir galima), nes impulsų skaičių, kurį turi priskaičiuoti skaitiklis, kad persipildytų, galima nustatyti su **PR2** registru.



16 pav. Timer2 skaitiklio darbą valdantys registrai.

Skaitiklio darbą valdo **T2CON** registras. Skaitikliui maitinimą įjungia **TMR2ON** bitas, bitai **T2CKPS0** ir **T2CKPS1** valdo 1-ojo daliklio dalinimo koeficientą, bitai **TOUTPS0**, **TOUTPS1**, **TOUTPS2** ir **TOUTPS3** valdo 2-ojo daliklio dalinimo koeficientą. Laiko tarpas tarp skaitiklio paleidimo ir požymio **TMR2IF** atsiradimo yra nusakomas formule:

$$T = \frac{4}{F_{osc}} D_1 D_2 \left(PR2 + 1\right), \qquad \textbf{(5)}$$

čia F_{osc} yra taktinis dažnis; D_1, D_2 yra daliklių vertės; *PR2* – registro **PR2** vertė. Laikoma, kad pradinė skaitiklio **TMR2** vertė yra 0.

Pavyzdys. Reikia sukonfigūruoti ***Timer2*** skaitiklį taip, kad kas 1 ms atsirastų **TMR2IF** požymis ir vyktų trūkis, procesoriaus taktinis dažnis F_{osc} = 4 MHz. Pasirenkame daliklius: D_1=4 ir D_2=10, tuomet

$$1000\,\mu s = \frac{4 \cdot 4 \cdot 10}{4\,\text{MHz}} \cdot (PR2 + 1)\,, \tag{6}$$

iš čia **PR2** registro vertė:

$$PR2 = 24\,. \tag{7}$$

Programa:

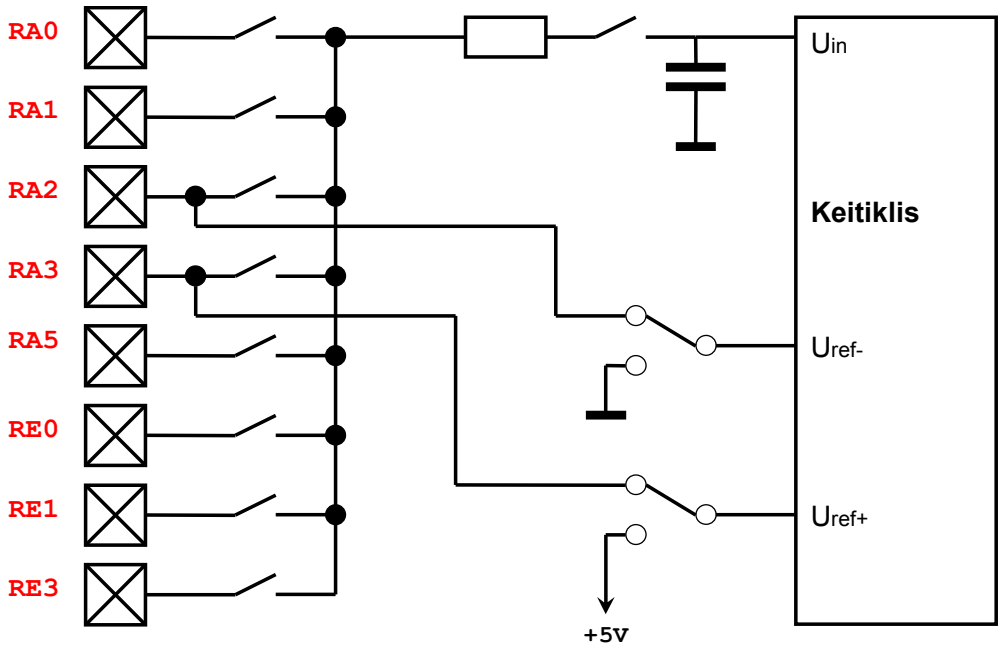
```
void main(void)
{
      PIR1bits.TMR2IF = 0;    // ištriname trūkio požymį
      PR2 = 24;
      TMR2 = 0;
      T2CON = 0b01001101;     // daliklių nustatymas
      PIE1bits.TMR2IE = 1;    // leidžiam skaitiklio trūkius

      ... // likusi programa Timer2 skaitiklio registrų nekeičia
}

... // trūkio apdorojimo paprogramė
```

1.10 Analoginis-skaitmeninis keitiklis

Analoginio-skaitmeninio (***AS***) keitiklio paskirtis – matuojamą įtampą įvertinti skaičiumi. Supaprastintai ***AS*** keitiklio darbą įsivaizduoti galimą ir taip: keitiklis matuojamų įtampų diapazoną suskaido į intervalus ir nustato, kuriam intervalui priklauso matuojamoji įtampa; šio intervalo numeris (o numeris yra skaičius) ir yra keitiklio rezultatas. Idealiu atveju, šie intervalai, kuriais keitiklis suskaido matuojmų įtampų diapazoną, turėtų būti vienodi, tačiau realybė yra kitokia, o šie nukrypimai yra keitiklio paklaidos. Intervalų skaičius yra keitiklio skiriamoji geba, kuri dažniausiai yra išreiškiama bitų skaičiumi. Jei ***AS*** keitiklio skiriamoji geba išreikšta bitais yra x, tai intervalų skaičius yra 2^x.

Mikrovaldikliuose integruoto ***AS*** keitiklio skiriamoji geba retai būna didesnė nei 10 bitų, nors AS keitiklių, realizuotų kaip atskiras lustas, skiriamoji geba gali būti 24 ar net daugiau bitų. Priežastis kodėl į mikrovaldiklį yra integruojamas tik 10 bitų keitiklis yra kaina (išorinio ***AS*** keitiklio su geromis charakteristikomis kaina yra artima mikrovaldiklio kainai) ir aplinka, kurioje veikia integruotas keitklis: mikrovaldiklio branduolys kuria elektromagnetinius triukšmus, kurie neigiamai įtakoja keitiklio darbą. Netgi integruoto 10 bitų keitiklio jauniausieji bitai „triukšmauja“ ir, norint išspausti didžiausia keitiklio tikslumą, reikėtų stabdyti mkrovaldiklio branduolį ir kitas integruotas mikrovaldiklio grandines kol vyksta keitimo procesas (tokios galimybės būna integruotos mikrovaldikliuose).

PIC18 šeimos mikrovaldikliuose yra integruotas 10 bitų keitiklis, yra keletas išvadų, kurie gali tapti analoginio signalo įėjimais ir gali būti matuojama prie jų prijungta įtampa. Mikrovaldiklio išvadų komutavimas su AS keitikliu yra parodytas 17 paveiksle, tiesa, skirtingiems mikrovaldiklių modeliamas, ši schema šiek tiek skiriasi.



17 pav. Analoginio-skaitmeninio keitiklio įėjimų komutavimo schema.

Nors yra integruotas tik vienas keitiklis, tačiau išvadą, ant kurio yra matuojama įtampa (žinoma, iš galimų variantų, žr. 17 pav.), galima nustatyti programiškai.

***AS* keitiklio veikimo principai.** Keitiklis matuoja nuo U_{ref-} iki U_{ref+} įtampų diapazone. Šios U_{ref-} iki U_{ref+} įtampos nebūtinai turi sutapti su maitinimo įtampomis, bet negali išeiti iš maitinimo įtampų diapazono; šios įtampos gali būti prijungtos prie mikrovaldiklio išvadų iš išorinio šaltinio arba prijungtos mikrovaldiklio viduje prie maitinimo įtampų. U_{ref-} ir U_{ref+} įtampų atžvilgiu vyksta nežinomos įtampos matavimas, todėl U_{ref-} ir U_{ref+} turi būti precizinės įtampos, šių įtampų šaltinis turi būti netriukšmingas. Tuo tarpu, skaitmeninių lustų maitinimo įtampos gali būti triukšmingos. Keitiklio matavimo rezultatas, išmatuotas skaičius K, kai buvo matuota nežinoma įtampa U_x yra:

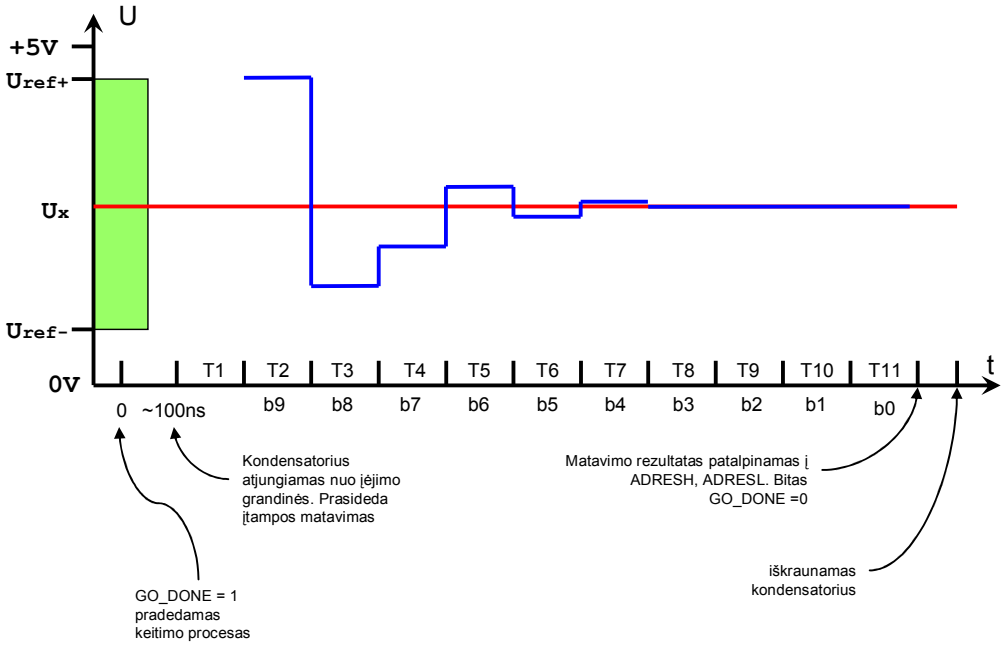
$$K = \frac{U_x - U_{ref-}}{U_{ref+} - U_{ref-}} 2^{10}, \quad \textbf{(8)}$$

ir nežinoma įtampa U_x yra:

$$U_x = U_{ref-} + \frac{K}{2^{10}}\left(U_{ref+} - U_{ref-}\right). \quad \textbf{(9)}$$

čia laikoma, kad keitiklis yra 10 bitų ir $U_{ref-}<U_x<U_{ref+}$. Jei pastaroji lygybė netenkinama, keitimo rezultatas yra mažiausia arba didžiausia vertė.

PIC18 mikrovaldikliuose yra integruotas nuoseklaus artėjimo keitiklis. Matavimo procesas vyksta taip: prie mikrovaldiklio išvado prijungta įtampa įkrauna kondensatorių, kondensatorius yra atjungiamas nuo išvado ir prijungiamas prie ***AS*** keitiklio grandinės. Toliau paeiliui yra nustatomi matavimo rezultato bitai: U_{ref-} ... U_{ref+} intervalas yra padalinamas į dvi dalis ir komparatorius nustato kuriam intervalui priklauso U_x įtampa,- nustatytas vyriausiasis bitas; gautasis intervalas vėl dalinamas pusiau ir vėl nustatoma kurioje pusėje yra U_x įtampa. Procesas kartojamas siaurinant intervalą, kol nustatomi visi bitai. 18 paveikslas iliustruoja matavimo eigą: paeiliui nustatant bitus, kodą K atitinkanti įtampa artėja prie U_x įtampos.



18 pav. Analoginio-skaitmeninio keitiklio darbo iliustracija.

Keitiklio darbą valdantys registrai truputį skiriasi senesniems ***PIC18*** šeimos mikrovaldikliams, kurių pavadinimo gale yra trys skaičiai (pvz.: ***PIC18F458***), ir naujesniems, kurių pavadinime yra keturi skaičiai (pvz.: ***PIC18F4550***). Taip pat

besivadinančių bitų paskirtis tiek senesniems, tiek naujesniems modeliams yra ta pati. ***AS*** keitiklio grandinei maitinimas yra įjungiamas su **ADON** bitu. Matavimo procesas yra pradedamas išstačius **GO/DONE=1**, kai matavimo procesas baigiasi, automatiškai **GO/DONE** bitas išsijungia **GO/DONE=1**. ***AS*** keitiklio darbui reikalingi taktiniai impulsai, jų šaltinį galima pasirinkti su **ADCS0-ADCS1** bitais.

Pasirinkto taktinių impulsų šaltinio dažnis negali viršyti tam tikros didžiausios vertės, kuri yra žymiai mažesnė už branduolio taktinių impulsų dažnį, todėl patartina taktiniu impusų šaltiniu ***AS*** keitikliui pasirinkti vidinį RC generatorių. Vidinio RC generatoriaus dažnis yra 4 MHz.

Mikrovaldiklio išvadai, kurie turi būti analoginės įtampos įėjimai, nustatomi **PCFG0-PCFG3** bitais, o kuris mikrovaldiklio išvadas turi būti prijungtas prie keitiklio pasirenkama su **CHS0-CHS3** bitais. U_{ref-} ir U_{ref+} įtampų šaltiniai pasirenkami senesniems mikrovaldiklių modeliams su **PCFG0-PCFG3** bitais, naujesniems modeliams – su **VCFG0, VCFG1** bitais.

Matavimo rezultatas yra **ADRESH** ir **ADRESL** registruose. Kadangi ***AS*** keitiklis yra 10 bitų, matavimo rezultatui talpinti reikalingi du baitai. Likę laisvi 6 bitai visada yra lygūs 0, tik šie bitai gali būti 6 jauniausi (matavimo rezultatas išlygintas pagal kairįjį kraštą) arba vyriausieji (matavimo rezultatas išlygintas pagal kairįjį kraštą). Kuris variantas bus naudojamas kontroliuoja **ADFM** bitas.

	7 bitas							0 bitas
ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	-	ADON
ADCON1	ADFM	ADCS2	-	-	PCFG3	PCFG2	PCFG1	PCSFG0

19 pav. PIC18Fxxx mikrovaldiklių analoginio-skaitmeninio keitiklio darbą valdantys registrai.

	7 bitas							0 bitas
ADCON0	-	-	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
ADCON1	-	-	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCSFG0
ADCON2	ADFM	-	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0

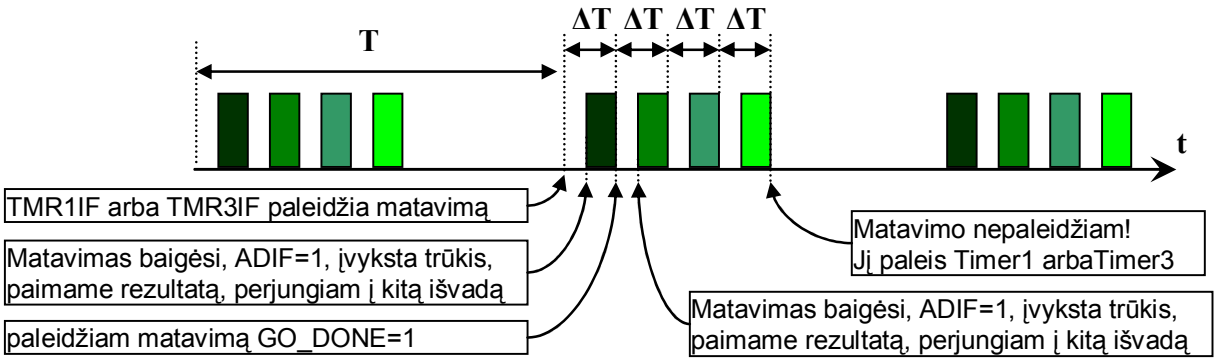
20 pav. PIC18Fxxxx mikrovaldiklių analoginio-skaitmeninio keitiklio darbą valdantys registrai.

Pavyzdys. Reikia nuskaityti prie ***PIC18F458*** mikrovaldiklio **RA0/AN0** išvado prijungtą įtampą, U_{ref-} ir U_{ref+} pasirinkti mikrovaldiklio maitinimo įtampas +5V ir GND, rezultatą patalpinti į dviejų baitų kintamąjį.

Programa:

```
void main(void)
{
      unsigned int res;     // čia bus patalpintas rezultatas

      TRISA |= 0x01;
      ADCON1 = 0b00001111; // rezultatas bus išlygintas pagal
                           // dešinįjį kraštą
                           // RA0 – analoginis įėjimas
                           // parinktos Uref- ir Uref+ įtampos
      ADCON0 = 0b11000001; // pasirinkta keitiklio taktiniai impulsai iš
                           // vidinio RC generatoriaus
                           // RA0 prijungtas prie keitiklio
                           // įjungtas maitinimas keitikliui
                              toje pačioje komandoje negalima įjungti maitinimo ir
                              paleisti matavimą, nes matavimo rezultatas bus blogas!

      ADCON0 |= 0x04;      // paleidžiam matavimą
      while(ADCON0 & 0x04);// laukiam kol baigsis matavimas
                           // patalpiname rezultatą į `res` kintamąjį
      res = ((unsigned int)ADRESH << 8) | (unsigned int)ADRESL;

      ...  // likusi programos dalis
}
```

Dar vienas AS keitiklio taikymo pavyzdys. Dažnai tenka matuoti ne vieną analoginę įtampą, o kelias. Mikrovaldiklis turi daug išvadų kurie gali tapti analoginės įtampos įėjimais, tačiau analoginis-skaitmeninis keitiklis yra tik vienas. Vienu metu įmanoma matuoti tik vieno išvado įtampą. Be to, dažnai reikia matuoti ir kelias įtampas ir griežtai vienodais laiko intervalais, taip kaip tai daro oscilografas. Kaip tai praktiškai realizuoti? Tarkime, reikia matuoti **RA0**, **RA1**, **RA2** ir **RA3** išvadų įtampas tiksliai kas T=10 ms su ***PIC18F4550*** mikrovaldikliu, veikiančiu 40 MHz dažniu. Laiko ašyje šis procesas atrodo taip:



21 pav. Keturių mikrovaldiklio išvadų įtampų matavimas vienodais laiko periodais.

Šiame paveiksle spalvoti stačiakampiai laiko ašyje žymį mikrovaldiklio trūkio paprogramės laiką, susijusį su ***AS*** keitiklio rezultato apdorojimu. „Nespalvotais“ laiko intervalais programa gali užsiimti bet kuo. Pirmąjį iš keturių matavimų paleidžia ***Timer1*** arba ***Timer3*** skaitiklis, likusius tris matavimus paleidžia trūkio paprogramė. Skaitiklio panaudojimas įgalina išlaikyti tikslų T periodą tarp matavimo serijų. Kaip įjungti automatinį ***AS*** keitiklio paleidimą? Tam reikės panaudoti ***ECCP*** modulį (kuris savo funkcijomis labai panašus į ***CCP*** modulį, nagrinėjamą sekančiame skyriuje) ir vieną iš skaitiklių: ***Timer1*** arba ***Timer3***. Naudojama ***ECCP*** modulio skaitiklio vertės palyginimo veika: **TMR1H** ir **TMR1L** (arba **TMR3H**, **TMR3L**) registrų vertė lyginama su **CCPR1H** ir **CCPR1L**, kuomet susilygina, skaitiklio turinys yra ištrinamas ir peleidžiamas ***AS*** keitiklio įtampos matavimas.

Pateikiamame pavyzdyje programa užpildo masyvus su keturių kanalų matavimo rezultatais. Tai vyksta trūkio paprogramėje. Pagrindinė programa gali sekti, kuomet matavimai pasibaigė ir apdoroti rezultatus bei pradėti matavimus iš naujo. Pasirinkime, kad kartu su ***ECCP*** moduliu naudosime ***Timer3*** skaitiklį, pasirinkime ***Timer3*** skaitiklio daliklį 1:8. Kadangi ***Timer3*** skaitiklis yra identiškas ***Timer1*** skaitikliui, tai pasinaudosime ($T = \frac{4}{F_{osc}} D\left(65535 - TMR0_{pr}\right)$, (2) formule:

$$10\,\text{ms} = \frac{4}{40000\,\text{kHz}} \cdot 8 \cdot \left(65535 - x\right), \tag{10}$$

iš čia surandame, kam lygi x vertė:

$$x = 53035 = 0\,\text{xCF}\,2\,\text{B}\ . \tag{11}$$

Taigi, **CCPR1H** ir **CCPR1L** vertės turės būti tokios: **CCPR1H**=0xCF ir **CCPR1L**=0x2B.

Pagrindinė programa su **main**() funkcija:

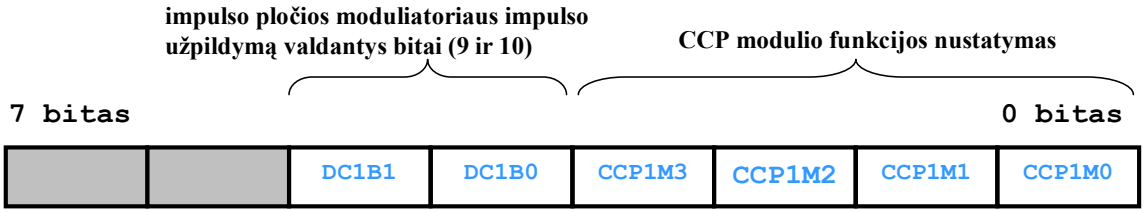
```
#define MASYVU_DYDIS  100

unsigned int ch0[MASYVU_DYDIS]; // RA0 kanalo įtampos
unsigned int ch1[MASYVU_DYDIS]; // RA1 kanalo įtampos
unsigned int ch2[MASYVU_DYDIS]; // RA2 kanalo įtampos
unsigned int ch3[MASYVU_DYDIS]; // RA3 kanalo įtampos

volatile char statusas; // 0 – vyksta masyvų pildymas
                        // 1 – masyvai užpildyti
volatile char indeksas; // elemento masyve indeksas
volatile char kanalas;  // matuojamas kanalas 0,1,2,3
volatile unsigned int res; // tarpiniams skaičiavimams

void main(void)
{
    indeksas = kanalas = statusas = 0; // pradinės vertės
    ADCON2 = 0b10000111; // taktinių impulsų šaltinis keitikliui –
                         // vidinis RC generatorius
    ADCON1 = 0b00001011; // RA0,RA1,RA2,RA3 – analoginiai įėjimai
    ADCON0 = 0b00000001; // įjungiam maitinimą ir 0 kanalą
    CCPR1H = 0xCF;       // vertė, su kuria lyginama skaitiklio vertė
    CCPR1L = 0x2B;
    CCP1CON = 0b00001011; // įjungiam CCP modulį
    TMR3H = 0;            // pradinė skaitiklio vertė
    TMR3L = 0;
    TMR3CON = 0b00111001; // įjungiam skaitiklį darbui su ECCP moduliu
    for(;;)
    {
        if(statusas) // ar baigėsi matavimai
        {
            ...      // apdorojam matavimo rezultatus
            TMR3H = TMR3L = 0;
            indeksas = kanalas = statusas = 0;
            T3CONbits.TMR3ON = 1;  // paleidžiam skaitiklį,
                                   //pradedam matavimus iš naujo
        }
        ... // programa kažkuo užsiima
    }
}
```

Trūkio paprogramė:

```
#pragma code
#pragma interrup int_isr
void int_isr (void)
{
    res = ((unsigned int)ADRESH << 8) | (unsigned int)ADRESL;
    switch(kanalas)
    {
        case 0:
            ch0[indeksas] = res; // patalpinam rezultatą į masyvą
            kanalas++;
            ADCON0bits.CHS1 = 0; // prijungiam AD keitiklį prie RA1
      išvado
            ADCON0bits.CHS0 = 1;
            ADCON0bits.GO_DONE = 1; // paleidžiam matavimą
            break;
        case 1:
            ch1[indeksas] = res;
            kanalas++;
            ADCON0bits.CHS1 = 1; // prijungiam AD keitiklį prie RA2
      išvado
            ADCON0bits.CHS0 = 0;
            ADCON0bits.GO_DONE = 1; // paleidžiam matavimą
            break;
        case 2:
           ch2[indeksas] = res;
           kanalas++;
           ADCON0bits.CHS0 = 1; // prijungiam AD keitiklį prie RA3 išvado
           DCON0bits.CHS1 = 1;
           ADCON0bits.GO_DONE = 1; // paleidžiam matavimą
           break;
        case 3:
           ch3[indeksas] = res;
           kanalas = 0;
           indeksas++;
           ADCON0bits.CHS0 = 0; // keitiklis prijungiamas prie RA0 išvado
           ADCON0bits.CHS1 = 0;
                  // matavimo nepaleidžiam, jį paleis ECCP modulis
           break;
    }
    if(indeksas >= MASYVU_DYDIS) // masyvai uzpildyti?
    {
        statusas = 1;            // informuojam pagrindinę programą
        T3CONbits.TMR3ON = 0;    // stabdom skatiklį, kad daugiau
                                 // nevyktų matavimai
    }
    PIR1.ADIF = 0;      // būtina ištrinti ADIF bitą
}

#pragma code low_vector=0x808       // nurodom, kad funkcija
void interrupt_at_low_vector(void)  // turi prasidėti nuo
{                                   // 0x808 ląstelės
    _asm
    goto int_isr     // 0x808 ląstelėje bus tik goto komanda
    _endasm
}
#pragma code
```

1.11 *CCP* modulis

Šio modulio pavadinimas – tai atliekamų funkcijų santrumpa (angl. Capture, Compare, PWM): įsiminti, lyginti ir impulso pločio moduliacija (angl. PWM – Pulse Width Modulator). Šis modulis turi 16 bitų registrą, kuriame gali „įsiminti“ arba su juo „palyginti“ vieno – ***Timer1*** arba ***Timer3*** – skaitiklio vertę, vykdyti impulso pločio moduliaciją kartu su ***Timer2*** skaitikliu. ***CCP*** modulio darbą valdo `CCP1CON` registras.



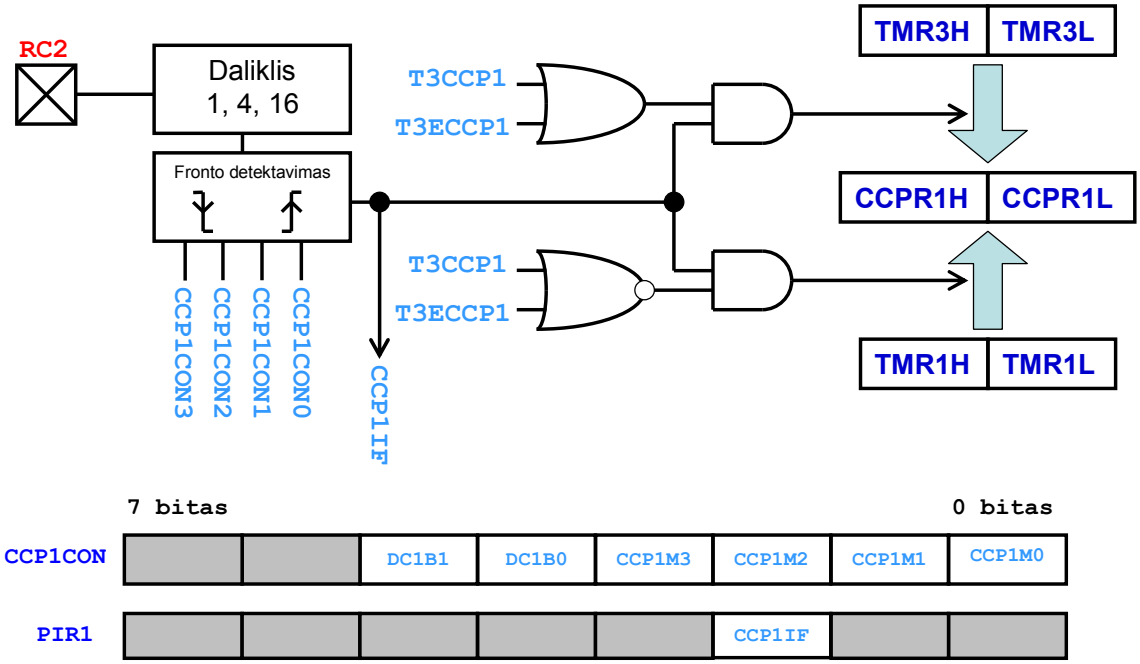
22 pav. CCP1CON registras.

Kai kuriuose ***PIC18*** šeimos mikrovaldikliuose yra ne vienas ***CPP*** modulis, o keli. Antrasis modulis turi daugiau funkcijų ir yra vadinamas ***ECCP*** moduliu.

Dabar detaliau apie ***CCP*** modulio funkcijas.

1.11.1. CCP modulis: skaitiklio vertės kopijavimo funkcija

Timer1 arba ***Timer3*** skaitiklių vertė (t.y. **TMR1H**, **TMR1L** arba **TMR3H**, **TMR3L** registrai) gali būti nukopijuota į **CCPR1L**, **CCPR1H** registrus. Kopijavimą inicijuoja kylantis arba krintantis impulso frontas ant **RC2** mikrovaldiklio išvado.



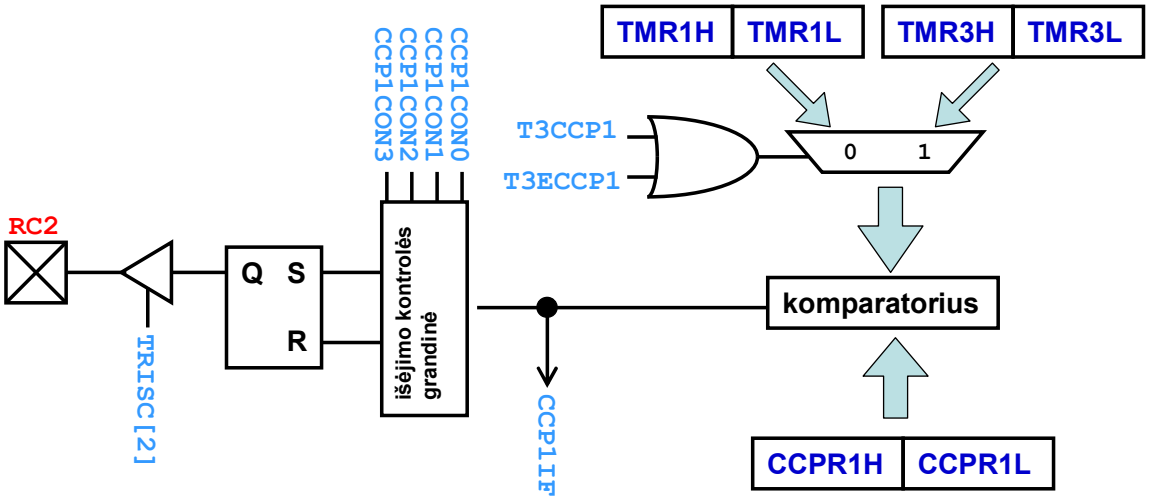
23 pav. CCP modulio schema, skaitiklio vertės kopijavimo funkcija.

Be to, gali būti nustatyta, kad skaitiklio vertė kopijuojama kas ketvirtą arba kas šešioliktą kartą. Kuris impulso frontas incijuoja kopijavimą, daliklio vertė yra nustatoma su **CCP1CON0**, **CCP1CON1**, **CCP1CON2** ir **CCP1CON3** bitais. Kurio skaitiklio: ***Timer1*** ar ***Timer3*** vertė bus kopijuojama, nustatoma su **T3CCP1** bitu. Kuomet įvyksta kopijavimas, įjungiamas **CCP1IF** bitas,- tai gali būti panaudota trūkio aktyvavimui.

Ši ***CCP*** modulio funkcija, kuomet yra kopijuojama skaitiklių vertė, gali būti panaudota tiksliam laiko intervalų tarp kelių impulsų matavimui.

1.11.2. ***CCP*** *modulis: skaitiklio vertės palyginimas*

CCP modulis lygina skaitklio ***Timer1*** arba ***Timer2*** vertę su registrų CCPR1H/CCPR1L verte. Kurio iš skaitiklių vertė yra naudojama palyginimui, nustato T3CCP1 ir T3ECCP1 bitai: jei T3CCP1=1 arba T3ECCP1=1, tuomet CCPR1H/CCPR1L vertė yra lyginama su ***Timer3*** skaitiklio verte, priešingu atveju – su ***Timer1*** skaitiklio verte. Esant sutapimui, įjungiamas bitas CCP1IF ir gali būti pakeista išvado RC2 būsena: į aukštą lygį, į žemą lygį, į priešingą lygį, nei buvo išstatytas prieš tai.



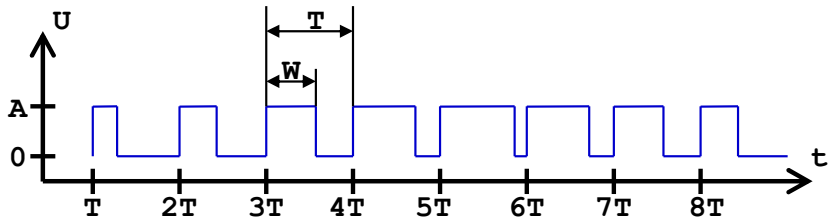
24 pav. CCP modulio schema, skaitiklio vertės palyginimas funkcija.

Ši ***CCP*** modulio funkcija suteikia galimybę generuoti pavienį impusą ar impulsų seką su tiksliai nustatyta impulso trukme.

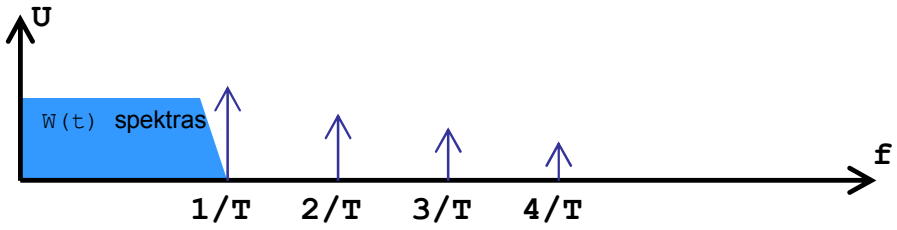
1.11.3. *CCP modulis: impulso pločio moduliacija*

Daugumoje mikrovaldiklių skaitmeninio-analoginio (*SA*) keitiklio nerasime, tačiau impulso pločio moduliatorius yra integruojamas į beveik visus. Taip yra neatsitiktinai, nes tai ką galima gauti su *SA* keitikliu – analoginę įtampą – galima ir su impulso pločio moduliatoriumi.

Impuslo pločio moduliacija: impulso periodas yra patovus, keičiasi impulso užpildymas (žr. 25 pav.).

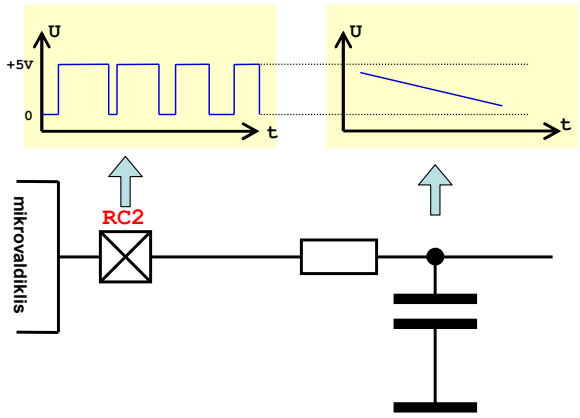


25 pav. Impulso pločio moduliacija.



26 pav. Stačiakampių impulsų spektras.

Jei pažiūrėtumėme tokio signalo Furje spektrą, pamatysime harmonikas, kurios visada yra stačiakampio signalo spektre, o žemų dažnių srityje bus funkcijos, kuria buvo moduliuotas impulso plotis, spektras. Norint išskirti tik $W(t)$ funkciją, reikalingas filtras, praleisiantis tik žemų dažnių signalą. Paprasčiausias yra RC filtras. Pažiūrėkite į 27 pav., kuris iliustruoja tai, ką pamatytumėte su oscilografu.



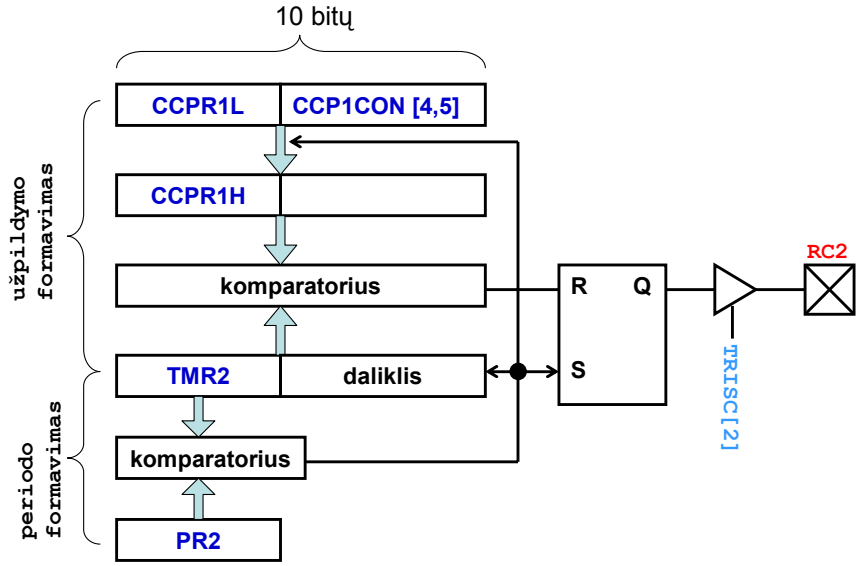
27 pav. RC filtras *W*(t) signalui išskirti.

Prijungę oscilografo gnybtą prie mikrovaldiklio išvado, matysite stačiakampių impulsų seką. Jų amplitudė sutampa su mikrovaldiklio maitinimo įtampa +5 V. O už RC filtro matysite signalą, kintantį $W(t)$ dėsniu, kurio įtampa yra iš intervalo 0 ... 5 V. Kokiu dikretiškumu galima keisti analoginę įtampą? Tokiu diskretiškumu, kokiu yra valdomas impulso užpildymas $W(t)$. ***PIC18*** šeimos mikrovaldikliuose impulso užpildymo nustatymui naudojami 10 bitų.

Impulso pločio moduliaciją vykdo ***CCP*** modulis kartu ***Timer2*** skaitikliu. ***Timer2*** skaitiklio kitiems tikslams jau nebegalėsite naudoti (žr. 28 pav.). Impulsų periodą formuoja ***Timer2*** skaitiklis:

$$T_{PWM} = \frac{4}{F_{osc}} D_1 \left(PR2 + 1\right), \tag{12}$$

čia F_{osc} – mikrovaldiklio taktinis dažnis, D_1 – 1-ojo daliklio vertė, **PR2** registras – didžiausioji ***Timer2*** skaitiklio vertė. Skirtingai nuo 1.9 skyriuje nagrinėtų ***Timer2*** skaitiklio funkcinių galimybių, dabar ***Timer2*** veikia kaip 10 bitų skaitiklis. Skaitiklio vertė didėja iki didžiausios vertės ir persiverčia į 0. Didžiausią vertę nustato **PR2** registras: **PR2** vertė lyginama su vyriausiais 8 skaitiklio bitais.



28 pav. CCP modulio schema, impulso pločio moduliacijos funkcija.

Impuso užpildymo *W* formavimas vyksta tokiu būdu: 10 bitų vertė yra nustatoma **CCPR1L** ir **CCP1CON** registruose; *W* vertė iš šių registrų ne iš karto paveikia impulso pločio formavimą, o tik tuomet kai baigiasi periodas, - periodo pabaigoje yra padaroma kopija ir įrašoma į vidinius registrus. Tokiu būdu yra išvengiama galimo *W* keitimo impulso viduryje. Persivertęs ***Timer2*** skaitiklis paduoda signalą į trigerio „S“ įėjimą, o ant **RC2** išvado atsiranda loginis 1. Loginis 1 išlieka tol, kol ***Timer2*** skaitiklio vertė pasiekia užpildymo *W* vertę, tuomet yra formuojamas signalas į „R“ trigerio įėjimą, o **RC2** išėjime atsiranda loginis 0. Gali atsitikti taip, kad nebus įjungtas loginis 0, o taip gali atsitikti tada, kuomet ***Timer2*** skaitiklio vertė nepasiekia *W* vertės, nes ***Timer2*** skaitiklio vertė didėja nuo 0 iki didžiausios, nustatomos su **PR2** registru. Kitaip tariant, moduliatoriaus skiriamoji geba tokiu atveju būtų ne 10 bitų, o mažiau ir ši skiriamoji geba priklauso nuo **PR2** registro:

$$S = \log_2\left(\frac{F_{osc}}{F_{PWM}}\right) = 2 + \log_2\left(PR2 + 1\right), \qquad \textbf{(13)}$$

čia *S* yra skiriamoji geba bitais, F_{osc} – mikrovaldikio taktinis dažnis, F_{PWM} – impulso pločio moduliatoriaus dažnis. Jei **PR2**=0xFF moduliatoriaus skiriamoji geba *S*=10, jei **PR2**=0x3F tai *S*=8, jei **PR2**=0x1F tai *S*=7. Paprastai yra pageidautina išgauti kuo didesnį moduliatoriaus dažnį (kuo trumpesnį periodą) ir kuo didesnę moduliatoriaus skiriamąją gebą, tačiau, kaip seka iš pateiktų formulių, šie pageidavimai yra prieštaraujantys vienas kitam. Tenka rinktis kompromisinį variantą.

Pavyzdys. Prie **RC2** mikrovaldiklio išvado yra prijungtas filtras – RC grandinė, norime gauti analoginę įtampą, proporcingą impulso užpildymui *W*, skiriamoji geba – 8 bitai.

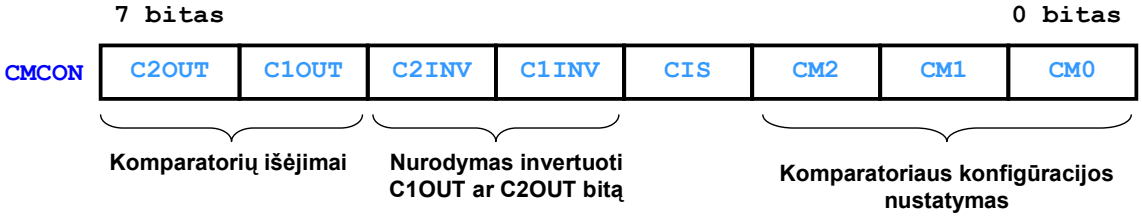
Skiriamoji geba 8 bitai bus, jei 0xFF≤**PR2**≤0x3F. Jei pasirinktume didesnę **PR2** vertę, tarkime **PR2**=0xFF, tai gautume mažesnį impulsų dažnį ir analoginės įtampos kitimo ribos už RC filtro būtų ne visas mikrovaldiklio maitinimo įtampų diapazonas. Taigi, pasirenkam **PR2**=0x3F ir daliklį D_1=1.

Programa:

```
char KeistiIpulsoUzpildyma(void); // jūsų funkcija, skaičiuojanti koks
                                  // turi būti impulso užpildymas

void Laukti(unsigned int t);      // vėlinimo funkcija

void main(void)
{
    PR2 = 0x3F;
    T2CON = 0b00000100;   // dalikliai 1:1, ijungiam skaitiklį
    TRISC &= ~(1<<2);     // įjungiam RC2 išvadą kaip skaitmeninį išėjimą
    CCP1CON = 0b00001100; // įjungiam CCP modulio PWM moduliatorių
    for(;;)
    {
        CCPR1L = KeistiIpulsoUzpildyma(); // pakeičiamas impulso uzpildymas
        Laukti(10); // prieš kitą impulso užpildymo pakeitimą
                    // reikia palaukti, kol nusistovės įtampa ant RC
                    // grandinės kondensatoriaus
    }
}
```

1.12 Komparatoriai

PIC18 šeimos mikrovaldikliuose yra integruoti du komparatoriai, kurių įėjimai/išėjimai yra susieti su skirtingais išvadais skirtinguose mikrovaldiklių modeliuose, tačiau funkcinės galimybės yra tos pačios. Komparatorius nuo operacinio stiprintuvo skiriasi tuo, jog komparatoriaus išėjime yra loginis „1“ arba „0“, priklausomai nuo įtampų skirtumo ženklo įėjimuose, t.y. komparatorius dirba netesinėje veikoje. Tuo tarpu operacinio stiprintuvo išėjime įtampa yra proporcinga įėjimo įtampų skirtumui, t.y. operacinis stiprintuvas dirba tiesinėje veikoje.

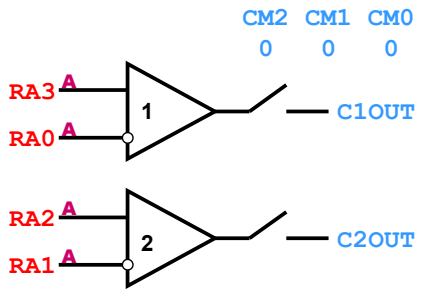


29 pav. Komparatoriaus darbą kontroliuojantis CMCON registras.

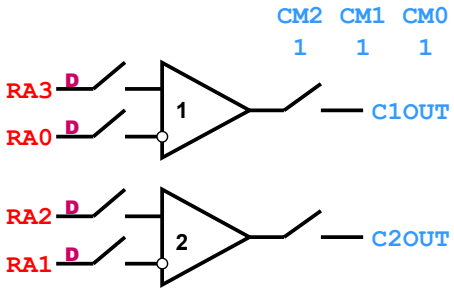
Komparatoriaus darbą kontroliuoja **CMCON** registras: bitai **C1OUT** ir **C2OUT** reiškia komparatoriaus išėjimų būseną, kuri gali būti invertuota su **C1INV** ir **C2INV** bitais, vieną iš galimų komparatoriaus veikų nustato **CM0**, **CM1** ir **CM2** bitų kombinacija, bitas **CIS** parenka įėjimus kai **CM2**,**CM1**,**CM0**=011. Komparatorių išėjimų būsenos atsispindi **C1OUT** ir **C2OUT** bituose, taipogi, komparatorių išėjimai gali būti sujungti su mikrovaldiklio išvadais. Komparatoriaus įėjimai yra analoginės įtampos įėjimai, išėjimai (jei jie sujungti su konkrečiu mikrovaldiklio išvadu) – skaitmeniai išvadai. Atkreiptinas dėmesys į tai, jog yra keletas skirtumų tarp ***PIC18Fxxx*** ir naujesnių ***PIC18Fxxxx*** modelių:

- ***PIC18Fxxx*** mikrovaldikliuose įjungus vieną iš komparatoriaus veikų, išvadai, kurie yra komparatorių įėjimai, automatiškai tampa analoginės įtampos įėjimais. Tuo tarpu ***PIC18Fxxxx*** mikrovaldikliuose taip nėra,- reikia papildomai nustatyti **ADCON1** registre, kurie išvadai bus analoginės įtampos įėjimais;
- Po „reset“ signalo ***PIC18Fxxx*** mikrovaldikliuose komparatorių įėjimai yra įjungti, o išėjimai – išjungti (būsenos bitukai 000). ***PIC18Fxxxx*** mikrovaldikliuose po „reset“ signalo komparatoriai pilnai yra išjungti (būsenos bitukai 111), tačiau mikrovaldiklio išvadus, kurie naudojami kaip įėjimai komparatoriams, ar jie yra analoginės įtampos įėjimai kontroliuoja **ADCON1** registras, kurio vertė po „reset“ signalo taip pat nustato, kad su komparatoriaus įėjimais susiję išvadai tampa analoginės įtampos įėjimais.

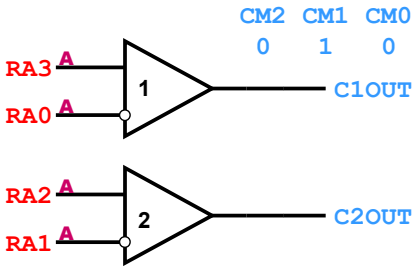
Galimas komparatorių konfigūracijas ***PIC18F4550*** mikrovaldiklyje iliustruoja 30 paveikslas, kaip jau minėta, viskas tinka ir kitiems mikrovaldiklių modeliams tik komparatorių įėjimai/išėjimai gali būti susieti su kitais mikrovaldiklio išvadais. Konfigūracija **CM2**,**CM1**,**CM0**=011 yra ypatinga tuo, jog tik šioje konfigūracijoje **CIS** bitukas leidžia komutuoti invertuojančius komparatoriaus įėjimus prie skirtingų mikrovaldiklio išvadų, o prie neinvertuojančių koparatoriaus įėjimų yra prijungta įtampa iš valdomo vidinio įtampos šaltinio. Jis vadinamas atraminės įtampos komparatoriams šaltiniu, tačiau gali būti panaudotas ir visai kitiems tikslams (apie tai – sekančiame skyriuje).



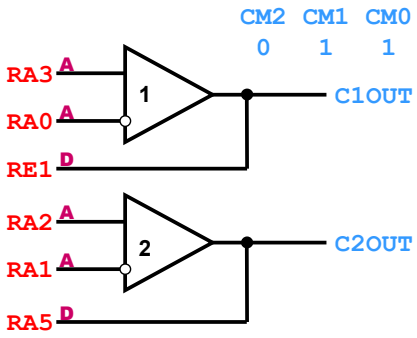
komparatorių išėjimai išjungti



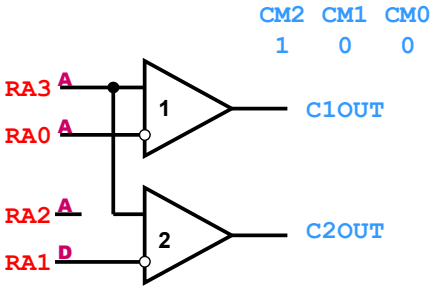
komparatoriai išjungti



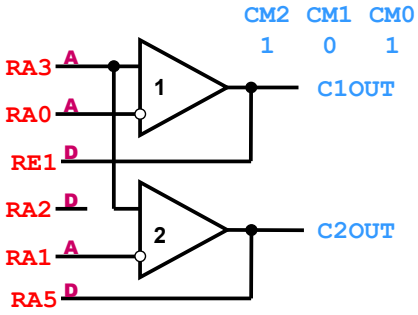
du nepriklausomi komparatoriai



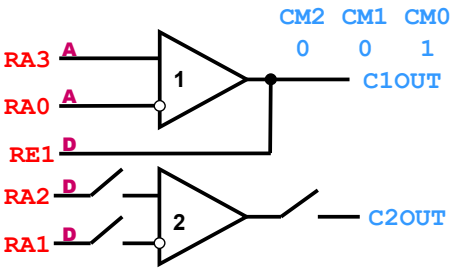
du nepriklausomi komparatoriai su išėjimais



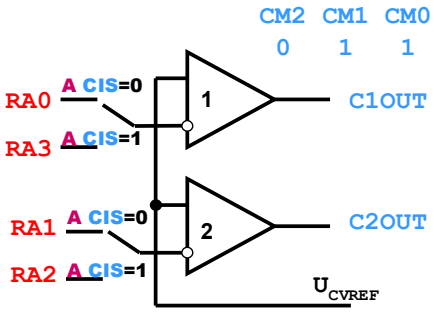
du komparatoriai su bendru įėjimu



du komparatoriai su bendru įėjimu ir išėjimais



vienas komparatorius su išėjimu

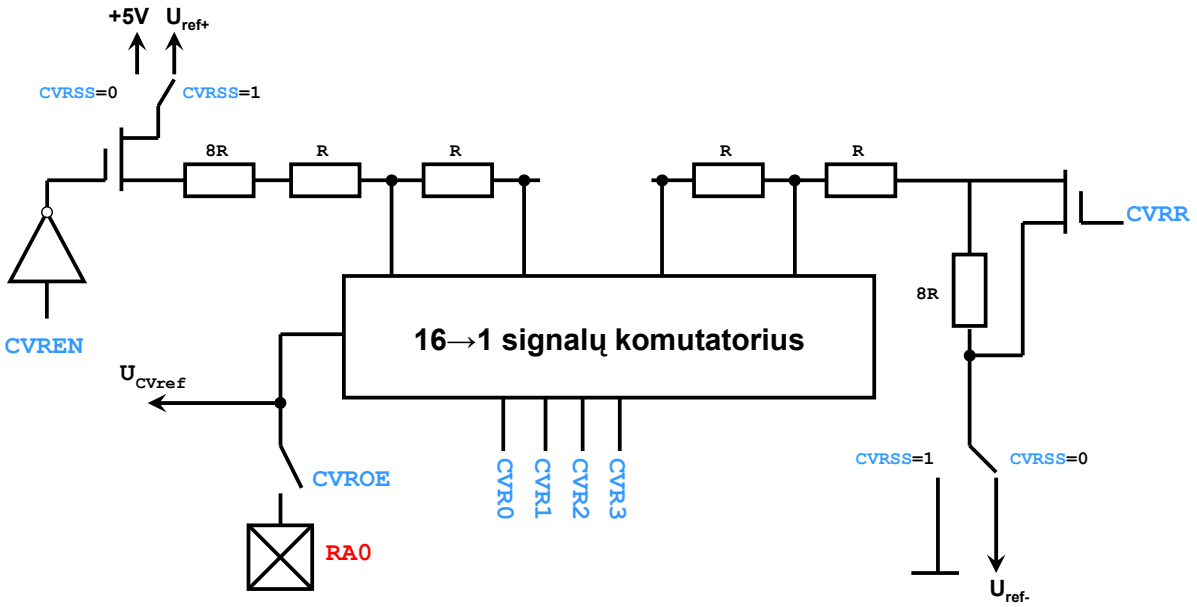


du nepriklausomi komparatoriai su išėjimais

30 pav. Komparatorių skirtingos konfigūracijos PIC18F4550 mikrovaldiklyje.

1.13 Atraminės įtampos šaltinis komparatoriams

Atraminės įtampos šaltinis komparatoriams, iš esmės, yra įtampą dalinanti rezistorių grandinė, kurios dalinimo vieną iš 16 galimų verčių galima pasirinkti. Nors ši grandinė ir yra visų pirma skirta komparatoriams, tačiau gali būti naudojama atskirai nuo komparatorių ir visai kitiems tikslams.



31 pav. Atraminės įtampos šaltinio komparatoriams grandinė PIC18F458 mikrovaldikyje.

Atraminės įtampos šaltinio konstrukcijos pagrindas yra rezistorių grandinė (žr. 31 pav.), kurios vienas galas yra pasirinktinai prijungtas prie maitinimo įtampos arba prie U_{ref+} šaltinio (įtampa, prijungta prie tam tikro mikrovaldiklio išvado, plačiau – ***AS*** keitiklio aprašyme), o kitas galas – prie bendro išvado („žemės“) arba U_{ref-} įtampos šaltinio. Nuo kurios rezistorių grandinės vietos įtampa taps U_{CVREF}, pasirenkama su 16→1 analoginių signalų komutatoriumi, o šis valdomas **CVR0-CVR3** bitais. Įtampos šaltinio grandinės darbą valdo **CVRCON** registras. **CVREN** bitas įjungia maitinimą visai grandinei. U_{CVREF} įtampa gali būti išvesta į mikrovaldiklio išvadą, jei **CVROE**=1 ir atitinkamas išvadas nustatytas kaip analoginės įtampos įėjimas. Mikrovaldiklių skirtingiems modeliams skiriasi išvadas, į kurį išvedama U_{CVREF} įtampa.

U_{CVREF} įtampa su **CVR0-CVR3** bitais išreikštu skaičiumi yra susijusi tokia formule:

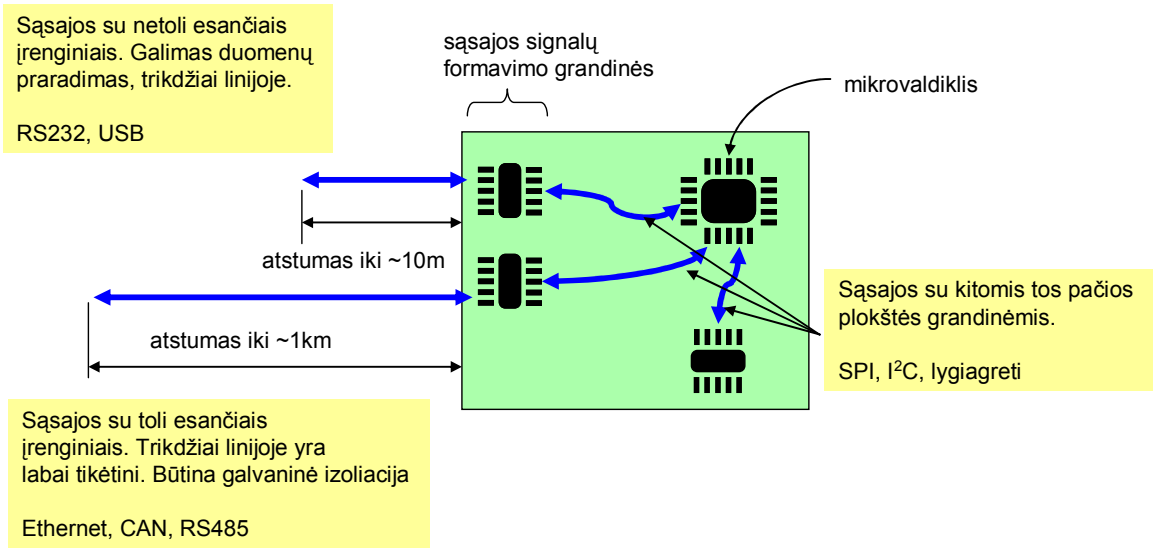
$$U_{CVREF} = \begin{cases} U_0 \dfrac{CVR}{24}, & jei\ CVRR = 1 \\ U_0 \dfrac{1}{4} + U_0 \dfrac{CVR}{32}, & jei\ CVRR = 0 \end{cases}, \tag{14}$$

kur $0 \le CVR \le 15$ yra **CVR0-CVR3** bitais išreikštas skaičius, įtampa U_0 yra

$$U_0 = \begin{cases} U_{VREF+} - U_{VREF-}, & jei\ CVRSS = 1 \\ maitinimo\ įtampa, & jei\ CVRSS = 0 \end{cases}. \tag{15}$$

3. Laidinės sąsajos

Mikrovaldiklis tiesiai ar su keliais papildomais elektroniniais lustais (skirtais suderinti įtampų lygius) gali būti jungiamas prie laidinių magistralių, per kurias vyksta duomenų mainai su kitais schemos elementais ar su nutolusiais įrenginiais. Mikrovaldikliuose būna integruota daug ir įvairių sąsajų, dažniausiai naudojamos ir į beveik visus mikrovaldiklius integruojamos sąsajos yra: ***I²C***, ***SPI***, nuoseklios asinchroninės (***RS232***, ***RS485*** ir panašios), rečiau pasitaikančios: ***CAN***, ***USB***, ***Ethernet*** ir kitos.



32 pav. Įvairios sąsajos elektroninėje plokštėje.

Sąsajos apibūdinamos tokias parametrais kaip fizinė duomenų perdavimo sparta (naudingų duomenų perdavimo sparta daugeliu atveju yra nuo daug faktorių priklausantis dydis ir netgi konkrečiai sąsajai gali būti labai skirtingas), didžiausias atstumas, klaidų aptikimo ir taisymo galimybės, įrenginių skaičius. Šių ir kitų parametrų suvestinė dažniausiai naudojamoms sąsajos yra pateikta 2 lentelėje.

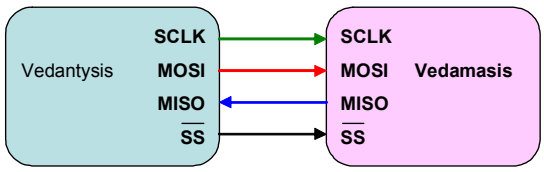
2 lentelė. Įvairių laidinių sąsajų palyginimas

sąsaja	**duomenų perdavimo sparta**	**įrenginių skaičius segmente**	**kabelio ilgis**
ethernet	10/100/1000 Mb/s	2	100 m, iki 4 kartotuvų, tuomet ilgis iki 500 m (5 segmentai)
RS232	0.24 - 115.2 kb/s	2	15 m (pagal standartą), realiai ~100 m
RS485	0.24 – 1000 kb/s	~256	1.3 km kai 64 kb/s su kartotuvu žymiai daugiau
CAN	10-1000 kb/s	~128	1 km kai 40 kb/s 40 m kai 1 Mb/s
USB	1.5/12/480 Mb/s	2 128 su USB hub	5 m. Iki 6 kartotuvų, tuomet ilgis iki 35 m (7 segmentai)

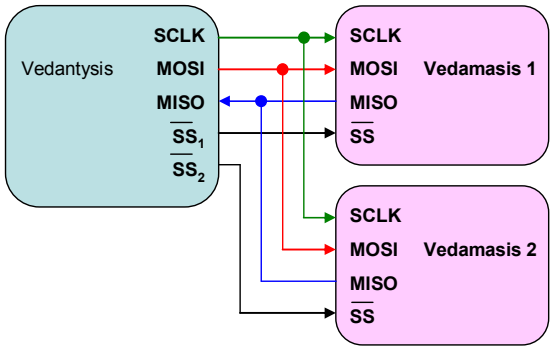
1.14 *SPI* sąsaja

SPI sąsaja pirmą sykį buvo panaudota Motorola įrenginiuose, šiuo metu į beveik visus mikrovaldiklius yra integruojami šią sąsają palaikantys elektroniai mazgai.

SPI yra paprasčiausia sinchroninė nuoseklaus duomenų perdavimo sąsaja, skirta keistis duomenimis tarp dviejų įrenginių, kurių vienas yra valdantysi (angl. master), kitas – valdomasis (angl. slave). Duomenys gali būti siunčiami iš karto dviem kryptimis. Nėra įrenginių adresacijos. Jei yra daugiau nei vienas vedamasis, tenka panaudoti papildomas mikrovaldiklio kojas ir papildomai numatyti kelių valdamųjų valdymą programoje. ***SPI*** sąsajoje nenumatytas patvirtinimas, kad duomenys gauti. ***SPI*** sąsaja nenumato, kad duomenis gaunantis įrenginys gali nespėti priimti duomenų ir reikia palaukti. Jeigu jums prisireikė aukščiau išvardintų funkcijų, kurių nėra ***SPI*** sąsajoje, tuomet turite konstruoti aukštesnio lygio protokolą.



33 pav. SPI sąsajos įrenginiai, kuomet yra tik vienas vedantysis ir vienas vedamasis įrenginys.



34 pav. SPI sąsaja, kuomet yra keletas vedamųjų įrenginių.

SPI sąsajos paskirtis: mikrovaldiklis keičiasi duomenimis su (dažniausiai toje pačioje plokštėje įlituotais) įrenginiais, kuomet duomenų srautas nėra didelis. Tokių įrenginių pavyzdžiai: ***EEPROM*** atmintis, lėti ***AS*** ir ***SA*** keitikliai, skaitmeniai potenciometrai. Didžiausia duomenų sparta gali siekti 10 Mb/s, kartais ir daugiau.

SPI sąsajoje yra naudojamos bent dvi signalinės linijos: viena jų perduodami sinchro impusai, kita – duomenys. Jei duomenys perduodami dviem kryptimis, tuomet reikalingos trys signalinės linijos. Jei reikalingas valdomojo įrenginio aktyvavimo signalas – prisideda dar viena linija.

SPI sąsajos signalai:

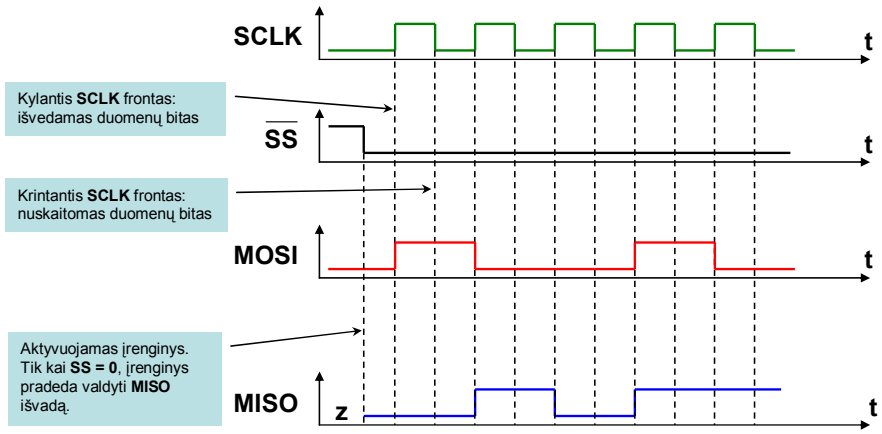
- ***SCLK*** – sinchro impulsai;
- ***MOSI*** (angl. Master Output, Slave Input) – duomenys perduodami iš valdančiojo įrenginio;
- ***MISO*** (angl. Master Input, Slave Output) – duomenys perduodami iš valdomojo įrenginio;
- ***SS*** – valdomojo įrenginio aktyvavimo signalas.

Duomenys išvedami nuosekliuoju būdu, sinchroniškai su ***SCLK*** impulsai. Kokiu būdu įvedami ir išvedami duomenys turi būti sinchronizuomaji su ***SCLK*** signalu yra nustatoma konfigūruojant mikrovaldiklio aparatūrinę grandinę. Svarbu, kad duomenis siunčiantis ir priimantis įrenginiai būtų vienodai suderinti sinchronizuotis pagal ***SCLK*** signalą.

Sinchronizavimo variantai:

- duomenys išvedami, kai yra ***SCLK*** signalo kylantis arba krintantis frontas;
- duomenys nuskaitomi, kai yra ***SCLK*** signalo kylantis arba krintantis frontas;
- kai kurie mikrovaldikliai gali būti sukonfiguruoti taip, kad duomenys būtų skaitomi per aukšto arba žemo ***SCLK*** impulso lygio vidurį.

Pavyzdyje (žr. 35 pav.) yra pateiktas sinchronizavimo variantas, kai duomenys iš valdančiojo įrenginio yra išvedami kartu su kylančiu ***SCLK*** frontu, įvedami – su krintančiu frontu.



35 pav. SPI sąsajos impulsų pavyzdys: duomenys iš valdančiojo įrenginio yra išvedami kartu su kylančiu SCLK frontu, įvedami – su krintančiu frontu.

Aparatūrinė ***SPI*** mikrovaldiklio grandinė vienu metu ir siunčia ir priima duomenis, nors duomenis reikia perduoti tik viena kryptimi. Kurie iš šių duomenų: siunčiami ar priimami, yra reikalingi ir turi prasmę, sprendžia programuotojas, rašydamas programą.

SPI* sąsaja *PIC18* mikrovaldikliuose.** ***PIC18 mikrovaldikliuose ***SPI*** sąsaja valdo ***MSSP*** (angl. Master Synchronous Serial Port) modulis, kurio dar viena funkcija yra ***I²C*** sąsaja. Mikrovaldikliuose yra realizuotos tiek ***SPI*** valdančiojo, tiek valdomojo įrenginio funkcijos, realizuoti visi aukščiau aptarti ***SPI*** sąsajos signalai, tik ***MOSI*** signalas vadinamas ***SDO***, o ***MISO*** signalas vadinamas ***SDI***. ***MSSP*** modulio darbą valdo **SSPCON1** registras, modulio darbo būsenos atsispindi **SSPSTAT** registre, o duomenys yra siunčiami/priimami per **SSPBUF** registrą. Pagrindiniai parametrai, kuriuos reikia nustatyti:

- pasirinkti, ar mikrovaldiklis bus valdantysis ar valdomasis įrenginys su **SSPM0-SSPM3** bitais, esančiais **SSPCON1** registre;
- sąsajos sinchro impulsų dažnį, jis nustatomas **SSPM0-SSPM3** bitais, esančiais **SSPCON1** registre. Galima pasirinkti arba procesoriaus taktinių impulsų šaltinį su dalikliu 1:4, 1:16, 1:64, arba sinchro impulsų šaltinis gali būti ***Timer2*** išėjimas;
- sinchro impulsų lygis, esant neaktyviai sąsajai (nevyksta duomenų mainai). Jį kontroliuoja **CKP** bitas;
- duomenys bus perduodami kylant ***SCLK*** frontui ar krintant. Tai kontroliuoja **CKE** bitas;
- duomenų įvedimas: duomenų bitas nuskaitomas išvedamo bito pabaigoje ar per vidurį. Tai kontroliuoja **SMP** bitas.

Pavyzdys. Tarkime prie mikrovaldiklio, kurio taktinis dažnis 40 MHz, yra prijungtas išorinis analoginis-skaitmeninis keitiklis ***MCP3204***, jo pagrindinės charakteristikos: skiriamoji geba 12 bitų, 4 kanalai, taktinių impulsų šaltinis – ***SPI*** sąsajos sinchro impulsai. Iš šių dviejų ***SPI*** sąsajos įrenginių, mikrovaldiklis yra valdantysis. Tarkime, valdomasis įrenginys – ***MCP3204*** yra aktyvuojamas signalu, kuris yra prijungtas prie mikrovaldiklio ***RC0*** išvado. ***MCP3204*** keitikliui kuomet, nevyksta duomenų mainai, ***SPI*** sinchro impulsų lygis turi būti aukštas (nustatom **CKP**=1), keitiklis duomenų bitą išveda kartu su krintančiu ***SCLK*** frontu (nustatom **CKE**=1), įveda – kartu su kylančiu frontu (nustatom **SMP**=1). Didžiausias keitiklio ***SCLK*** sinchro impulsų dažnis yra 2 MHz, todėl su **SSPM0-SSPM3** bitais pasirenkam ***SCLK*** dažnis *F*osc/64, t.y. 625 kHz.

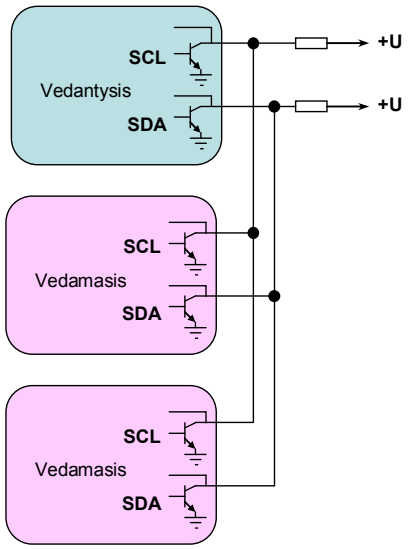
Programa:

```
unsigned int ReadMCP3204(char ch);

void main(void)
{
    unsigned int itampa;
    TRISC &= ~1;          // nustatom RC0 kaip skaitmeninį įšėjimą
    PORTCbits.RC0 = 1;    // nustatom logini 1 ant RC0 išvado
    SSPSTAT = 0b11000000; // SPI sąsajos nustatymas su aptartais parametrais
    SSPCON  = 0b00110010;
    for(;;)
    {
        itampa = ReadMCP3204(0); // matuojam 1 kanalo įtampą
        // ...
        itampa = ReadMCP3204(1); // matuojam 2 kanalo įtampą
        // ...
        itampa = ReadMCP3204(2); // matuojam 3 kanalo įtampą
        // ...
        itampa = ReadMCP3204(4); // matuojam 4 kanalo įtampą
        // ...
    }
}

    // funkcija, kuri nustato keitiklio parametrus ir
    // nuskaito išmatuotą įtampą.
    // ch – kanalo numeris 0,1,2 arba 3
unsigned int ReadMCP3204(char ch)
{
    unsigned char res_hi,res_lo;
    unsigned int res;

    PORTCbits.RC0 = 0;  // aktyvuojam keitiklį

    // reikalinga pauzė keitikliui pasiruošti darbui

    SSPBUF = 0b00000110;       // perduodam keitiklio nustatymus
    while( !SSP2STATbits.BF ); // laukiam kol baigsis siuntimas

    SSPBUF = (ch << 6) & 0x0F; // perduodam kanalo nr ir priimam vyriausiąjį
                               // išmatuotos įtampos kodą
    while( !SSP2STATbits.BF ); // laukiam kol baigsis siuntimas
    res_hi = SSPBUF;           // priskiriam kintamajam

    SSPBUF = 0;
    while( !SSP2STATbits.BF ); // laukiam kol baigsis siuntimas
    res_lo = SSPBUF;           // priskiriam kintamajam

    // konvertuojam į baitų kintamąjį:
    res = ((unsigned int)res_hi << 8) | (unsigned int)res_lo;

    PORTCbits.RC0 = 1;  // išjungiam keitiklį

    return res;  // grąžinam matavimo rezultatą
}
```

1.15 I^2C sąsaja

I^2C (angl. inter-integrated circuit) sąsaja 1992 m. sukūrė Philips kompanija, pirmoji šios sąsajos taikymo sritis buvo televizorių elektronika. Pradžioje buvo numatyta 100 kb/s duomenų perdavimo sparta, tačiau sparčiai populiarėjant šiai sąsajai, 1998 ir 2000 m. buvo toliau tobulinamas ***I^2C*** standartas, šiuo metu numatantis duomenų perdavimo spartą iki 3.4 Mb/s.

I^2C taikoma prie valdiklio jungiant ***EEPROM*** atminties, ***AS*** ir ***SA*** keitiklių, ***LCD*** valdymo, išplėtimo buferių lustus.



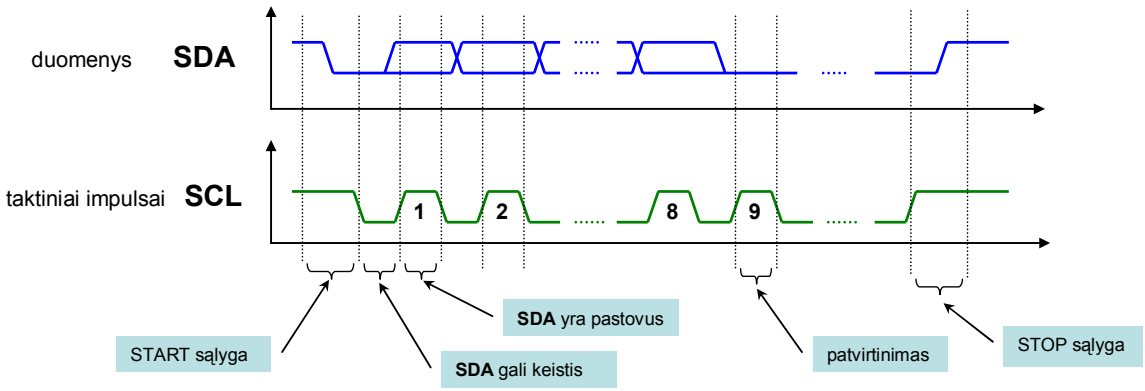
36 pav. I^2C sąsaja.

Apie ***I^2C*** galima pasakyti, kad tai yra sinchroninė nuoseklaus duomenų perdavimo sąsaja, naudojamos tik dvi linijos: viena iš jų yra perduodmi duomenys, kita – sinchro impulsai; abi linijos yra skirtos dvikrypčiam duomenų perdavimui; lusto išvadai yra „atviro kolektoriaus“ tipo, – taip realizuojama aparatūrinė „arba“ funkcija (žr. 36 pav.), dominuojantis yra loginis 0 (jei bent vienas tranzistorius atsidaręs, įtampa linijoje 0 V, nežiūrint likusių tranzistorių būsenos). ***I^2C*** sąsajoje įrenginių bendravimas vyksta pagal valdantysis – valdomasis logiką, tačiau valdančiųjų įrenginių gali būti ne vienas, naudojama įrenginių adresacija. Didžiausias įrenginių skaičius, kuris gali būti sujungtas į magistralę yra ribojamas talpos, kuria šie įrenginiai apkrauna elektrines linijas: didžiausia talpa 400 pF.

I^2C standartas numato tris duomenų perdavimo spartų diapazonus:

- iki 100 kb/s;
- iki 400 kb/s;
- iki 3.4 Mb/s.

I^2C sąsajos signalai: ***SCL*** – perduodami sinchro impulsai, ***SDA*** – perduodami duomenys. Kadangi duomenims yra skirta tik viena ***SDA*** linija, duomenys gali būti perduodami tik viena kryptimi: arba iš valdančiojo valdomajam arba atvirkščiai.

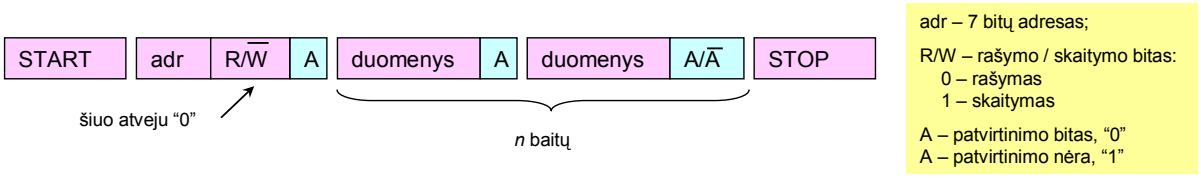


37 pav. I²C sąsajos oscilogramos pavyzdys.

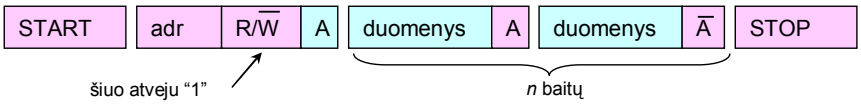
Duomenys yra perduodami tokiu būdu: kai ***SCL*** lygis yra žemas, į ***SDA*** liniją yra išvedama duomenų bito reikšmė, kai ***SCL*** yra aukštas ***SDA*** lygis negali kisti. Kai ***SDA*** lygis kinta ***SCL*** esant aukštam, yra ypatingos situacijos, žyminčios duomenų perdavimo pradžią („start“ požymis) ir pabaigą („stop“ požymis). Duomenys yra perduodami baitais (po 8 bitus). Po „start“ požymio vedantysis įrenginys perduoda 7 ar 10 bitų adresą po kurio seka bitukas, reiškiantis, kokia, skaityto ar rašymo, operacija bus atliekama ir patvirtinimo bitukas, kuriuo patvirtinamas sėkmingas duomenų priėmimas.

Galima išskirti tris pagrindinius duomenų perdavimo scenarijus:

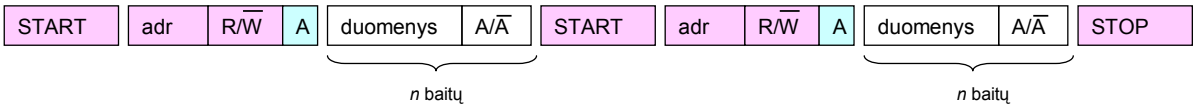
- valdantysis įrenginys perduoda duomenis valdomajam; duomenų perdavimo kryptis nesikeičia;



- valdantysis įrenginys priima duomenis po pirmojo patvirtinimo;



- kombinuotas formatas, duomenys yra ir perduodami ir priimami;

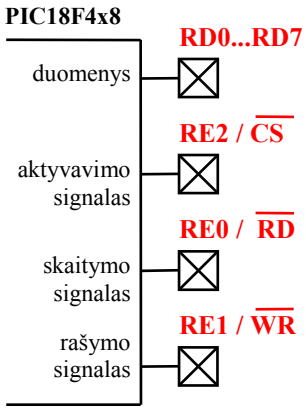


1.16 Lygiagreti sąsaja

Lygiagrečioje sąsajoje, kaip seka ir iš pavadinimo, duomenys yra perduodami lygiagrečiuoju būdu, t.y. kiekvienam duomenų bitui perduoti yra reikalingas bent vienas laidas. Lygiagretus duomenų perdavimo būdas, lyginant su nuosekliu, turi spartos pranašumą: jei lygiagrečioje ir nuoseklioje vienas bitas yra perduodamas per tą patį laiko tarpą, tai tiek kiek lygiagrečiai yra perduodama bitukų, tiek kartų didesnį duomenų kiekį galima perduoti lygiagrečia sąsaja per tą patį laiko tarpą. Šis privalumas kartu yra ir trūkumas: lygiagretus duomenų perdavimas reiškia, kad turi būti naudojamas daugiagyslis kabelis, kuris, buitinės technikos vartotojo požiūriu, yra storas, nelankstus ir nepatogus. PC kompiuteriuose išoriniams įrenginiams prijungti vienintelė lygiagreti buvo ***LPT*** sąsaja, skirta kompiuteriui su spausdintuvu sujungti; vėliau, ***LPT*** sąsają išstūmė patogesnė ***USB*** sąsaja.

Taigi, lygiagrečioji sąsaja yra naudojama arti esantiems įrenginiams sujungti, dažniausiai, šie įrenginiai yra toje pačioje elektroninėje plokštėje įlituoti lustai. Tipinis lygiagrečio sąsajos naudojimo pavyzdys yra duomenų mainai tarp procesoriaus ir ***RAM*** atminties, ***EEPROM*** atminties, įvedimo/išvedimo buferių. Duomenys perduodami po 8, 16, 32 ar net 64 bitus. Be signalų, kurie reiškia duomenis, lygiagrečioje sąsajoje yra naudojami dar ir sinchronizacijos signalai, kurių kiekis ir paskirtis būna įvairūs. Egzistuoja daug ir įvairių lygiagrečiųjų sąsajų atmainų.

Lygiagreti sąsaja *PIC18* mikrovaldikliuose. Elektroninė grandinė, nepriklausomai nuo programos atliekanti duomenų priėmimą bei perdavimą per lygiagrečiąją sąsają, mažai kuo naudinga, nes vis tiek programa turi paruošti duomenis perdavimui arba juos priimti, o sinchronizacijos signalų generavimas programiniu būdu, nesudaro didesnių sunkumų. Žinoma, galima įsivaizduoti tokias situacijas, kai atskira elektroninė grandinė būtų naudinga: (a) elektroninė grandinė gali nepriklausomai nuo programos priimti ar perduoti didesnį duomenų kiekį, tiesiogiai kreipdamasi į ***RAM*** atmintį; ši galimybė mikroprocesorių pasaulyje vadinama ***DMA*** vardu (angl. DMA – Direct Memory Access); (b) mikrovaldikliui reikia labai sparčiai iš anksto nežinomu laiko momentu priimti ar perduoti duomenis. ***PIC18*** šeimos mikrovaldikliuose elektroninė grandinė lygiagrečiajai sąsajai palaikyti yra integruota į senesnius ***PIC18F4x8*** modelius bei naujesnius ***PIC18Fxxxx***, kuriuose integruota ir ***USB*** sąsaja. Pastaruoju atveju, lygiagrečioji sąsaja gali būti panaudota duomenų persiuntimui iš ***USB*** sąsajos į lygiagrečiąją ir atvirkščiai.



38 pav. Lygiagrečioji sąsaja PIC18 mikrovaldikliuose.

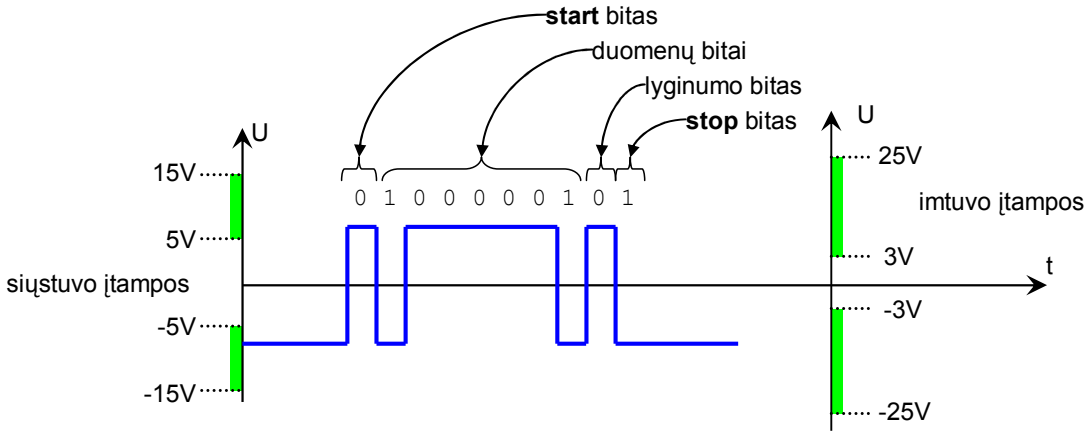
Panagrinėkime ***PIC18F4x8*** mikrovaldikliuose integruotą lygiagrečiąją sąsają. Duomenims perduoti yra naudojamos 8 linijos, tam yra skirti **RD0-RD7** išvadai. Sinchronizavimo signalai: aktyvavimo signalas CS yra sutapatintas su ***RE2*** išvadu, RD – su ***RE0***, WR – su ***RE1***, visų jų aktyvus lygis yra loginis 0. Sinchronizacijos signalų mikrovaldiklis negeneruoja, o tik į juos reaguoja. Kai CS lygis yra žemas, RD krintantis frontas aktyvuoja duomenų įvedimą iš magistralės į mikrovaldiklio **PORTD** registrą. Kai CS lygis yra žemas, WR krintantis frontas aktyvuoja duomenų iš **LATD** registro išvedami į magistralę. Lygiagrečiosios sąsajos grandinė įjungiama su **PSPMODE**=1 bitu, esančiu **TRISE** registre. Tiek skaitymo signalas RD, tiek rašymo WR gali aktyvuoti trūkį, abiems signalams yra naudojamas tas pats trūkio požymio bitas **PSPIF**. Lygiagrečiosios sąsajos elektroninės grandinės naudojimas turi prasmę tik kartu su trūkiais: skaitymo signalas RD automatiškai aktyvuoja trūkį ir duomenų išvedimą, o trūkio paprogramėje yra paruošiamas sekantis baitas išvedimui, įrašant jį į **LATD** registrą; rašymo signalas WR aktyvuoja trūkį ir duomenų įvedimą, o trūkio paprogramėje duomenys yra nuskaitomi iš **PORTD** registro ir nusiunčiami į kitą atminties vietą.

1.17 RS232C sąsaja

RS232C sąsaja buvo sukurta dar 1962 m ir vis dar naudojama iki šiol. Iki atsirandant ***USB*** sąsajai, ***RS232C*** buvo diegiama į visus personalinius kompiuterius, šiuo metu vis dar gaminamos pagrindinės plokštės personaliniams kompiuteriams su integruota ***RS232C*** sąsaja, nors reikėtų pastangų, norint tokią plokštę surasti. Mikrovaldiklių pasaulyje ***RS232C*** yra vis dar plačiai naudojama dėl savo paprastumo tiek elektroninės schemos požiūriu, tiek programos sudėtingumo požiūriu. Dažnai ***RS232C*** naudojama mikrovaldiklio programai derinti.

RS232C sąsaja duomenys yra perduodami asinchroniniu nuosekliu būdu. Šia sąsaja gali būti sujungti tik du įrenginiai, daugelio įrenginių tinklas – neįmanomas. Vienu metu gali vykti duomenų ir perdavimas ir priėmimas, tam naudojamos dvi signalų linijos. Kadangi istoriškai ***RS232C*** visų pirma pradėta naudoti modemams prijungti prie personalinio kompiuterio, ***RS232C*** standarte yra numatyta nemažai papildomų sinchronizavimo linijų, kurios dažniausiai nėra naudojamos mikrovaldiklių taikymo sferoje.

Duomenys ***RS232C*** sąsaja yra perduodami nuosekliai, paketais turinčiais pradžią žymintį „start“ ir pabaigą žymintį „stop“ bitą. Sąsajoje naudojamos dviejų poliariškumų įtampos: loginį „1“ atitinka neigiama įtampa, loginį „0“ – teigiama įtampa (žr. 39 paveikslą). Gali būti perduodami 7 arba 8 duomenų bitai ir papildomai lyginumo bitas (jei reikia), „stop“ bito ilgis gali būti 1T, 1.5T ir 2T, kur T yra vieno bito perdavimo trukmė. Tiek siųstuve, tiek imtuve turi būti vienodai nustatyta: duomenų bitų skaičius, ar perduodamas lyginumo bitas, „stop“ bito ilgis, duomenų perdavimo sparta.



39 pav. RS232C sąsajos signalo oscilograma.

Rimtas ***RS232C*** sąsajos trūkumas – menkas atsparumas elektromagnetiniams trukdžiams. Imtuvo įėjime įtampa yra matuojama bendro laido atžvilgiu, o esant ilgesniems laidas, siųstuvo ir imtuvo bendrojo laido potencialas gali skirtis. To priežastimi gali būti tie patys elektromagnetiniai laukai. Dėl šios priežasties yra numatyta, kad imtuve -3 V ... +3 V įtampų diapazonas turi būti ignoruojamas. Vėliau šiuo požiūriu ***RS232C*** buvo tobulinama ir, panaudojus diferencialinį signalo perdavimo principą, sukurtos atsparesnės trukdžiams ***RS485***, ***RS422***, ***RS423*** ir kitos sąsajos.

RS232C sąsajos standartas numatė iki 20 kb/s duomenų perdavimo spartą ir iki 15 m ilgio arba 2700 pF talpos kabelį. Tačiau praktiškai pavyksta pasiekti žymiai didesnę duomenų perdavimo spartą naudojant žymiai ilgesnius kabelius. Personaliniuose kompiuteriuose integruota ***RS232C*** palaiko iki 115.2 kb/s spartą. Praktiškai su UTP CAT-5 kabeliu (5 pF/m) pasiekiamos duomenų perdavimo spartos ir kabelio ilgiai:

- 19.2 kb/s – 15 m,
- 9.6 kb/s – 150 m,
- 4.8 kb/s – 300 m,
- 2.4 kb/s – 900 m.

Duomenų baito perdavimas per ***RS232C*** sąsaja vyksta tokiu būdu:

- imtuvas ir siųstuvas pradiniu laiko momentu yra nesinchronizuoti tarpusavyje;
- siųstuvas duomenų perdavimą pradeda siųsdamas „start“ bitą, pagal šį bitą įvyksta imtuvo sinchronizacija (nustatomi skaitikliai su kuriais vėliau imtuvas sprendžia kurioje vietoje laiko ašyje yra pirmas, antras ir likuosieji bitai. Atskirai perduodamų sincronizacijos signalų ***RS232C*** sąsajoje nėra);
- vyksta bitų priėmimas, lyginumo bito (jei yra nustatyta) ir „stop“ bito priėmimas.

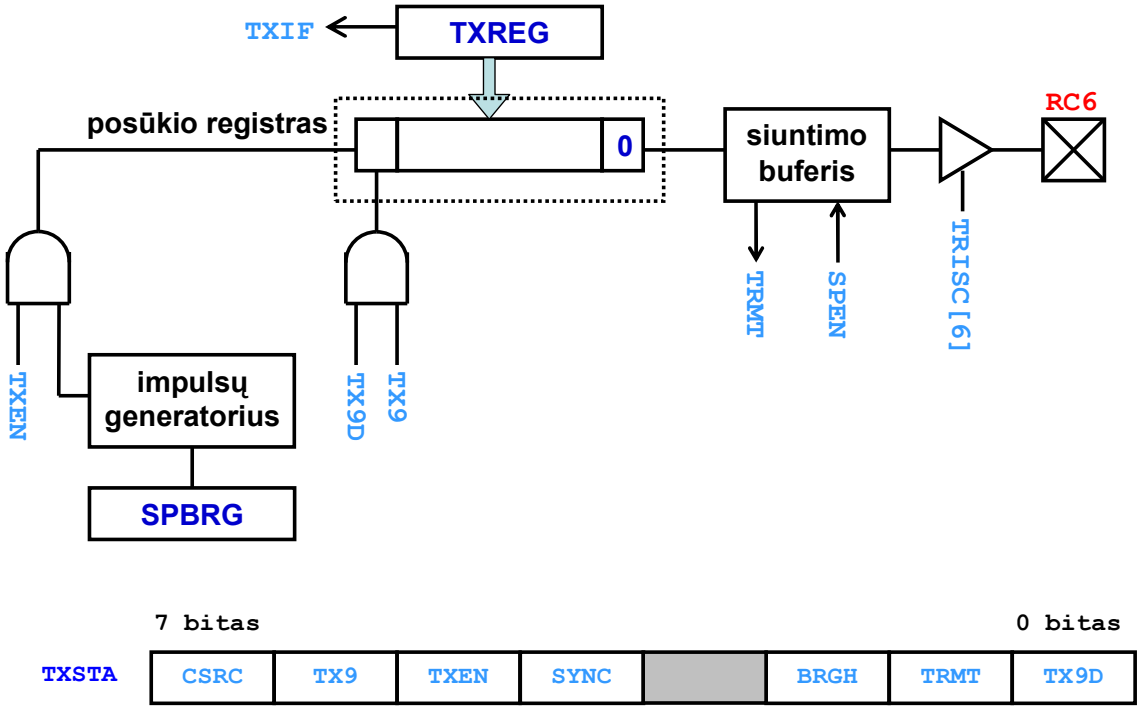
Siųstuvas su imtuvu sinchronizuojasi pagal „start“ bitą ir jų tarpusavio sincronizacija turi išlikti bent 12 bitų ilgio laikotarpiu (1 „start“ + 8 duomenų + 1 lyginumo + 2 „stop“ bitai), o perduodant kitą baitą vėl vyksta sinchronizacija iš naujo. Reikalavimas, kad sichronizacija išliktų 12 bitų perdavimo laikotarpiu yra labai silpnas, praktiškai tai reiškia, kad imtuvo ir siųstuvo duomenų perdavimo spartos gali gana stipriai skirtis, o duomenys vis tiek bus perduoti teisingai.

Nuoseklios asinchroninės sąsajos *PIC18* mikrovaldikliuose. Šioms sąsajoms yra skirtas ***USART*** modulis (angl. Universal Sincronous Asynchronous Receiver Transmitter). Kaip seka iš pavadinimo, šis modulis yra skirtas perduoti duomenis ne tik asincroninėmis sąsajomis, bet ir sinchroninėmis, t.y. naudojant atskirą liniją sichronizavimo signalui. Toliau

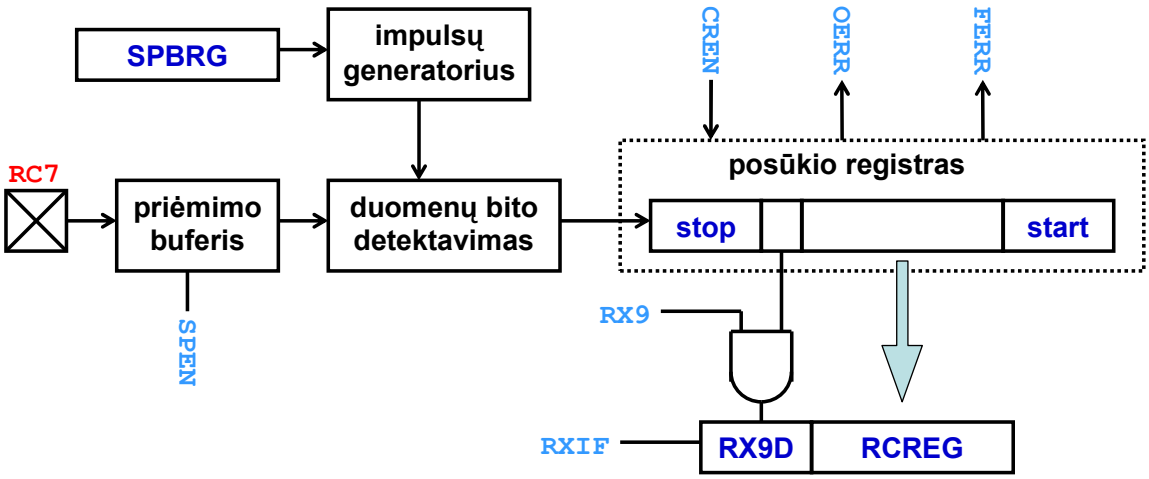
bus paaiškinta kaip naudotį šį modulį tik asinchroninėms sąsajoms, tokioms kaip ***RS232C***, ***RS485***.

USART modulis asinchroninėje veikoje vienu metu gali tiek priimti, tiek išsiųsti duomenis. Duomenų siuntimui yra naudojamas ***RC6*** išvadas (turi būti sukonfigūruotas kaip skaitmeninis išėjimas), priėmimui – ***RC7*** išvadas (turi būti sukonfigūruotas kaip skaitmeninis įėjimas). ***PIC18*** šeimos mikrovaldikliuose ***USART*** modulis formuoja tik 1 „stop“ bitą, kitų galimybių, kurios yra numatytos standarte, nėra.

Supaprastintai ***USART*** modulio duomenų siuntimo principą PIC18Fxxx mikrovaldikliams iliustruoja 40 paveikslas. Siunčiamas duomenų baitas įrašomas į **TXREG** registrą ir automatiškai prasideda jo siuntimas: **TXREG** turinys yra kopijuojamas į postūmio registrą ir bitas po bito kartu su „start“, „stop“ bitais priekyje ir gale yra išsvedami į ***RC6*** išvadą, laiko intervalus formuoja generatorius, valdomas **SPBRG** registru. ***USART*** moduliu gali būti siunčiami baitai po 8 arba 9 bitus, jei siunčiami 9 bitai, tai 9 bitų siuntimo galimybę įjungia bitas **TX9**=1, o devintojo duomenų bito vertė nurodoma **TX9D** bite ir 9-ojo duomenų bito prasmė palikta programuotojo nuožiūrai. Išsiuntus baitą, yra įjungiamas **TXIF**=1 bitas, kuris gali aktyvuoti trūkį. **TXIF** bito nereikia ir negalima ištrinti, jis automatiškai išsitrina, įrašius baitą į **TXREG** registrą, t.y. pradėjus duomenų baito siuntimą. Siuntimo funkcijas kontroliuoja **TXSTA** registras: **TXEN**=1 įjungia siuntimo galimybę; bitas **SYNC** nustato koks vyks duomenų perdavimas, sinchroninis ar asinchroninis (šiame skyriuje visur buvo nagrinėtas ***USART*** modulio veikimas, kai **SYNC**=0, t.y. kai vyksta asinchroninis duomenų perdavimas).



40 pav. Supaprastinta USART modulio duomenų siuntimo loginė schema.



41 pav. Supaprastinta USART modulio duomenų priėmimo loginė schema.

USART modulio duomenų priėmimo principą ***PIC18Fxxx*** mikrovaldikliams iliustruoja 41 paveikslas: nuolat tikrinamas įtampos lygis ant ***RC7*** išvado, detektavus „start“ bitą, fiksuotais laiko tarpais yra matuojamas ***RC7*** išvado signalas ir gauta vertė, loginis 0 arba 1, yra rašomi į postūmio registrą. Priėmus visą baitą, postūmio registro vertė yra kopijuojama į **RCREG** registrą ir įjungiamas **RCIF**=1 bitukas, kuris gali aktyvuoti trūkį. **RCIF** bito nereikia ir negalima ištrinti, jis automatiškai išsitrina, nuskaičius priimtą baitą iš **RCREG** registro. Priėmimo funkcijas valdo **RCSTA** registras: **SPEN** bitas įjungia maitinimą visam ***USART*** muduliui; **CREN** bitas leidžia pasirinkti ar bus priimtas vienas duomenų baitas ir bus laukiama kol jį programa nuskaitys, ar bus priimama tiek duomenų, kiek jų atsiunčiama (jei programa nespės jų nuskaityti iš **RCREG** registro, nauji duomenų bus užrašyti ant viršaus).

Laiko intervalų generatorius tiek ***USART*** priimančiai tiek siunčiančiai dalims yra valdomas **SPBRG** registro. Duomenų perdavimo sparta f yra nusakoma formule, kuri priklauso nuo **BRGH** bito, esančio **TXSTA** registre, vertės:

$$f = \frac{F_{osc}}{64 \cdot (SPBRG + 1)}, \text{ jei } \mathbf{BRGH} = 0, \tag{16}$$

$$f = \frac{F_{osc}}{16 \cdot (SPBRG + 1)}, \text{ jei } \mathbf{BRGH}=1. \tag{17}$$

Duomenų perdavimo spartą, kaip seka iš pateiktų formulių, galima nustatyti diskretiškai (**SPBRG** yra 8 bitų registras, todėl jo vertė formulėse 0..255), ir visai tikėtina, kad nepavyks tiksliai nustatyti norimos spartos, tačiau jau buvo minėta, kad čia idealaus tikslumo ir nereikia, dar visai priimtina jei jūsų nustatyta sparta skirsis nuo norimos iki 5%.

Naujesniuose ***PIC18Fxxxx*** šeimos mikrovaldikliuose ***USART*** modulyje yra įvesti patobulinimai, susiję su tikslesniu duomenų perdavimo spartos nustatymu (tam yra panaudoti du registrai) ir yra papildomos grandinės automatiniam duomenų perdavimo spartos nustatymui, priimtamt duomenis, pagal specialų symbolį. Papildomos galimybes valdo **BOUDCON** registras, kurio nėra ***PIC18Fxxx*** mikrovaldikliuose.

Pavyzdys. ***PIC18Fxxx*** mikrovaldiklio taktinis dažnis 40 MHz, duomenų perdavimo sparta – 9.6 kb/s (viena iš standartinių verčių), siunčiame 8 bitų duomenų paketą, 1 „stop“ bitas. Pasirenkam $f = \frac{F_{osc}}{64 \cdot (SPBRG + 1)}$, **jei BRGH = 0,**
(16) formulę, nes ji duoda didesnį dalinimo koeficientą ir surandam **SPBRG** registro vertę:

$$9.6 = \frac{40000}{64 \cdot (x + 1)}, \tag{18}$$

iš čia x=64. Tai yra supavalinta iki sveiko skaičiaus vertė, realiai mikrovaldiklio duomenų perdavimo sparta bus 9.615 kb/s, t.y. gauname 0.2 % paklaidą.

Programa:

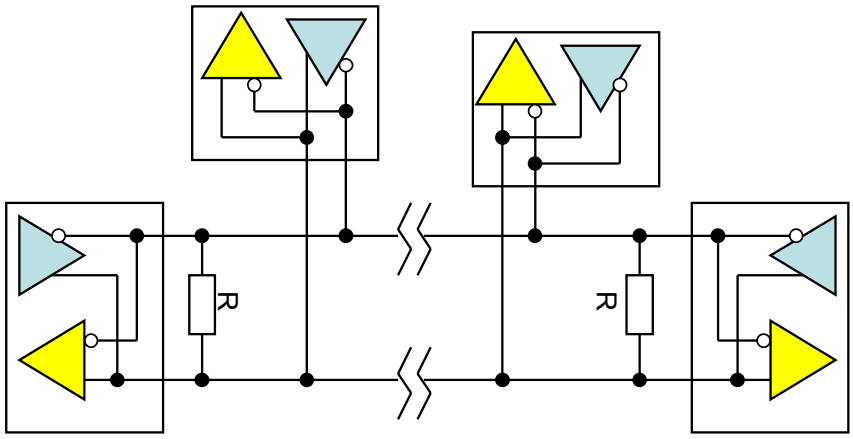
```
void main(void)
{
    char d;
    TRISC &= ~(1<<6);   // nustatom RC6 išvadą kaip skaitmeninį išėjimą
    SPBRG = 64;         // nustatom 9.6 kb/s duomenų perdavimo spartą
    TXSTA = 0b00100000; // įjungiam asinchroninį duomenų perdavimą
    RCSTA = 0b10010000; // įjungiam USART modulį
    ...
    TXREG = 'a';        // išsiunčiam raidės „a“ ASCII kodą
    for(;;)
    {
        If(PIR1bits.RCIF) // ar gautas duomenų baitas?
        {
            d = RCREG;    // priskiriam gautą baitą kintamajam d
            ...           // duomenų apdorojimas
        }
        ...               // likusi programa
    }
}
```

1.18 RS485 sąsaja

RS485 sąsaja, lyginant su **RS232C**, išlaikė duomenų kodavimo paprastumą, tačiau leidžia pasiekti žymiai didesnius kabelio ilgius (iki 1.2 km, kai duomenų perdavimo sparta yra iki 100 kb/s) ir žymiai didesnę duomenų perdavimo spartą (iki 35 Mb/s kai kabelis yra ne ilgesnis nei 12 m). Pagrindinis patobulinimas palietė signalo perdavimo būdą: **RS485** sąsajoje signalas perduodamas kaip dviejų įtampų skirtumas, tam naudojama dviejų laidų pora. Kiekviena iš šių dviejų įtampų nuo bendro lygio gali skirtis nuo -7 V...+12 V (pavyzdžiui, potencialo poslinkį gali sukelti elektromagnetiniai laukai), imtuvas matuoja tik įtampų skirtumą. Didžiausias įrenginių skaičius, kurie gali būti sujungti į vieną tinklą, priklauso nuo imtuvo įėjimo varžos. Daugumos imtuvų įėjimo varža yra 12 kΩ, tuomet didžiausias įrenginių skaičius yra 32.

Yra naudojami du įrenginių jungimo į **RS485** tinklą būdai: dviejų laidų kabeliu (taip, kaip parodyta 42 pav.) ir keturių laidų kabeliu. Antras jungimo atvejis naudojamas kuomet yra centrinis tinko įrenginys (valdiklis), tuomet, viena laidų pora jungia valdiklio siųstuvų išėjimus su kitų tinklo įrenginių imtuvų įėjimais, o kita laidų pora jungia valdiklio imtuvų įėjimus su kitų tinklo įrenginių siųstuvų išėjimais.

Vienu metu duomenis **RS485** tinkle gali siųsti tik vienas įrenginys, kitų įrenginių siųstuvai turi būti išjungti. Taigi, tinkle yra įmanomos kolizijos (vienu metu siunčia keli įrenginiai), o ar jos realiai vyks, priklauso nuo aukštesnio lygio protokolo. Jei tinkle yra vienas valdiklis, o kiti įrenginiai tinkle neturi teisės siųsti duomenis be valdiklio leidimo, tuomet kolizijų nebus. Jei įrenginiai gali pradėti bet kuriuo laiko momentu duomenų siuntimą, tuomet kolizijos yra įmanomos ir aukštesnio lygio protokolas turi į tai atsižvelgti. Jei vyksta kolizijos, tuomet tik iki 37 % viso duomenų srauto yra naudingi duomenys (jei nevyktų kolizijos, 100 % duomenų srauto yra naudingi duomenys). Yra plačiai naudojami Profibus, Modbus aukštesnio lygio protokolai, kurie užtikrina patikimą duomenų perdavimą **RS485** tinkle.



42 pav. Įrenginių jungimas į RS485 tinklą.

RS485* sąsaja *PIC18* mikrovaldikliuose.** ***PIC18 šeimos mikrovaldikliai tiek ***RS485***, tiek ***RS232C*** sąsajoms turi tą patį ***USART*** modulį, kuris jau buvo nagrinėtas 1.17 skyriuje. Dažnai aukštesniame duomenų perdavimo ***RS485*** sąsaja protokole yra naudojami 9 bitai duomenų pakete. Devintasis bitas naudojamas atskirti kuomet perduodamas adresas, o kuomet duomenys. ***PIC18*** šeimos mikrovaldikliai turi galimybę siųsti bei priimti 9 bitų duomenų paketus. 9 bitų siuntimas įjungiama su **TX9**=1 bitu, o perduodamas 9-tasis bitas yra **TX9D**, abu bitai yra **TXSTA** registre. 9 bitų priėmimo galimybė įjungiama su **RX9**=1 bitu, o priimtas 9-tasis bitas **RX9D**, abu bitai yra **RCSTA** registre. Taipogi yra galimybė įjungti adreso detekciją su **ADDEN**=1 bitu, esančiu **RCSTA** registre, tuomet, **USART** modulis priims tik duomenų paketus, kuriuose 9-tasis bitas yra lygus 1. Tokiu būdu, mikrovaldiklio programai nereikia reaguoti į visus duomenų paketus, o tik į tuos, kuriais yra perduodamas adresas. Kai programa atpažįsta savo adresą, adreso detekciją gali išjungti ir priimti visus duomenų paketus.

1.19 CAN sąsaja

Apie 1980 m. Bosh firmoje prasidėjo tyrimai kaip galima būtų pakeisti automobiliuose dideles laidų pynes viena duomenų perdavimo magistrale, prie kurios galėtų jungtis visi automobilyje esantys įrenginiai. Naujoji magistralė pirmą kartą buvo pristatyta automobilių įrangos gamintojų SAE kongrese 1986 m. Magistralė pavadinta „Automotive Serial Controller Area Network“. Spaudoje iškart pasirodė straipsniai apie naują ir perspektyvią sąsaja, o po 2 mėn. Intel pagamino pirmąjį **CAN** lustą. Tačiau, kad Bosh sukurtas naujos magistralės standartas pilnai būtų įgyvendintas elektroniniame luste, prireikė dar 4 metų.

CAN magistralė pasirodė tinkanti ne vien automobilių elektronikai valdyti, jos taikymo sričių įvairovė gerokai pranoko kūrėjų lūkesčius. Visų pirma automobilių elektronika: **CAN** magistralė yra diegiama į Fiat, Renault, Saab, Volkswagen, Volvo gamintojų automobilius; paprastai diegiamos dvi **CAN** magistralės – viena variklio ir kitiems važiavimo įrenginiams valdyti, kita – komforto įrenginiams, diagnostikai ir monitoringui. Nors ir projektuota taikymui automobiliuose, **CAN** magistralė pirmiausia buvo pradėta taikyti pramoniniuose įrenginiuose: Švedijos tekstilės įrangos gamintojai pradėjo naudoti **CAN** tinklą anksčiau už automobilininkus. Šiuo metu **CAN** magistralė naudojimo sritys išsiplėtė: visų pirma, **CAN** magistralė naudojama pramoninės paskirties įrenginių automatizavimui, automobilių pramonėje, medicinos įrenginiuose, aviacijoje ir kitur, kur reikalingas įvairių įrenginių tinklo lankstus valdymas ir itin patikimas duomenų perdavimas.

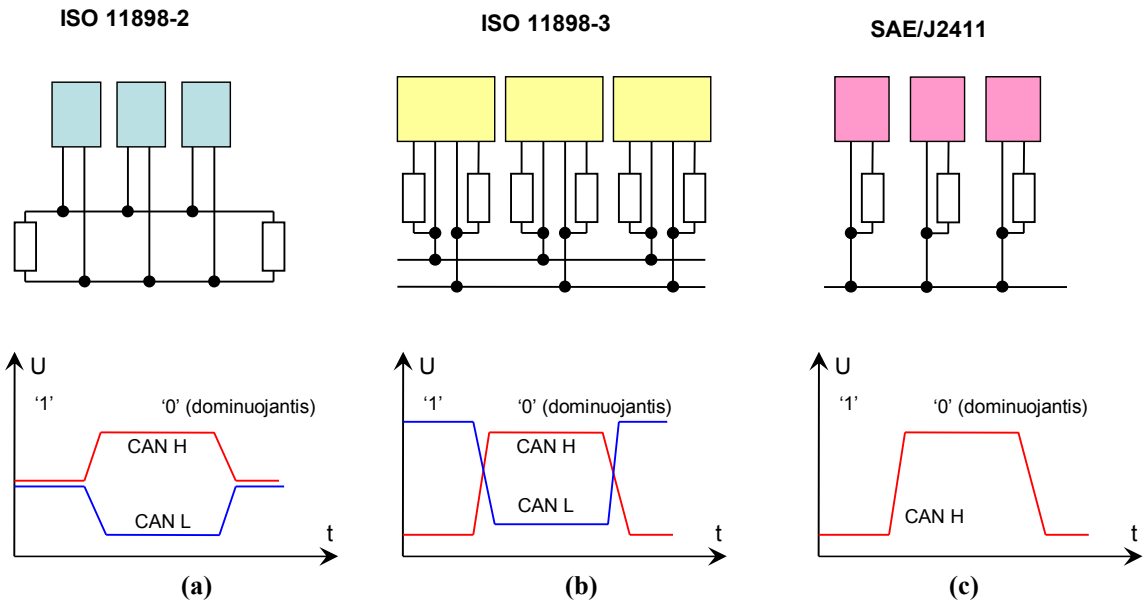
CAN magistralės savybės:

- tinkle esantys įrenginiai turi savo adresus; duomenų paketai pasiekia visus įrenginius, šie paskui sprendžia kaip reaguoti;
- tinkle gali būti daugiau nei vienas vedantysis įrenginys;
- duomenų paketai turi savo prioritetą, t.y. kolizijos (pradeda siųsti du įrenginiai iš karto) atveju siunčiamas paketas su aukštesniu prioritetu, nesugadinant jau pradėto siųsti paketo;
- procesoriaus gedimas viename įrenginyje nesutrikdo kitų tinklo įrenginių duomenų perdavimo;
- labai išvystyti klaidų aptikimo ir klaidingų paketų kartojimo mechanizmai;

Duomenų paketo siuntimas, paketo pristatymo ir klaidų kontrolė yra vykdoma tik apartūrinė **CAN** sąsajos dalis, o programuotojas šių veiksmų tiesiogiai nekontroliuoja. Tai supaprastina mikrovaldiklio programą ir nuima nuo programuotojo pečių dalį atsakomybės už duomenų perdavimo patikimumą. Iš kitos pusės, šias **CAN** sąsajos funkcijas realizuojanti elektroninė grandinė yra gan sudėtinga, o tai pakelia mikrovaldiklio kainą ir neretai pasitaiko, kad **CAN** sąsajos grandinėje lusto projektavimo metu yra įveliamos klaidos. Microchip mikrovaldikliai šiuo požiūriu irgi nėra išimtis. **CAN** standartas apibrėžia du žemiausius lygmenis iš 7, esančių **ISO/OSI** ryšio (sąsajos) apibrėžime: fizinį lygmenį ir duomenų lygmenį. Šių lygmenų funkcijas atlieka **CAN** modulis, t.y. elektroninė dalis, likusios funkcijos pagal **ISO/OSI** ryšio apibrėžimą yra paliktos programuotojo nuožiūrai.

Fizinis CAN lygmuo. **CAN** sąsajos fiziniame lygmenyje turi būti realizuota aparatūrinė „arba“ funkcija. Panašiai aparatūrinė „arba“ funkcija yra jau nagrinėtoje **I^2C** sąsajoje: įrenginiai su atviro kolektoriaus išvadu užtrumpina liniją ir nesvarbu, ar tai daro vienas ar keli įrenginiai,- čia dominuojantis yra loginis „0“, žemas įtampos lygis. **CAN** sąsajoje irgi yra panašiai, tačiau yra ne vienas standartas liečiantis fizinius **CAN** sąsajos įtampų lygius. Pagrindiniai yra trys: ISO 11898-2, ISO 11989-3 ir SAE/J2411. Kadangi egzistuoja ne vienas **CAN** sąsajos fizinio lygmens standartas, tai elektroninėje schemoje

reikės naudoti skirtingus lustus ***CAN*** sąsajos įtampų lygiui suderinti, tačiau mikrovaldiklis ir programa lieka ta pati (duomenų lygmens standartas yra ISO 11898-1).



43 pav. CAN sąsajos įtampos ir prietaisų jungimas į tinklą.
(a), (b) – diferencialinė ryšio linija, (c) – vienalaidė ryšio linija.

Čia ėjo kalba apie fizinio signalo „1“ ir „0“ lygius. Duomenų loginiai „1“ ir „0“ nebūtinai koduojami fizinio signalo „1“ ir „0“ bitais. Pavyzdžiui, ypač paplitusiame Mančesterio kodavimo metode duomenų „1“ koduoja fizinio signalo pokyčiu iš „0“ į „1“, o duomenų „0“ koduoja fizinio signalo pokyčiu iš „1“ į „0“. Kitas pavyzdys: ***USB*** sąsajoje naudojamas ***NRZI*** (angl. Non-Return to Zero, Inverted) kodavimas, kur duomenų „0“ yra koduojamas fizinio signalo pokyčiu, o duomenų „1“ – fizinio signalo pokyčio nebuvimu. Tokių duomenų bitų vertės kodavimu fizinio signalo lygiu yra nemažai. ***CAN*** sąsajoje naudojamas ***NRZ*** (angl. Non-Return to Zero) kodavimas. Įrenginiai ***CAN*** sąsajoje sinchronizuojasi pagal fizinio signalo „1“ → „0“ frontus, ne tik pagal pirmąjį frontą pradedant siųsti paketą, bet vyksta ir pakartotinė sichronizacija pagal visus paketo fizinio signalo „1“ → „0“ frontus. Jei iš eilės eina vienodi fiziniai bitukai, tuomet nėra signalo pokyčio ir įrenginiai gali netekti tarpusavio sinchronizacijos, todėl norint išvengti šios negerovės, ***CAN*** standarte yra numatyta, kad jei iš eilės eina 5 vienodi fiziniai bitai, tai 6-tasis yra įterpiamas papildomai ir priešingos vertės (angl. bit stuffing), imtuvuose šie papildomi bitai yra pašalinami. Taigi, ***CAN*** tinkle įrenginiai turi sugebėti išlaikyti sinchronizaciją laikotarpiu, lygiu 10 bitų perdavimo laikui. Šis ir keletas kitų reikalavimų (baigtinis signalo sklidimo laikas, ***PLL*** dažnio dauginimo grandinės nestabilumui) nustato griežtus reikalavimus taktinių impulsų generatoriui (turima galvoje, kad ***CAN*** modulis yra integruotas mikrovaldiklyje). Mikrochip valdikliams virš 125 kb/s duomenų spartos ***CAN*** tinkle tinka tik kvarciniu rezonatoriumi stabilizuotas taktinių impulsų šaltinis, o iki 125 kb/s tinka ir keraminis rezonatorius, tačiau mikrovaldiklis su RC taktinių impulsų generatoriumi tikrai negalės dirbti ***CAN*** tinkle.

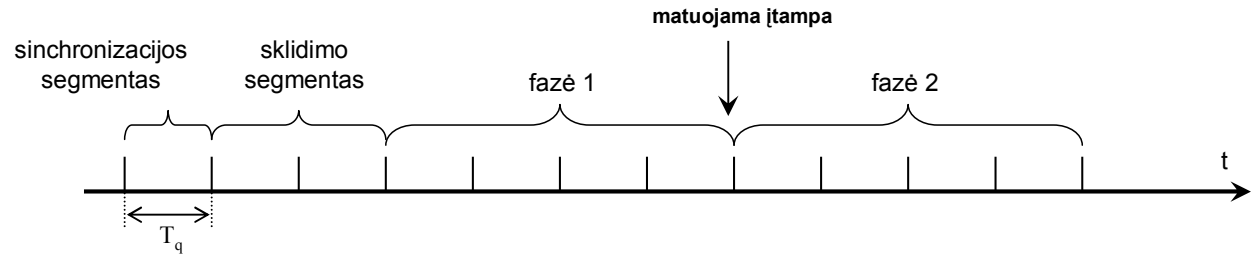
Dažniausiai įrenginiai į tinklą jungiami pagal ISO 11898-2 standartą. Pagal šį standartą galimą pasiekti iki 1 Mb/s duomenų perdavimo spartą, signalui sklisti yra naudojamas susuktos poros kabelis (bendras laidas nėra reikalingas), kabelio galuose jungiamos suderintos apkrovos. Didžiausias kabelio ilgis yra 40 m kai duomenų perdavimo sparta – 1 Mb/s ir 1 km kai duomenų perdavimo sparta yra 40 kb/s.

Įrenginiai į ***CAN*** tinklą jungiami pagal ISO 11898-3 standartą, kai reikia didesnio duomenų perdavimo patikimumo ir didelė perdavimo sparta nėra reikalinga. Signalas sklinda dviejų gyslų kabeliu (yra reikalingas bendras laidas, nepavaizduotas 43 paveiksle), laikoma, kad kabeliai pakankamai trumpi, jog būtų galima nesiskaityti su signalo atspindžiais. Vienam iš laidų nutrūkus ar esant užtrumpintam, duomenų perdavimas išlieka įmanomas. Duomenų perdavimo sparta galima iki 125 kb/s.

Pagal SAE/J2411 standartą reikalingas tik vienas laidas ir, žinoma, bendras. Dažniausiai naudojamas automobiliuose, nes reikia tik vieno laido, o bendro laido vaidmenį atlieka mašinos korpusas. Duomenų perdavimo sparta yra iki 41.6 kb/s.

Siuntėjas išstatydamas bitą į ***CAN*** liniją kartu tikrina ar įtampų lygiai yra tokie, kokius jis išstatė, nes jei siuntėjas išstatė fizinį „1“ bitą, kiti įrenginiai jį gali pakeisti į „0“. Pagal tai yra atpažįstamos kolizijos tinkle, t.y. atvejis, kai pradeda siųsti keli įrenginiai. Kolizija yra atpažįstama pagal 12 pradinių bitų paketo pradžioje. Įrenginys, kuris detektavo, kad jo bito vertę kažkas pakeitė, nutraukia siuntimą ir laukia kol tinklas atsilaisvins. Likęs įrenginys tiesiog tęsia paketo siuntimą. Yra keletas bitų, kuriuos perduodamas siuntėjas išstato fizinį „1“ ir laukia ar kas nors jį pakeis į „0“: ***ACK*** bitas, kuriuo priėmėjai informuoja siuntėją, kad bent vienas įrenginys tinkle priėmė paketą, ir ***CRC Del*** bitas, kuriuo priėmėjai informuoja siuntėją, kad aptiko pakete klaidą. Jei nė vienas įrenginys tinkle negavo paketo arba bent vienas aptiklo klaidą, siuntėjas automatiškai pakartotinai siučia tą patį paketą. Taigi, kiekvieno bito siuntimo metu vyksta dvipusis ryšys.

1 bitas *CAN* sąsajoje. Kadangi siuntėjas siųsdamas 1 bitą turi kontroliuoti ar dar kas nors tinkle nekeičia jo duomenų (kad tai būtų įmanoma padaryti, turi būti atsižvelgta į signalo vėlavimą), ***CAN*** modulyje vieno bito siuntimas yra sudarytas iš daug smulkesnių fazių. Šios fazės yra: sinchronizacijos, sklidimo, 1-oji ir 2-oji fazės. Kiekviena fazė yra sudaryta iš tam tikro laiko kvantų, kurių trukmė T_q (žr. 44 paveikslą), skaičiaus. Sinchronizacijos fazės trukmė visada yra $1T_q$, visų signalų frontų trukmė ***CAN*** sąsajoje turi būti mažesnė už $1T_q$. Sklidimo trukmei kompensuoti yra numatyta sklidimo fazė, kurios reikšmė $1...8T_q$. 1-os ir 2-os fazių sandūroje siuntėjas nuskaito fizinio bito linijoje vertę. 1-os fazės laikas gali būti trumpinamas tam tikrą laiko kvantų T_q skaičių, o 2-os fazės laikas gali būti ilginamas pagal tam tikras sichronizavimo taisykles.

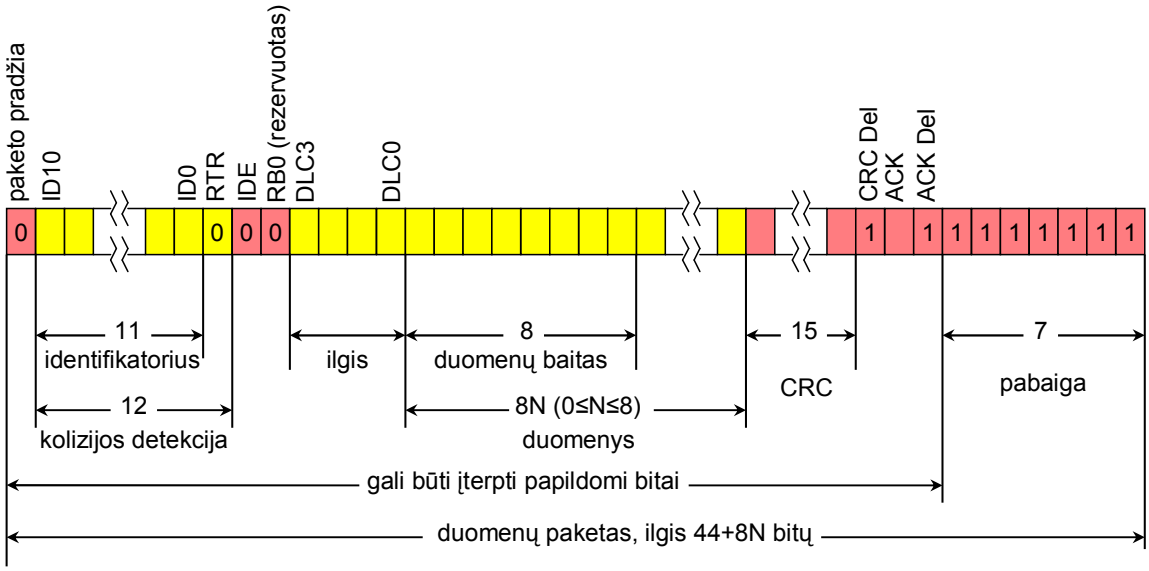


44 pav. Vieno bito siuntimo fazės.

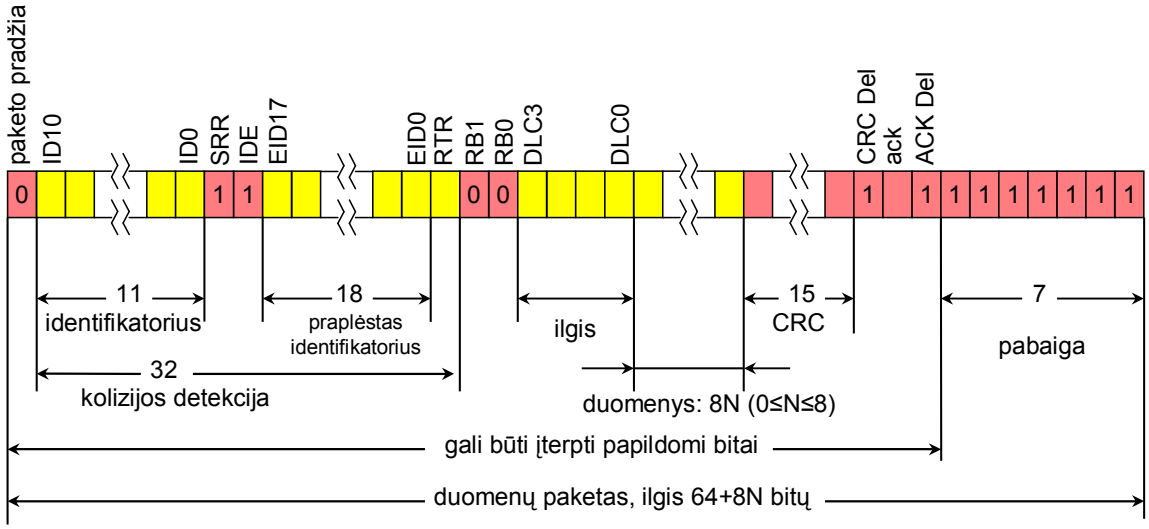
Duomenų perdavimo sparta f yra $1/T_{bito}$, kur T_{bito} yra vieno bito perdavimo trukmė. Vieno bito T_{bito} trukmė yra visų fazių trukmių suma: $T_{bito}= N \cdot T_q$. N gali kisti nuo 5 iki 25.

Duomenų lygmuo. *CAN* paketų tipai:

- standartinis paketas, 11 bitų identifikatorius:
  - duomenų,
  - ***RTR***,
- išplėstas paketas, 29 bitų identifikatorius:
  - duomenų,
  - ***RTR***,
- klaidos paketas,
- vėlinimo paketas.



45 pav. 11 bitų (standartinio) CAN duomenų paketo struktūra.



46 pav. 29 bitų (išplėsto) CAN duomenų paketo struktūra.

Programuotojas betarpiškai dirba su pirmaisiais dviem (duomenų) paketų tipais, o klaidos ir vėlinimo paketų programuotojas tiesiogiai nekontroliuoja. Duomenų paketai yra

sudaryti iš informacijos, kurią kontroliuoja programuotojas: identifikatoriaus, ***RTR*** (angl. Remote Transmit Request) bito, duomenų ilgio ir pačių duomenų; informacija, kurios nekontroliuoja programuotojas: tarnybiniai bitai, kontrolinė suma. Identifikatorius ***CAN*** paketuose gali būti 11 arba 29 bitų ilgio. Įprasta, kad identifkatorius yra naudojamas ***CAN*** tinklo įrenginių adresams, (bendrai paėmus, jo paskirtis priklauso nuo programuotojo), nes ***CAN*** modulis turi filtrus, kurie gali filtruoti paketus pagal identifikatorių neapkraudami programos. ***CAN*** pakete gali būti 0 .. 8 duomenų baitai. ***RTR*** bitu siuntėjas informuoja adresatą, kad jis laukia duomenų. Jei yra įjungtas ***RTR*** bitas, pakete duomenų nebūna. Šio bito prasmė: ***CAN*** modulis automatiškai, be programos įsikišimo gali išsiųsti atsakymą (duomenų paketą), jei gauna paketą su ***RTR*** bitu ir tam tikru identifikatoriumi. Žinoma, programa turi įjungti šią galimybę ir iš anksto turi paruošti atsakymą į tokius paketus.

CAN* sąsaja *PIC18* mikrovaldikliuose.** ***CAN sąsaja yra integruota senesniuose ***PIC18Fxx8*** ir naujesniuose ***PIC18Fxx8x*** modeliuose. Naujesniuose modeliuose ***CAN*** modulis turi žymiai daugiau galimybių, tačiau čia apsiribokime senesniais modeliais. Taigi, ***CAN*** modulis turi 3 siuntimo buferius, 2 priėmimo buferius, penkis filtrus (du priklauso vienam priėmimo buferiui, trys – kitam). Visą šį sudėtingą mechanizmą valdo daugybė registrų.

CAN modulis turi būsenas: išjungtas, normalaus darbo, tik klausymosi, konfigūravimo ir derinimo (paketas iš siuntimo buferių perduodamas tiesiai į priėmimo buferius). Išjungta būsena reiškia, kad ***CAN*** modulis yra neaktyvus, jam išjungtas maitinimas. Darbinėje būsenoje ***CAN*** modulis atlieka visas numatytas operacijas. Tik klausymosi būsenoje ***CAN*** modulis paketo priėmimo metu neišstato ***CAN*** linijoje ***ACK*** ir kitų tarnybinių bitų, t.y. ***CAN*** modulis neinformuoja siuntėjo, kad sėkmingai priėmė paketą ar kad įvyko klaida. ***CAN*** modulį valdantys registrai gali būti keičiami tik konfigūravimo būsenoje. Derinimo būsenoje paketas iš siuntimo buferių perduodamas tiesiai į priėmimo buferius, ši būsena naudinga tik programos derinimo metu. Būsenos nustatomos **CANCON** registre, o **CANSTAT** informuoja apie dabartinę būseną. Norint nustatyti ***CAN*** modulio būseną, reikia modifikuoti **CANCON** registrą, tačiau būsena iš karto nepasikeičia, praeis tam tikras laiko tarpas, kol pasikeis ***CAN*** modulio būsena,- tą momentą galimą kontroliuoti su **CANSTAT** registru.

CAN duomenų perdavimo sparta. Laiko intervalą T_q ir vieno bito perdavimo fazes kontroliuoja **BRGCON1**, **BRGCON2** ir **BRGCON3** registrai. Bitų grupė **BRP0-BRP5** nustato intervalą T_q:

$$T_q[\mu s] = \frac{2 \cdot (BRP + 1)}{F_{osc}[MHz]}, \qquad (19)$$

čia F_{osc} yra mikrovaldiklio taktinis dažnis, BRP – **BRP0-BRP5** bitais išreikštas skaičius 0...63. Bitai **SJW0**, **SJW1** nustato kiek 1-ą ir 2-ą fazes galima ilginti ar trumpinti sichronizavimosi metu (T_q laiko intervalais vienetais). Jei mikrovaldiklio taktinių impulsų generatorius yra stabilizuotas kvarciniu rezonatoriumi, tai užtenka $1T_q$ pataisos, t.y. **SJW0**=0, **SJW1**=0. Bitai **SEG1PH0** ... **SEG1PH2**, **SEG2PH0** ... **SEG2PH2** ir **PRSEG0** .. **PRSEG2** nustato 1-os, 2-os ir sklidimo fazių trukmes, atitinkamai.

CAN paketo siuntimas. ***PIC18Fxx8*** šeimos mikrovaldikliai turi 3 siuntimui skirtus buferius. Jų paskirtis ir valdymas yra visiškai identiški. Jei yra įjungiamas siuntimas iš kelių buferių vienu metu, ***CAN*** paketų siuntimo eiliškumas yra toks kaip ir ***CAN*** paketų buferiuose prioritetai. Kelių siuntimo buferių naudingumas pasireiškia kai yra naudojama operacinė sistema, yra keli procesai, kurie siunčia ***CAN*** paketus, tuomet yra patogu kiekvienam procesui ***CAN*** paketo siuntimui panaudoti kitą buferį. Norint išsiųsti ***CAN*** paketą iš 0 buferio, reikia:

- ***CAN*** paketo identifikatorių įrašyti į **TXBOEIDH**, **TXBOEIDL**, **TXBOSIDH**, **TXBOSIDL** registrus;
- jei siunčiamas paketas su 29 identifikatoriumi, įjungti **EXIDE** bitą **TXBOSIDL** registre;
- jei siunčiamas RTR paketas, įjungti **TXRTR** bitą **TXB0DLC** registre;
- nurodyti duomenų baitų skaičių **TXBODLC** registre;
- įrašyti siunčiamus duomenis į **TXB0DM0** – **TXB0DM7** registrus;
- aktyvuoti siuntimą įjungiant **TXREQ** bitą **TXB0CON** registre.

Aktyvavus siuntimą su **TXREQ** bitu, mikrovaldiklio ***CAN*** modulis tolimesnius siuntimo veiksmus vykdo automatiškai, jei reikia, paketo siuntimas yra kartojamas ir t.t. Programuotojas gali kontroliuoti siuntimo klaidų skaičių, esantį **TXERRCNT** registre, prireikus gali nutraukti siuntimą su **ABAT** bitu, esančiu **CANCON** registre (yra nutraukiamas ***CAN*** paketų siuntimas visuose buferiuose) .

CAN paketų priėmimas. ***PIC18Fxx8*** šeimos mikrovaldikliai turi 2 priėmimui skirtus buferius. Keli priėmimo buferiai suteikia labai naudingo funkcionalumo. Visų pirma, paketai gali būti filtruojami pagal paketo identifikatorių ir vienokios filtro vertės gali būti nustatytos vienam priėmimo buferiui, o kitas – kitokios. Tokiu būdu, ben programos įsikišimo vyksta ***CAN*** paketų filtravimas ir rūšiavimas. Antra, programa gali nespėti paimti ***CAN*** paketo iš priėmimo buferio, o jau yra priimas kitas paketas. Naudinga, kad esant užimtam vienam buferiui, paketas bus priimtas į kitą buferį. Toliau nenagrinėsime kaip nustatyti filtrus, įjungus maitinimą mikrovaldikliui, ***CAN*** paketų filtrai yra nusatyti priimti visus ***CAN*** paketus. Norint sukonfigūruoti priėmimo 0 buferį, reikia atlikti tokius veiksmus:

- nustatyti kokius paketus nurėsite priimti: tik standartinius (11 bitų identifikatorius), tik išplėstinius (29 bitų identifikatorius) ar visus. Tai kontroliuoja **RXM0**, **RXM1** bitai, esantys **RXB0CON** registre;
- jei norite, kad 0 buferiui esant pilnam, ***CAN*** paketą priimtų 1-as buferis, reikia įjungti **RXB0DBEN** bitą **RXB0CON** registre;

Jei buferis priėmė paketą, įjungiamas **RXFUL** bitas **RXB0CON** registre, kurį reikia išjungti po to kai yra nuskaitomas visas ***CAN*** paketas iš šių registrų:

- priimto ***CAN*** paketo identifikatorius yra **RXB0EIDH**, **RXB0EIDL**, **RXB0SIDH**, **RXB0SIDL** registruose;
- jei **EXID** bitas **RXB0SIDL** registre yra įjungtas, tuomet yra priimtas išplėstinis paketas, priešingu atveju – standartinis;
- įjungtas **RXRTR** bitas **RXB0DLC** registre reiškia, kad yra priimtas **RTR** paketas;
- priimtų duomenų baitų skaičius yra **RXB0DLC** registre;
- priimti duomenys yra **RXB0D0** – **RXB0D7** registruose;

Priėmimo metu įvykusių klaidų skaičių galima sužinoti **RXERRCNT** registre.

Literatūra

1. **M.A.Mazidi, R.D.McKinlay, D. Causey,** *PIC Microcontroller and Embedded Systems,* (Pearson, New Jersey, 2008).
2. *PIC18Fxx8 Data Sheet*, (Microchip, 2003).
3. *PIC18F2455/2550/4455/4550 Data Sheet*, (Microchip, 2006).
4. **B.C.Baker**, *Analog and Interface Guide*, (Microchip, 2005).
5. *Understanding ESR in Capacitors*, Sencore.
6. **R.Schmitt**, *Electromagnetics Explained,* (Elsevier, Woburn, 2002).
7. **V.Soffel**, *Asyncronous Interfaces Overview: UART and LIN bus*, (Microcontroller Proc Corporation, 2003).
8. The *I^2C – Bus Specification. Version 2.1*, (Philips, 2001).
9. Basics of the RS-485 Standart, www.bb-elec.com
10. **D.Caban**, *Software implementation of the I^2C protocol*, (Circuit Cellar Online, 2002).
11. **D.Kalinsky, R.Kalinsky**, *Introduction to Serial Peripheral Interface*, (Embedded System Design, 2002).
12. **K.Pazul**, *Controller Area Network (CAN) basics*, (Microchip AN713, 1999).
13. **P.Richards**, *CAN Physical Layer Discussion*, (Microchip AN228, 2002).
14. *CAN Specification. Version 2.0*, (Bosch, Stuttgart, 1991).
15. **D.Anderson**, *USB System Architecture (USB2.0)*, (Addison-Wesley, Massachusetts, 2001).