Ekološko snovanje elektronskih naprav

ENOTA 8: Mikrokrmilniški sistemi 1.del

Ime avtorja: **Andrej Sarjaš**

8.1. Uvod v mikrokrmilniške sisteme 2
8.2. Struktura mikrokrmilnika 4
8.2.1. Pogosti termini 6
8.3. Struktura mikrokrmilnika 7
8.3.1. Jedro procesorja 7
8.3.2. Pomnilnik 9
8.4. Periferne enote 12
8.4.1. Digitalni vhodi in izhod 12
8.4.2. Analogni vhodi in izhodi 14
8.4.3. Števci in časovniki 18
8.4.5. Komunikacijski vmesniki 20
8.3. Napotki za nizko energetsko rabo krmilnika ter ekološki vidiki 24

Vsebina poglavja:

- Sistemi realnega časa
- Komponente realnega časa
- Snovanje programa v sistema realnega časa

Funded by the Erasmus+ Programme of the European Union

Ecoinnovation Skills for European Designers, Številka projekta: 562573-EPP-1-2015-1-SI-EPPKA2-SSA. Opomba: Vsebina publikacije je izključno odgovornost avtorja in v nobenem primeru ne predstavlja stališč Evropske komisije. Komisija ni odgovorna za kakršno koli uporabo gradiva, ki bi lahko bila storjena na podlagi vsebine, ki jih gradivo vsebuje.

8.1. Uvod v mikrokrmilniške sisteme

Od časa, ko je Intel predstavil prvi mikroprocesor 4004 se je začelo obdobje razvoja mikrokrmilnikov. Moderna struktura krmilnika TMS1802 od podjetja Texas Instruments, ki je bil razvit za uporabo v kalkulatorjih, je do leta 1971 bil uporabljen na mnogih sistemih realnega časa, kot so: ure, merilne naprave, blagajne itd.. Z razvojem nove serije TMS100, katera se je začela v letu 1974 je mikrokrmilnik vseboval periferne enote, ki so osnovne komponente tudi v današnjih modernih mikrokrmilnikih (RAM,ROM, I/O). TMS100 je bil takrat poznan pod vzdevkom mikroračunalnik. Prvi krmilnik, ki je doživel pravi uspeh in široko uporabo je bil Intelov 8048, kateri je imel integrirano tipkovnico. Tedanje obdobje se je nadaljevalo z modeli Intel 8051, kakor tudi Motorolo 68HCxx.

Današnjo proizvodnjo mikrokrmilnikov lahko štejemo v bilijonih kosov na leto, kjer imamo veliko množico različnih proizvajalec. Današnje elektronske sisteme si težko predstavljamo brez mikrokrmilnika in njihova uporaba nenehno narašča. Naštejmo nekaj skupin naprav, katere uporabljajo mikrokrmilniške sisteme,

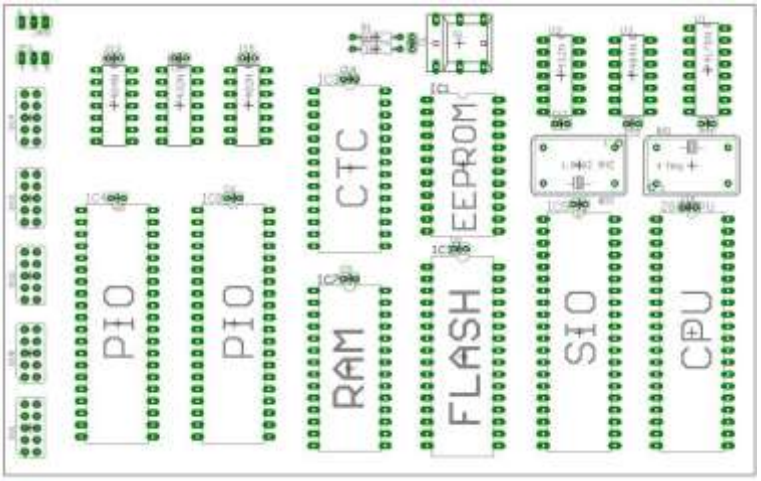
- Gospodinjski aparati (mikrovalovne pečice, pralni stroji, kavin avtomat itd..).
- Telekomunikacije (telefoni, pametni telefoni, modemi, usmerjevalniki itd..).
- Zabavna elektronika (televizije, predvajalniki glasbe in videa, igralne konzole itd..)
- Industrija (vodenje in nadzor naprav in proizvodnih procesov).
- Avtomobilska industrija (nadzor motorja, varnostni sistemi vožnje itd..).
- Vesoljska tehnologija.

Mikrokrmilniški sistemi so doživeli velik razcvet in so odličen nadomestek analognim sistemov in vezjem. Z vključevanjem mikrokrmilniškega sistema v dizajn naprave, lahko bistveno zmanjšamo velikost naprave, povečamo učinkovitost, zanesljivost ter lažjo nadgradnjo. Iz ekološkega vidika je naprava na osnovi mikrokrmilniškega sistema bistveno varčnejša, poraba material je znatno nižja in reciklaža je preprostejša. Moderni mikrokrmilniki ob osnovnih perifernih enotah (RAM, FLASH, I/O, komunikacijski moduli) vsebujejo ločene enote, s katerimi lahko omogočimo znižano porabo energije v času, ko sistema ne potrebujemo ali je neaktiven (Standby mode, Weakup mode itd..). Z dobrim poznavanjem arhitekture mikrokrmilnika ter koncepta izvajanja programa na sistemu, lahko razvijemo učinkovit sistem ali napravo, ki je ekološko skrbnejša. Ekološko snovanje pri mikrokrmilniški sistemih ima tako pomembno vlogo, saj obstaja veliko možnosti in načinov, kako zagotoviti zanesljivo delovanje, nizko porabo ter nizko porabo materiala.

Torej če se vrnemo nazaj, kaj je mikrokrmilnik? Kakšna je razlika med mikrokrmilnikom in procesorjem? Zakaj v bistvu potrebujemo mikrokrmilnik in kakšna je njegova naloga? Za primer vzemimo enostavno napravo za gretje vode z nastavljivo temperaturo. Princip delovanje naprave je sledeč:

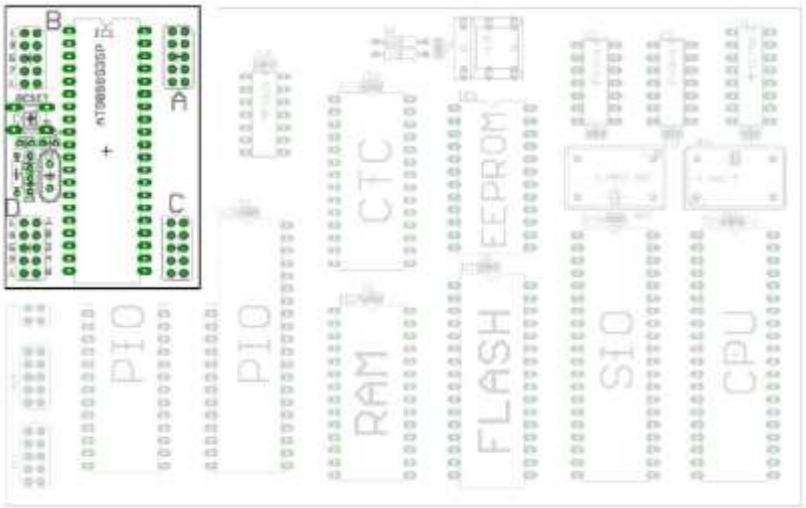
- Periodična meritev temperature.
- Vodenje grelca, glede na trenutno in želeno temperaturo (ON/OFF).
- Vmesniki za upravljanje naprave (tipke, tipkovnica).
- Prikaz temperature na zaslonu.

Pričnemo z dizajniranjem tiskanega vezja-PCB (PCB-printed circuit board) in uporabimo processor Zilog's Z80. Ob procesorju še dodamo dve vhodno izhodni enoti – PIO, serijski vmesnik SIO, časovnik, SRAM, FLASH, EEPROM. Rezultirajoče tiskano vezje je predstavljeno na sliki 1.



Slika 1. Tiskano vezje s procesorjem Z80.

Enako problem lahko rešimo z mikrokrmilnikom ATmega16. Dizajn tiskanine za enka sistem je podan na sliki 2.



Slika 2. Tiskano vezje s krmilnikom ATmega16.

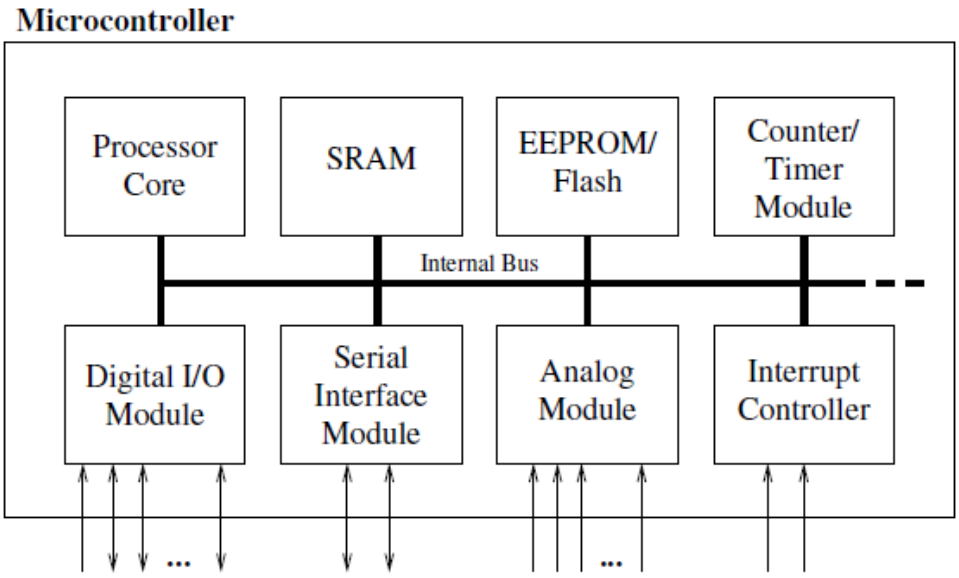
Če pogledamo obe tiskanini je razvidno, da je tiskanina na sliki 2. manjša od tiskanine na sliki 1. Na kratko lahko strnemo mikrokrmilnik je mikroprocesor s perifernimi enotami v enem čipu. Primer med Z80 in ATmega16 smo izbrali iz razloga, ker mikrokrmilnik ATmega16 je razvit na osnovi Z80. Torej vsebuje Z80 in ima integrirane

periferne enote, kot so; serijski in I/O vmesnik, ADC, SRAM, ROM, FLASH in EPPROM. V tem primeru je tudi razvidna poglavitna uporabnost in prednost mikrokrmilnika.

Pri razvoju naprave imamo veliko možnosti, ko se odločimo za razvoj naprave na osnovi mikrokrmilnika moramo izbrati proizvajalca in družino mikrokrmilnika. Družina mikrokrmilnika je ponavadi povezna z učinkovitostjo aritmetično logične enote (8,16,32,64,128bitne). Za dano aplikacijo je potrebno določiti tudi velikost čipa. Mikrokrmilniki se v isti družina ločijo po velikosti čipa ozirom prostih pinov. Tako lahko imamo 48, 100, 144, 176 pinska ohišja iste družine. Če ne potrebujemo veliko zunanjih vmesnikov je smiselno izbrati čip z manj pini, tako privarčujemo na ceni naprave ter velikosti tiskanine.

8.2. Struktura mikrokrmilnika

Trenuten razvoj mikrokrmilniški sistemov je strm. Trenutno so najpogosteje v uporabi 16-32 biti sistemi s hitrostjo ure od 10MHz-300MHz. Notranja zgradba krmilnikov je v osnovi enaka. Slika 3. prikazuje osnovno zgradbo mikrokrmilnika.



Slika 3. Zgradba mikrokrmilnika.

Vse periferne enote mikrokrmilnika slika 3. so med seboj povezna z osrednjim vodilom ('Internal Bus'). Naštejmo tipične module, ki so skupin večini mikrokrmilnikom;

- **Procesor CPU:** Procesor ali CPU ('Central processing unit') je aritmetično logična enota, ki je sposobna matematičnih operacij (seštevanje

odštevanje) ter vodi notranje registre (pomnilniški register, programski register, akumolator itd.).

- **Pomnilnik:** Pomnilnik služi za shranjevanje končnih ali vmesnih podatkov ter spremenljivk. Pomnilnik mikrokrmilnika je razdeljen na programski in podatkovni del. Pri zmogljivejših krmilnikih poznamo tudi DMA ('Direct Memory Acess'), ki skrbi, da periferen enote direktno komunicirajo s pomnilnikom in pri tem ne prekinejo delo procesorja.
- **Prekinitveni kontroler:** Prekinitve so ključne za učinkovito izvajanja programa v realnem času. Prekinitve prekinjajo osnovno izvajanje programa v primeru dogodka na perifernih enotah. Prav tako so uporabne v primeru, ko želimo znižati porabo energije ('Sleep mode').
- **Števci in časovniki:** Večina krmilnikov ima vsaj dva števca, ki sta namenjena merjenju časa, štetju zunanjih dogodkov ter intervalov. Prav tako imajo mnogi kontrolorji časovnika enkoderski način štetja ter PWM - modulacijo ('Pulse Width Modulation'). PWM modulacija je pogosto uporabljena, kot DAC pretvorba (DAC- 'Digital to Analog Conversion'). Resolucija časovnika/števca je odvisna od sistemske ure, ki je določena z zunanjim ali notranjim oscilatorjem ter dolžino številnega registra. Dolžina številnega registra je določen z bitnostjo registra. Tako lahko imamo 8,16 ali 32 bitne števce.
- **Digitalni vhodi/izhodi:** Pogosto jih srečamo pod oznako GPIO ('General purpose Input and Output'). Število vhodov in izhodov je omejeno s tipom krmilnika ter njegovo izvedbo.
- **Analogni vhod/izhodi:** Glede na manjše krmilnike, večina krmilnikov ima vgrajena analogne pretvornike ADC (ADC – 'Analog to Digital Conversion'). Zmogljivejše serije krmilnikov imajo več ločenih ADC-cejev, lahko tudi do štiri ločene enote. Te enote so preko multiplekserja povezane na zunanje pine. V praksi pomeni, da lahko uporabimo več multipleksirnaih analognih vhodov. Število multipleksiranih fizičnih analognih vhodov na pinih krmilnika je tako mnogo višje, kot število ADC enot v krmilniku. ADC enote se ločijo glede na resolucijo pretvorba, tako ločimo od 6 do 12 bitne pretvornike. Resolucijo ADC-ja določa kontrolni register ADC enote, ki se nastavlja s pomočjo programske kode. V novejših serijah krmilnikov je trend ta, da se integrirajo tudi DAC enote, ki so ključne za procesiranje izhodnih signalov.
- **Komunikacijski vmesniki:** Za periferne naprave na tiskanini, ki niso del čipa se najpogosteje uporabljajo tri vrste komunikacijskih standardov. To so serijska povezava USART, I2C in SPI. Za vse povezave je značilno, da je lahko razdalja med napravama dolga le nekaj centimetrov do največ metra.
- '**Watchdog' časovnik:** Watchdog časovnik je poseben števec, ki se proži neodvisno glede na delovanje krmilnika. Ta časovnik skrbi, da se v primeru napake delovanja krmilnik ponovno resetira.
- **Enota za Razhroščevanje:** Pri snovanju programa za krmilnik je praktično, da uporabljamo vmesnik za razhroščevanje. Ta vmesnik lahko dostopa do notranjih registrov krmilnika v času izvajanja programa. Na ta način je lažje iskati napake v programski kodi ter odpravljati morebitne težave pri

izvajanju programa na krmilniku. Pogosto ga srečamo pod terminom JTAG (JTAG- 'Joint Test Action Group').

Če na kratko strnemo. Mikrokrmilnik je okrnjen procesor, ki ima integrirane prej naštete periferne enote. Največja prednost mikrokrmilnika je cena. Z njim lahko prihranimo denar, zmanjšamo velikost naprave ter sistemu dodamo določeno računsko moč. Slabost mikrokrmilnika je ta, da ne more dosegati hitrosti analognih vezij. Pri sistemih z zelo kratkim reakcijskim časom, moramo določen del sistema nadomestit z analognim vezjem. V večini primerov reakcijski čas ni ključnega pomena pri delovanju naprave. Novi mikrokrmilniki že dosegaj reakcijski čas v rangu nekaj 10 mikro sekund.

8.2.1. Pogosti termini

Pogosto se srečamo z različnimi termini, ki so zavajajoči ali so napačno v uporabi.

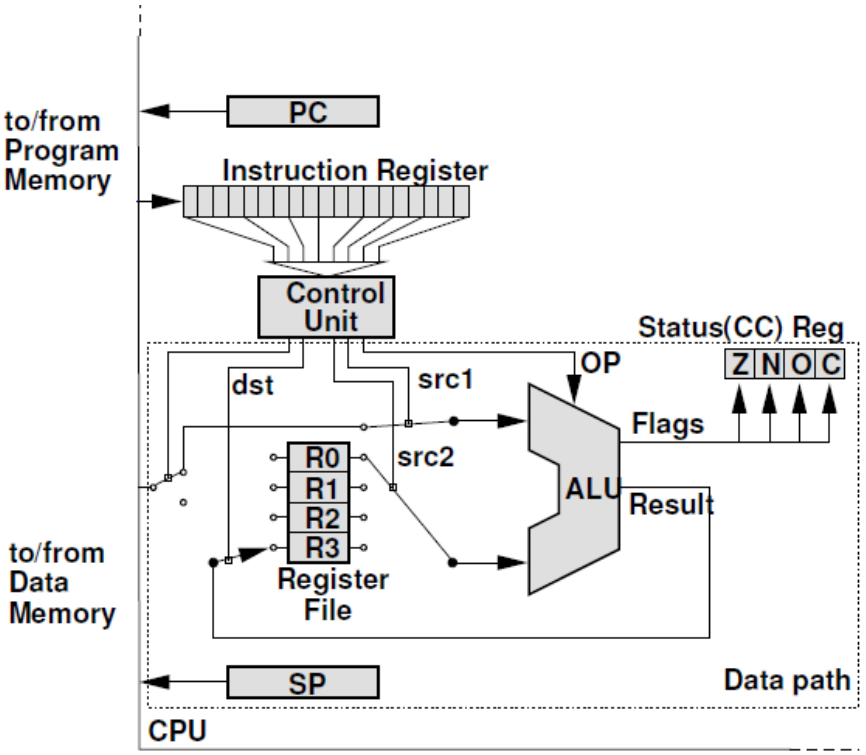
- **Mikroprocesor:** To je normalni CPU, ki ga lahko najdemo v osebnih računalnikih. Komunikacija med perifernimi napravami se vrši preko glavnega vodila. Vse zunanje enote (spomin, USB, časovniki) so povezani z glavnim vodilom. Mikroprocesor ne more delovati sam, saj potrebuje periferne naprave, kot je spomin, vhodno/izhodne enote itd.. Mikroprocesor ni mikrokrmilnik.
- **Mikrokrmilnik**: Vsebuje vse periferen enoto in lahko deluje samostojno. Namensko je načrtan za vodenje in nadzor sistemov. Glede na mikroprocesor ima mikrokrmilnik integrirane periferne enote v samem čipu.
- **Mešani signalni krmilnik:** Mešani signalni krmilnik lahko procesira tako analogne, kot digitalne signale.
- **Vgrajeni sistem:** Veliko področje uporabe mikrokrmilnikov so vgrajeni sistemi. Termin vgrajeni sistem pomeni, da je krmilnik vgrajen v sistem ali napravo. Za primer vzemimo telefon, robot, avtomobil itd.. To so sistemi, katerih delo in delovanje nadzorujejo krmilniki.
- **Sistem realnega časa:** Pomeni, da je sistem nadzorovan in voden v točno določenem času. Prav tako pomeni, da se reakcija sistema vrši točno v določenih časovnih intervalih – realni čas. Časovni interval pogosto vrednotimo, kot sistemski čas, ki ga določa mikrokrmilnik ali procesor.
- **Digitalni signali procesor DSP:** To so procesorji, ki so ob aritmetični logični enoti seštevanje in odštevanje sposobni v enem ciklu izvršiti še množenje in deljenje. Takšni procesorji so namenjeni digitalnemu signalnemu procesiranju (Fouriereva transformacija, Izračun korelacije, avto korelacije in konvolucije). Vodilni proizvajalci DSP krmilnikov so: Texas Instruments, STMicroelectronics, Freescale, Microship, Nxp Semiconducter itd.

8.3. Struktura mikrokrmilnika

V tem poglavju si bomo pogledali zgradbo in delovanje poglavitnih elementov mikrokrmilnika.

8.3.1. Jedro procesorja

Jedro procesorja CPU je glavni del vsakega mikrokrmilnika. Slika 4. prikazuje osnovno strukturo procesorja.



Slika 4. Struktura procesorja

Jedro CPU-ja je sestavljeno iz podatkovne poti ('Data path'), katera izvršuje inštrukcije iz kontrolne enote, ki krmili podatkovno pot. Jedro CPU-ja aritmetično logična enota- ALU, ki je sposobna opravljati le osnovne računske operacije, kot so; seštevanje, odštevanje ter bitni komplement. V osnovi ALU vzame dve vrednosti in jih vrne na izhod. ALU shranjuje tudi stanje v statusni register (Status CC Reg.) Oznake v statusnem registru pomenijo Z-rezultat je nič, N-rezultat je negativen, O-operacija je povzročila preliv 'Overflow', C- operacija prenaša podatek 'Curry'.

Naloga kontrolna enota ('Control Unit') je, da izvršuje inštrukcije in določa katera instrukcija se bo vršila v danem trenutku. Kontrolna enota prav tako shranjuje indeks inštrukcij v programski števec-PC ('Program Counter') ter vsebino inštrukcije v inštrukcijski register- IR ('Instruction Register'). Inštrukcija določa katera vrednost iz

registra bo poslana v ALU in kam bo shranjena. Glede na strukturo kontrolen enote tako ločimo dve arhitekturi.

- RISC: RISC ('Reduced Instruction Set Computer') arhitektura je preprostejša, katera za izvedbo vzame le nekaj urinih ciklov. Prednost RISC sistemov je, da uporabljajo točno določeno dolžino mikro-kode za naslavljanje in inštrukcije. Kot rezultat je hitro izvajanje inštrukcij, ampak te se relativno male.
- CISC: CISC ('Complex Instruction Set Computer') struktura, ki jo upravlja kompleksna mikro-kodna inštrukcija. Takšna inštrukcija za izvedbo potrebuje več urinih ciklov. Inštrukcija ima variabilno dolžino in omogoča mnoge učinkovite inštrukcije napram RISC strukture.

Izbira katera struktura je boljša je odvisno od aplikacije za katero potrebujemo krmilnik. Če rešitev pogosto potrebuje kompleksne inštrukcije potem je smiselno uporabiti CISC strukturo. Seveda je hitrost izvajanja inštrukcije povezana z glavno uro CPU-ja. Jedro mikrokrmilnika pogosto sestavlja RISC struktura, kjer računalniški procesorji temeljijo na CISC strukturi.

Na sliki 4. je inštrukcijski in podatkovni pomnilnik prikazan, kot dva ločena subjekta. To ni vedno stalnica, ampak si inštrukcijski in podatkovni pomnilnik lahko delita isti pomnilnik. Glede na tip pomnilnika ali je ločen ali skupen, ločimo dve vrsti arhitekture procesorja.

- **Von Neumann-va arhitektura:** Pri tej arhitekturi je inštrukcijski in podatkovni pomnilniki skupen. Prav tako je skupno vodilo, s katerim dostopamo do pomnilnika. Na žalost pri tej arhitekturi lahko pride do konflikta med inštrukcijo ter podatki, ki lahko povzroči neželeno zakasnitev sistema. To slabost je splošno znana kot Von Neumann-ovo ozko grlo ('Von Neumann bottleneck').
- **Harvard-ska arhitektura:** Pri tej arhitekturi sta pomnilnika ločena, kjer je ločeno tudi vodilo. Pri tej arhitekturi ni možen konflikt med podatkom in inštrukcijo, kar posledično izboljša učinkovitost procesorja. Slabost sistema je ta, da ta arhitektura potrebuje več komponent, kot so dvojni pomnilnik, dvojno vodilo ter enoto, ki omogoča hkratno dostopanje do podatkovnega ter inštrukcijskega pomnilnika.

Hitrost izvrševanja nalog CPU-ju je odvisna od različnih faktorjev. V glavnem je hitrost odvisna od arhitekture CPU-ja, kar pomeni ali je sistem RISC ali CISC. Prav tako je hitrost izvajanja odvisna od dolžine inštrukcije (8,16,32,64 bitov). Za primer vzemimo 32bitna inštrukcija se izvede v enem ciklu na 32bit CPU-ju, kar je hitreje če se ista inštrukcija izvede na 8bit CPU-ju. Osem bitni CPU za izvedbo inštrukcije potrebuje štiri cikle. Na koncu je hitrost izvajanj inštrukcij pogojeno z glavno uro (zunanji ali notranji kristal), kar pomeni da 160MHz uri inštrukcije izvajajo hitreje, kot pri 10MHz uri.

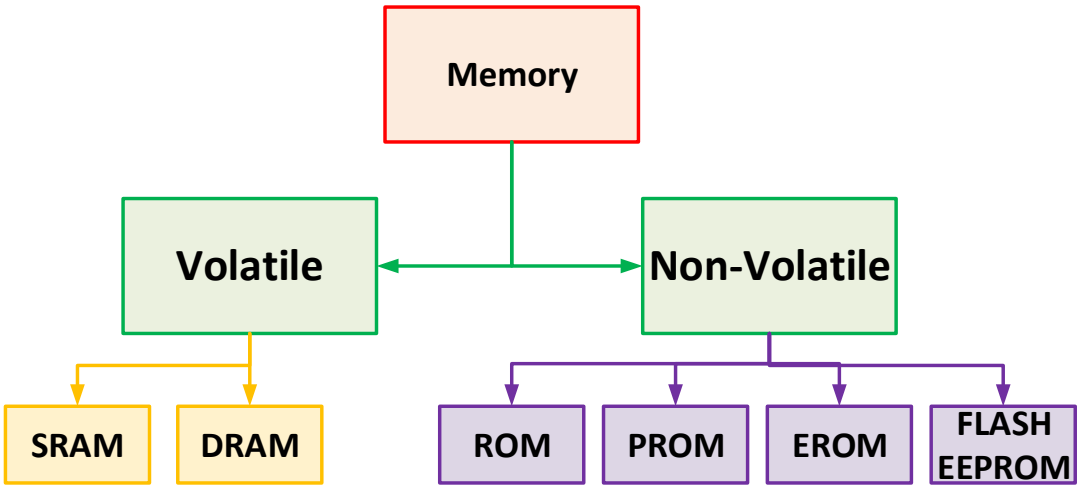
8.3.2. Pomnilnik

V prejšnjem poglavju smo se ukvarjali s strukturo procesorja, ki za svoje delovanje potrebuje pomnilnik. Glede na nalogo pomnilnika ter strukturo ločimo tri vrste pomnilnikov:

- **Register:** Registerski pomnilnik je relativno majhen pomnilnik vgrajen v CPU. CPU potrebuje pomnilnik za zapis stanj CPU-uja ter nastavitve perifernih enot, kot so ADC, DAC, I2C itd.. Ta pomnilnik imenujemo tudi kratkoročni pomnilnik, saj pri izkopu napajanja izgubim shranjeno vrednost pomnilnika.
- **Podatkovni pomnilnik:** V podatkovni pomnilnik lahko podatke shranjujemo dolgoročno in je ponavadi dodan k CPU-ju, kot periferna enota. Podatkovni pomnilnik je bistveno večji, kot register.
- **Inštrukcijski pomnilnik:** Enako, kot podatkovni pomnilnik je relativno velik in je implementiran ob CPU-ju, kot periferna naprava. Pri Von Neumann-ovi arhitekturi sta inštrukcijski in podatkovni pomnilnik ena periferna naprava s skupnim vodilo. Pri mikrokrmilniških sistemih je pomnilnik implementiran na skupnem pomnilniku ('Single Chip Memory'), le da je ločen po sektorjih.

Zgoraj nevedni pomnilniški načini, so najpogostejši načini uporabe pomnilnika CPU-ja. Prav tako obstajajo še drugi pomnilniški načini in registri, kot so 'Pipeline'- registri, 'Cachs'-registri in različni 'buffer-ji'. Glede na mikrokrmilniške strukture je velikost pomnilnika integrirana v čip in je fiksna. Pomnilnika ni možno dodati ali povečati. Zato so mikrokrmilniki namenjeni zgolj za preprostejše naloge.

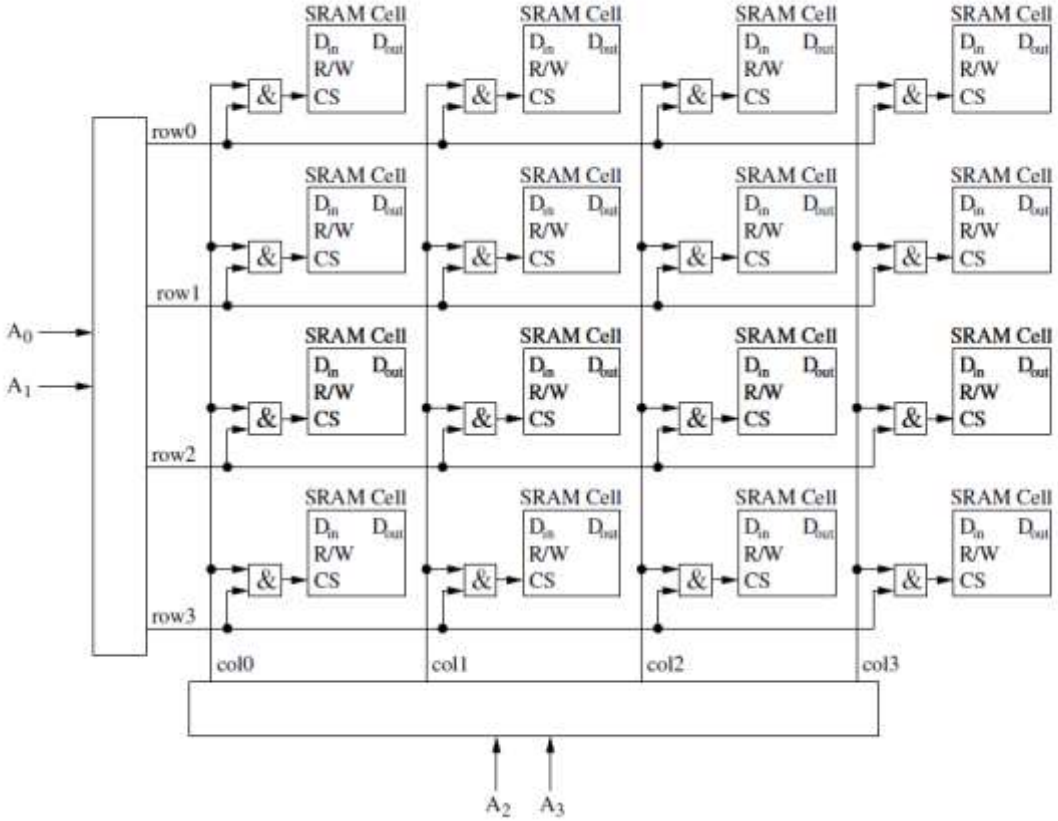
Sedaj smo obravnavali pomnilnik kot enoto, ki je služila določenim nalogam CPU-ja. Iz programskega stališča lahko pomnilnik karakteriziramo, kot trajni in ne trajni pomnilnik ('Non-Volatile memory ' in 'Volatile'). Netrajni pomnilnik lahko razvrstimo, kot dinamični ali nedinamični. Za trajni pomnilnik uporabljamo kratice glede na strukturo in tip delovanja; ROM, PROM, EPROM, EEPROM, FLASH. Slika 5. prikazuje razvrstitev pomnilnikov.



Slika 5. Tip pomnilnikov

Netrajni pomnilnik je tip pomnilnika, ki hrani podatke tako dolgo dokler ne izključimo napajanja ali resetiramo sistem. Poraja se vprašanje zakaj ne uporabljamo samo trajnega pomnilnika? Odgovor je preprost. Trajni pomnilnik je počasnejši, kot ne trajni pomnilnik in zahteva več operacijskega časa za shranjevanje, brisanje ter branje. Če naredimo grobo časovno primerjavo. Branje ne trajnega pomnilnika se meri v nanosekundah, pri čemer branje trajnega pomnilnika traja v milisekundah.

- **Statični RAM-SRAM:** Beseda RAM izhaja iz termina ('Random Access Memory'), kar pomeni, da lahko naključno dostopamo do lokacij pomnilnika. Vsaka lokacija je dostopna s pomnilniško adreso. Nasprotje RAM-u poznamo pomične registre, kjer se podatki lahko berejo/pišejo le sekvenčno. Takšen pomnilnik imenujemo pomično registri tipa FIFO, FILO, LIFO, LILO itd.. ('FIFO- 'First IN first OUT', LIFO-'Lasti IN first OUT' 'itd.). SRAM pomnilnik za shranjevanj podatkov uporablja D-flipflop celice. D-flipflop je v osnovi sestavljen iz vsaj šestih tranzistorjev. Vsaka D-flipflop celica lahko shrani en bit (0 ali 1) in vsaka celica ima naslov. Slika 6. prikazuje 16 bitni matrična pomnilnik z D-flipflopi. $A_{0,1}$ in $A_{2,3}$ pomeni vrstični/stolpčni naslov.



Slika 6. 16bit SRAM pomnilnik.

- **Dinamični RAM –DRAM:** Glede na SRAM, dinamični RAM dosega bistveno višje kapacitete. Sedaj vemo, da SRAM za shranjevanje 1 bita potrebuje vsaj šest tranzistorjev. Za višjo kapaciteto SRAM-a potrebujemo več celic,

ki znatno povečajo fizično velikost čipa in ceno. Če je možno reducirati pomnilnik, lahko uporabimo preprostejši način shranjevanje bitne vrednosti. DRAM shranjuje podatke v shranjevalniku električnega naboja (kondenzator). Takšen način znatno zmanjša število elementov za shranjevanje enega bita ('D-flipflop') in tako povečamo kapaciteto pomnilnika pri bistveno manjši velikosti čipa. Če želimo shraniti logično vrednost v DRAM pomnilniku moramo napolniti kondenzator na določeni lokaciji. Enako kot SARM tudi DRAM uporablja naslove za naslavljanje posameznih celic. Glede na preprostejšo strukturo in nižjo ceno ima DRAM pomnilniki, določene slabost napram SRAM pomnilnika. DRAM pomnilnik je počasnejši. Prav tako za vzdrževanje pomnilnika potrebujemo osveževanje. Osveževanje je potrebno, ker se kondenzator čez čas izprazni.

V nasprotju z SRAM-om in DRAM-om uporabljamo tudi trajni pomnilnik. Trajni pomnilnik je uporabljen takrat, ko želimo podatek shraniti za dalj časa tudi takrat, ko sistem ni napajan.

- **ROM:** Kratic ROM pomeni ('Read Only Memory ') bralni pomnilnik. Bralni pomnilnik pomeni, da ne moremo shranjevati nobenih podatkov. Ponavadi se v ROM pomnilnik shranijo podatki iz strani proizvajalca in so ključni za delovanje naprave. V ROM-u je shranjen BIOS za osebne računalnike ali vrednosti registrov za nastavitev mikrokrmilnika.
- **PROM:** Pomnilnik ROM se uporablja le pri masovnih proizvodnja naprav, drugače je njegova proizvodnja za malo količino ali pilotske projekte predraga. Alternativa ROM-u je PROM ('Programmable Read Only Memory'). PROM je možno programirati le enkrat, saj vsebuje varovalko, ki prepreči večkratno pisanje v PROM. PROM poznamo tudi pod kratico OTP ('One Time Programmable memory'). PROM ni primeren za razvoj, kjer se vsebina pomnilnika ter programa spreminja.
- **EPROM:** EPROM je kratica za ('Erasable Programmable Read Only Memory'). EPROM je v bistvu bralni pomnilnik, ki ga lahko večkrat preprogramiramo. Programiranje EPROM zahteva poseben postopek, zato ga ponavadi ne moremo izbrisati iz strani mikrokrmilnika. Vrednosti EPROM-a je shranjena v FET ('Field effect transistors'). FET tranzistor se vodi s pinom-GATE. Visoka napetost na pinu-gate zapira tranzistor. Ko vrata enkrat zapremo, vrata ostanejo zaprta tudi če vrat več ne napajamo. Tako lahko shranimo digitalno vrednost 1 ali 0. Zaradi nepravilnosti v FET-u se vrata čez čas lahko odprejo. Proizvajalci ponavadi podajo, kako dolgo lahko shranimo podatek v EPROM-u. Ta čas je ponavadi nekaj deset let. Vsak EPROM ima okno (odprtino), katera nam omogoča brisanje. Brisanje ERPOMA se izvede skozi okno z ultra vijolično svetlobo UV. UV svetloba nam odpre FET-e, kar pomeni da je pomnilnik izbrisan. Brisanje EPROM traja nekje 30-40 minut.
- **EEPROM:** EEPROM pomeni ('Electrically Erasable and Programmable ROM') elektronsko programljiv pomnilnik. V osnovi EEPROM deluje na enak način, kot EPROM le, da pri brisanju pomnilnika ne potrebujemo posebnih zunanjih visokih napetostih in UV svetlobe. Visoka napetost za

vzbujanje FET je vgrajena znotraj čipa in se imenuje črpalka naboja ali 'Charge pump'. Ponavadi ima EEPROM enko, kot EPROM življenjski cikel, ki je pogosto omejen na 100 000 ciklov brisanja.

- **FLASH:** FLASH je okrnjena verzija EERPOM pomnilnika in deluje na enak način kot EEPROM. Razlika med FLASH pomnilnikom in EEPROM-om je ta, da ne moremo brisati posamezne celice posebej, ampak samo celi pomnilnik ali določeni sektor. Razlog za vpeljavo FLASH pomnilnika je visoka cena EEPROM pomnilnika. Prav tako, kot EEPROM ima tudi FLASH omejeno število vpisov.
- **NVRAM:** NVRAM je kombinacija stalnega in nestalnega pomnilnika ('Non-Volatile RAM'). Takšen pomnilnik ima enak princip delovanja kot SRAM le, da ima dodano napajalno baterijo. Prav tako obstaja različica pomnilnika, kjer sta združena EERPOM in SRAM pomnilnik v istem čipu.

Kadar operiramo s podatki iz pomnilnika je pomembno poznati način zapisa v pomnilnik. Za zapisovanje podatkov poznamo dva pristopa:

- **Big Endian:** Je način zapisa ali branja podatkov, kjer so višji bajti shranjeni na začetku pomnilniške lokacije. Na primer če shranimo podatek '0x7799' na naslovu pomnilnika '0x00' in '0x01'. Pri Big Endiand se na naslov '0x00' shrani vrednost '0x77' in na naslov '0x01' podatek '0x99'.
- **Little Endian:** Je način zapisa, kjer se višji bajti shranijo na višjih lokacijah. Iz prejšnjega primera. Se podatek '0x7799', del '0x77' shrani na lokacijo '0x01' in '0x99' na lokacijo '0x00'.

8.4. Periferne enote

V prejšnjih poglavjih smo obravnavali razliko med mikrokrmilnikom in procesorjem. V tem poglavju bomo predstavili periferne enote, ki so najpogosteje integrirane v čipu sodobnega mikrokrmilnika.

8.4.1. Digitalni vhodi in izhodi

Digitalni vhodi in izhodi – I/O se uporabljajo za krmiljenje in nadzor zunanjih naprav in so poglavitna enota znotraj mikrokrmilnika. Kot posledica, vsak mikrokrmilnik ima vsaj 1-2 digitalna I/O pina, ki se lahko povežeta na zunanje naprave. Ponavadi imajo mikrokrmilniki več kot 32 I/O-jev, ki jih lahko uporabimo za različne namene. Pri povezovanju I/O-jev je potrebno paziti na dopustne napetostne nivoje, ki jih dopušča posamezni mikrokrmilnik. Napetostni nivoji so ponavadi strogo povezani z napajalno napetostjo mikrokrmilnika. Najpogostejši napajalni napetosti sta 5 in 3.3 volta.

Digitalno vhodi in izhodi so pogosto grupirani v porte. Vsak port lahko vsebuje 8, 16 ali 32 I/O pinov. Grupiranje števila pinov v posamezni port je odvisno od proizvajalca ter dolžine kontrolnih registrov mikrokrmilnika. Najpogosteje so vsi I/O pini dvosmerni, kar pomeni, da se lahko uporabljajo, kot vhodi ali izhodi. Večina I/O pinov ima tudi

dodatne funkcionalnosti, kot so števci, zunanje prekinitve, komunikacijski vmesniki ter analogno digitalno ADC ali digitalno analogno DAC pretvorbo. Vse funkcionalnosti I/O porta je možno nastaviti v programi preko kontrolnih registrov posameznega porta.

Digitalni vhodi izhodi se imenujejo zato, ker napetostni potencial na vhodu pina pretvorijo v logično '1' ali '0'. Napetostni nivoji logične '0' in '1' so določeni na naslednji sliki 7. Posplošeno lahko govorimo da je logična ničla nekje napetostni potencial pod 0-1V. Logična '1' je napetostni potencial nad 2.4V-Vcc. Vcc je napajalna napetost mikrokrmilnika. Napetostni potenciali variirajo od proizvajalca do proizvajalca ter načina I/0-pina ali je uporabljen, kot vhod ali izhod [2].

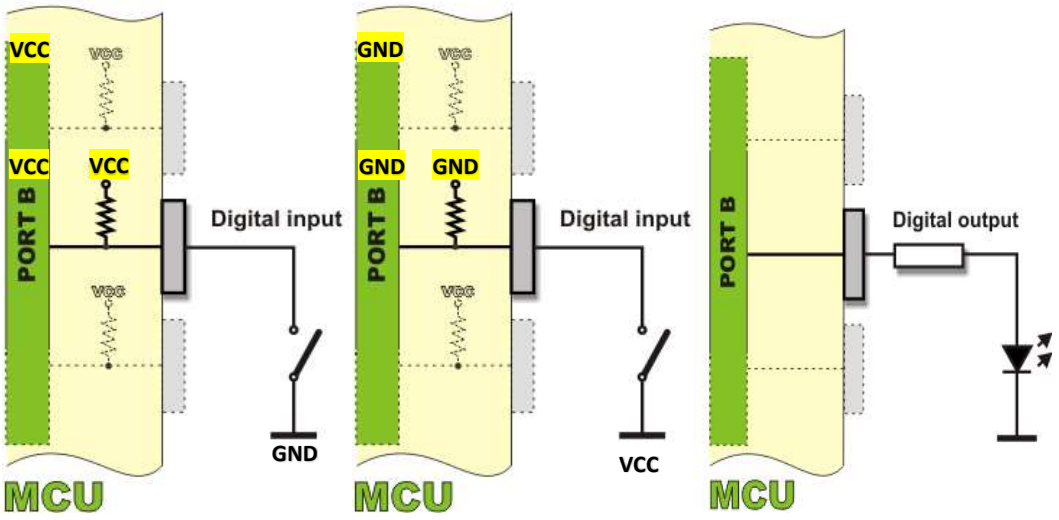
GPIO input/output pin electrical characteristics	
Output low voltage V_{OL}	< 0.40 V [1] < 0.66 V [2] < 0.40 V [3] < 0.40 V [4]
Output high voltage V_{OH}	> 2.40 V [5] > 2.64 V [6] > 2.90 V [7]
Input low voltage V_{IL}	< 0.80 V [8] < 0.54 V [9] < 1.15 V [10]
Input high voltage V_{IH}	> 2.00 V [11] > 2.31 V [12] > 2.15 V [13]
Hystereses	> 0.25 V [14] 0.66 - 2.08 V [15]
Schmitt trigger input low threshold V_{T-}	1.09 - 1.16 V [16] 0.9 [17]
Schmitt trigger input high threshold V_{T+}	2.24 - 2.74 V [18] 0.90 V [19]
Pull-up/down resistance	40 - 65 KΩ [20] 100 KΩ [21]
Pull-up/down current	< 50 uA [22] < 28 uA [23]
Pin capacitance	5 pF [24]
Bus hold resistance	5-11 KΩ [25]

Slika 7. Napetosti potenciali I/O pinov različnih proizvajalcev.

Mikrokrmilniki najpogosteje uporabljajo pozitivno logiko, kar pomeni da je logična '1' potencial nad 2.4V in logična '0' pod 1V. Pri negativni logiki pomeni, da je logična '1' pod 1V in logična '0' nad 2.4V. Pogosto se uporabljajo mešani sistemi, kjer želimo povezati mikrokrmilnike z različnimi napajalnimi napetostmi. Na primer imamo sistem napajan z 5V in 3.3V. Pogosto proizvajalci mikrokrmilnikov z nižjo napajalno napetostjo proizvedejo I/O-je, ki so 5V tolerantni. Lastnosti pinov ter električne karakteristike pinov so opisane v podatkovnih listih ('Datasheet') za vsako družino in proizvajalca posebej.

Pogoste se pri preprosti nastavitvi I/O pinov srečamo s terminom 'Pull-up' , 'Pull-down' ali 'Floating'. Termin 'Pull' pomeni, da I/0 pinu določimo privzeti napetostni potencial, ko na pinu nimamo zunanjega potenciala. Pri načinu 'Pull-up' je privzeti potencial napajalna napetost pri 'Pull-down' je privzeti potencial GND. Termin 'floating' pomeni, da pin pustimo plavajoč, kar pomeni, da nimamo privzetega potenciala. V tem primeru je pin podvržen zunanjim vplivom, kar lahko povzroči neželene motnje v

sistemu, če pin uporabljajmo, kot diskretni vhod ali izhod. Plavajoč način se pogosto uporablja pri ADC ali DAC pretvorbi. Slika 8. prikazuje način 'Pull-up', 'Pull-down' in 'Floating'.



a) *b)* *c)*

Slika 8: Način I/O pinov; a) 'Pull-up', b) 'Pull-down', c) 'Floating'

8.4.2. Analogni vhodi in izhodi

Glede na digitalne vhode in izhode je z analognimi vhodih/izhodih možno brati ali procesirati različne napetostne nivoje. Branje analognih vrednostih je povezano z ADC periferno enoto v krmilniku. Osnovni podatki ADC pretvornika so resolucija, hitrosti pretvorbe in tip pretvorbe. Resolucija ADC-ja je podana s številom bitov. Iz števila bitov lahko določimo na koliko napetostnih nivojev je razdeljeno napetostno območje ADC-ja. Privzeto napetostno območje ADC-ja je ponavadi napajalna napetost ADC-ja. Večina mikrokrmilnikov ima nastavljivo napetostno območje, ki jo lahko pripeljemo iz zunanjega referenčnega vezja. Prav tako imajo mikrokrmilniki nastavljivo resolucijo ADC-ja. Kar pomeni, da lahko izbiramo med 6,8,10,12 bitno resolucijo pretvornika. Vzemimo primer, uporabljam 12bitni ADC ter napajalno napetost 3.3V. Najmanjšo napetost, ki jo lahko izmerimo z ADC-jem je podana s spodnjih izrazom,

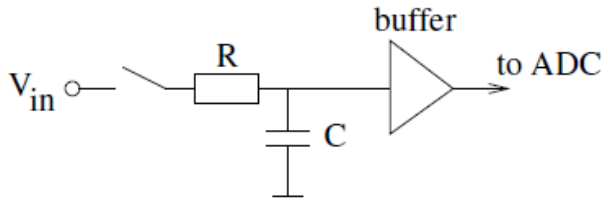
$$V_{RES} = \frac{V_{REF}}{2^N - 1},$$

,kjer je V_{REF} napetostno območje ADC-ja, N-število bitov in V_{RES} napetostna resolucija pretvornika. Z dan primer (12bit, 3.3V) je napetostna resolucija pretvornika V_{RES}= 0.805mV. Mnogi pretvorniki omogočajo nastavitev tako zgornje in spodnjo napetostno reference.

To pomeni, da lahko polno resolucijo pretvornika uporabimo med dvema potencialoma. Vzemimo primer meritve napetostnih potencialov med 1.5V in 2.2V. Z nastavitvijo V_{REF-}=1.5V in V_{REF+}=2.2V lahko 12bitno resolucijo razdelimo le med 0.7V (V_{REF+} - V_{REF-}). Resolucija pretvornika med potencialoma 1.5-2.2V je sedaj, $V_{RES} = \frac{V_{REF+} - V_{REF-}}{2^N - 1} = \frac{2.2 - 1.5}{2^{12} - 1} = 0.171mV$.

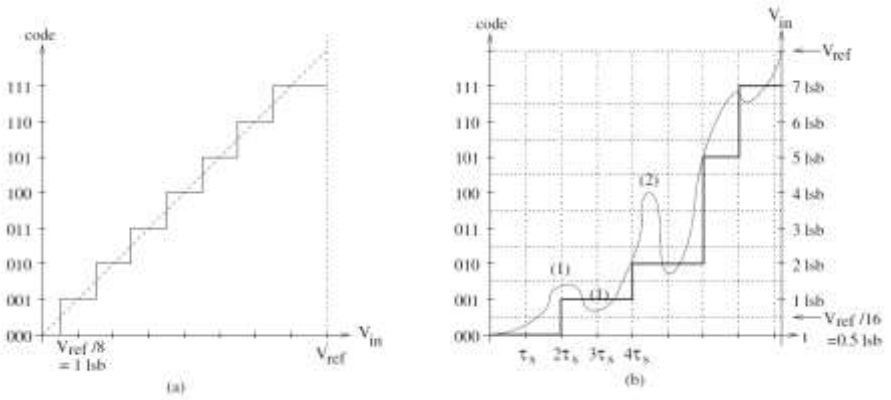
Naslednji ključen podatek je hitrost pretvorbe, ki se ponavadi podaja v številu odtipkov na sekund ('sps – samples per second'). Hitrost tipanja ADC pri mikrokrmilnik znaša od nekaj 100ksps do nekaj 3Msps odvisno od proizvajalca in družine krmilnika. Podatek 3Msps pomeni, da ADC naredi 3 milijone odtipkov v eni sekundi. Hitrost tipanja je ključni podatek pri digitalnem procesiranju signalov.

Zajemanje ADC signala lahko razdelimo v tri faze. Na začetku tipanja uporabimo držalnik signala ('sample & hold'). Držalni signala drži signal tako dolgo, dokler traja pretvorba. Držalnik signala slika9. predstavimo, kot stikalo in kondenzator, v katerega shranimo napetost tipanega signala.



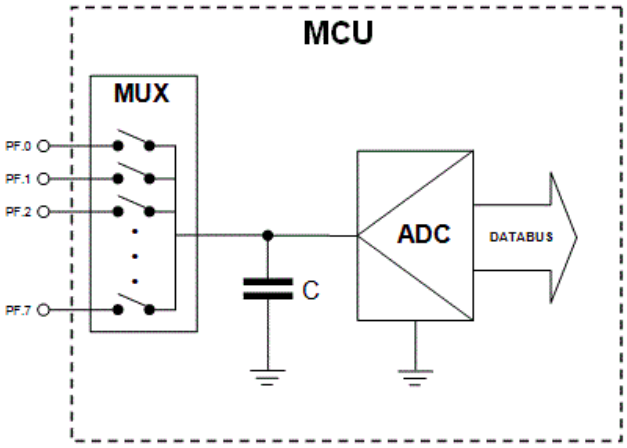
Slika 9: 'Sample and hold' shema

Druga faza pretvorbe je kvantizacija. Kvantizacija je postopek, kjer razdelimo napetostne nivoje merilnega območja. En kvant ADC-ja predstavlja resolucijo ADC-ja, ki smo jo predstavili s prejšnjim izrazom. Ker je postopek kvantizacije zaokrožanje prave analogne vrednosti pomeni, da s tem vnašamo pogrešek meritve. Pogrešek imenujemo tudi kvantizacijski pogrešek ADC pretvornika. Zadnji korak pretvorbe je kodiranje, kjer vsakem kvantiziranemu nivoju delimo binearno vrednost. Na sliki 10. je prikazan postopek kvantizacije 3 bitnega ADC-ja s časom tipanja τ_s.



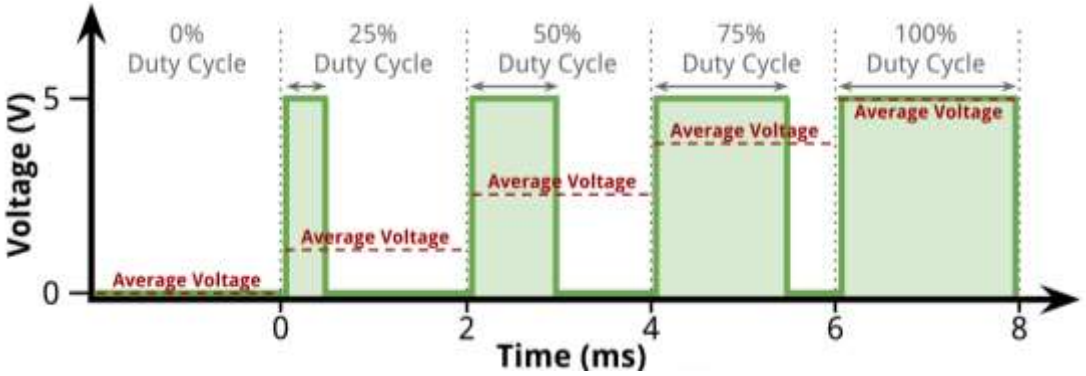
Slika 10: Postopek kvantizacije: a) Napetostni nivoji ADC-ja 0-V_{REF}, b) Kvantiziranje signala.

V mikrokrmilniku je ADC enota povezna z I/O pini kontrolerja. Tako lahko beremo več različnih analognih signalov le z enim ali dvema ADC-jema. Vhodi v ADC so multipleksirani. Multipleksirane pine imenujemo tudi kanali ADC-ja. Naprednejši mikrokrmilniki vsebujejo 3-4 ločene ADC-je, kjer preprostejši imajo le en 1 ADC. Slika 11. prikazuje multipleksiranje vhodih pinov na ADC-ju.



Slika 11: Multipleksirani vhodni pini ADC-ja.

Funkcija digitalno-analognega pretvornika DAC je, da pretvori binearni vrednost v analogni signal. DAC se največkrat uporablja za procesiranje poljubnih izhodih signalov iz krmilnika. DAC ima podobne karakteristike, kot ADC. Resolucija DAC-je je določena s resolucijo (številom bitov) ter hitrost vzorcev na sekundno ('SPS- samples per second'). Napetostni nivo izhodnega signala je določen z napetostno referenco, ki je privzeto nastavljena na napajalno napetost krmilnika. DAC enote vsebujejo le naprednejši krmilniki in število enot je ponavadi manjše, kot število ADC-jev. Če krmilnik nima DAC enote, analogni signal lahko procesiramo s pulzno širinsko modulacijo-PWM, slika 12 ('PWM-pulse width modulation'). PWM signal se generira s fiksno frekvenco in nastavljivim prevajalnim razmerjem ('Duty cycle'). S prevajalnim razmerjem generiramo želen napetostni potencial na izhodnem pinu. Povprečna vrednost ene periode PWM signala predstavlja izhodno analogno vrednost.



Slika 12: Pulzno širinska modulacija.

8.4.3 Prekinitvene rutine

Programi, ki se izvajajo na mikrokrmilniških sistemih moraj reagirati na določene dogodke. Dogodki se med seboj razlikujejo po dolžini trajanja ali so ponovljivi in po kompleksnosti. Tako mora krmilniki reagirati že na posamezno spremembo potenciala na vhodnem pinu, kakor tudi na prejemanje ali oddajanje podatkov preko komunikacijskih vmesnikov.

Prekinitvena rutina pomeni, da krmilnik prekine izvajanje glavnega programa in se posveti prekinitveni funkciji. Če uporabljamo več prekinitvenih rutin je potrebno nastaviti prioritete. Prioriteta določa, katera rutina ima prednost pri izvajanju. Pri snovanju program je smiselno razvrstiti prekinitvene rutine po prioritetah in kako se bodo izvajala. Če uporabljamo rutine, ki se morajo izvrševati v točno določenem času je smiselno, da ji nastavimo visoko prioriteto. V nadaljevanju bomo predstavili nekaj najpogostejših prekinitvenih rutin.

- **Časovna prekinitev:** Časovna prekinitev je vezana na uro časovnika in se izvaja skladno s števcem časovnika. Časovna prekinitev se izvrši ob točno določenem času. Prekinitev je lahko periodična ali nadzorovana preko statusnega registra prekinitve. Nadzor prekinitve pomeni, da jo omogočimo takrat, ko jo potrebujemo. Časovne prekinitve se pogosto uporabljajo za merjenje časa in sinhronizacijo različnih periodičnih opravil.
- **Prekinitev števca:** Prekinitev števca je v osnovi enaka, kot časovna prekinitev, le da ta ni vezana striktno na čas. Prekinitev se izvede, ko števec doseže določeno vrednost. Štetje števca ponavadi prožijo zunanje naprave preko diskretnih vhodov. Prekinitev števca se pogosto uporablja za štetja pulzov.
- **Zunanja prekinitev:** Zunanja prekinitev je prekinitev, ki se sproži takrat, ko se na vhodu spremeni napetostni potencial. Zunanja prekinitev bere samo logično '0' ali '1'. Prekinitev lahko deluje tako, da zaznava pozitivno ali negativno fronto diskretnega vhoda.
- **Prekinitev ob prejemanju podatkov:** Prekinitev ob prejemanju podatkov služi za sinhronizacijo s pošiljateljem. To pomeni, da rutina zazna, kdaj so podatki na vodilu. Prekinitvena rutina se pogosto uporablja pri asinhronem prejemanju podatkov. Asinhrono prejemanje podatkov pomeni, da oddajni podatkov ne pošilja periodično ob določenem času, ampak naključno. Prekinitev skrbi, da ne zgrešimo prejemanja podatkov.
- **Prekinitev ob pošiljanju podatkov:** Prekinitev ob pošiljanju podatkov skrbi, da se podatkovni paket pošlje v danem trenutku. Prekinitev skrbi, da krmilnik ne opravlja drugega dela razen pošiljanja podatkov.
- **Prekinitev ADC-ja:** Prekinitev ADC je čas, ki je potreben ADC prebere analogno vrednost. Prekinitev služi, da krmilnik počaka ADC pretvorbo in nato nadaljuje s drugimi opravili. Prekinitev ADC-ja je smiselna, kadar za naslednja opravila potrebujemo podatek iz ADC-ja.
- **Prekinitev DAC-ja:** Podobno, kot prekinitev ADC-ja ima enako vlogo prekinitev DAC. DAC proži prekinitev, le kratek čas, katerega potrebuje za pretvorbo binarne vrednosti v analogni izhod.
- **Prekinitev stanje pripravljenosti ('Sleep mode'):** Prekinitev stanje pripravljenosti je uporabna v primeru, ko krmilnik ne izvaja operacij konstanto. Takrat lahko nastavimo krmilnik, da gre v stanje pripravljenosti. V stanju pripravljenosti ima krmilnik nizko porabo energije. Prekinitev stanje pripravljenosti služi, da krmilnik večino energije porabi le za spremljanje določenega zunanjega dogodka. Vse ostale periferne naprave

so izključen. Ob želenem dogodku prekinitev vzbudi krmilnik in ta deluje z vsemi potrebnimi perifernimi enotami.

Prekinitvena rutina je delo programske kode, ki se izvede ob prekinitvi. V prekinitveni rutini je pomembno, da pobrišemo prekinitveno zastavico. Prekinitvena zastavica je bit v statusnem registru, ki se postavi na vrednost '1', ko je prišlo do prekinitve. Če želimo, da se prekinitev izvede ponovno je potrebno zastavico ponovno postaviti na '0'. Zastavico postavimo na '0' znotraj programske kode, ki je del prekinitven rutine. Pri pisanju programa prekinitven rutin moramo biti še posebej pozorni, da se koda ne izvaja dlje časa, kot je možnost ponovne prekinitve. To je še posebej pomembno pri prekinitvah, ki se izvajajo periodično in hitro.

Podali bomo nekaj ključnih dejstev, kdaj je smiselno uporabljati prekinitvene rutine:

- Naloge so morajo izvajati naključno.
- Dolgi časovni intervali med dvema nalogama.
- Sprememba stanje je pomembna.
- Spremljanje kratkih pulzov.
- Za polno operativnost krmilnika je potrebno izvajati le nekaj nalog.
- Dogodki so generirani iz strani drugi naprav. Ni prisotnih špic v signalu in mehanskega odbojnega efekta ('Bouncing effect').
-

Nekaj napotkov, kadar ne potrebujemo prekinitvene rutine in se naloge izvajajo v glavnem programu:

- Operator je človek.
- Izvajanje dogodkov ni časovno kritično.
- Vhodni impulzi so dolgi.
- Signal je zašumljen.
- Glavni programi ni dolg.

Pri snovanju programa za mikrokrmilnik je potrebno celovito analizirati izvajanje programa. Potrebno je preučiti, kaj se izvaja v prekinitveni rutini in kaj se bo izvajalo v glavnem programu. Pri uporabi večih prekinitev je potrebno biti previden, saj izvajanje prekinitve zahteva dodatno uporabo pomnilnika RAM-a. V primeru, da se hkrati sproži več prekinitev, lahko krmilniku zmanjka pomnilnika in tako lahko izgubimo določene informacije iz prekinitvenih rutin. Razhroščevanje podobnih težav je zelo težavno in časovno potratno.

8.4.3. Števci in časovniki

Števci in časovniki so ključna periferna enota vsakega mikrokrmilnika, ki so pogosto povezani z ostalimi enotami. Mikrokrmilniki imajo integriranih več različnih števcev, kateri se ločijo glede na število bitov (8,16,32 bitini). Časovniki se uporabljajo za različna opravila kot so; zakasnitve, merjenje časa, merjenje frekvence. Najpreprostejša

uporaba časovnika je uporaba samo števca. Časovniki lahko generirajo različne dogodke, prekinitve ali procesirajo moduliran PWM signal.

Vsak časovnik je števec. Števec časovnika lahko šteje navzgor ali navzdol. Način štetja in je možno konfigurirati znotraj kontrolnega registra. Stanje števca se lahko prebere v podatkovnem registru števca. Vsak števec ima svoj kontrolni, statusni in podatkovni register. Velikost števca je določena z njegovo bitno dolžina. Bitna dolžina podaja največje možno število števca 2^N-1, kjer je 'N' število bitov. Na primer 8 bitni števec lahko štej od 0-255 pri čemere 16bitni števec šteje od 0-65535. Če števec povežemo z uro dobimo časovnik. Časovnik proži štetje glede na vhodni takt (uro). Štetje časovnika je lahko proženo na pozitivno ali negativno fronto urinega takta. Takt ure, ki jo uporablja pri časovniku je lahko zunanji ali notranji. V primeru uporabe notranjega takta, je mišljeno, da uporabljamo uro CPU-ja, pa čeprav je ta pripeljan iz zunanjega kristala. Ta način uporabe ure se imenuje asinhronski način ('Synchronus Mode'). Zunanji takt ure časovnika pomeni, da posebej za časovnik dovedemo novo uro. Zunanja ura časovnika ni povezana s taktom CPU-ja, zato se imenuje asinhronski način štetja ('Asynchronus Mode'). Zunanja ura pri časovniku se pogosto uporablja za koledar (ure, dnevi, meseci itd..) Za koledar, takt ure znaša natančno 32.768KHz. Časovnik za koledar imenujemo tudi RTC števec (RTC-'Real Time Clock'). Časovnik je možno konfigurirati na različne načine štetja ter časovno okno štetja. Časovni inkrement časovnika je čas enega inkrementa števca. Časovni inkrement in časovna dolžina štetja je možno nastaviti v registrih časovnika.

Glavni parametri časovnika so:

- **Ura:** Ura časovnika je določena s hitrostjo perifernega vodila za dan števec. Hitrost perifernega vodila je določena v sistemskih nastavitvah krmilnika. Nekateri preprostejši krmilnika nimajo ločenega perifernega vodila za posamezno enoto in je ura enaka vsem perifernim enotam. Prav tako je pri uri potrebno določiti izvor ure, drugače asinhronski ali asinhronski način.
- **Skalirni faktor ('Prescaler'):** Skalirni faktor določa, deljenje ure časovnika. S skalirnim faktorjem določimo inkrement časovnika ter čas trajanja štetja v primeru, da štejemo do konca podatkovnega registra števca. Inkrement časovnika je časovna resolucija časovnika in določa, kako natančno lahko merimo čas. Vrednost skalirnega faktorja je shranjena v PSC-registru.
- **Perioda:** Perioda časovnika določa časovno okno štetja. Pomeni, da je največja dolžina periode enaka dolžini podatkovnega registra števca. Podatkovna dolžina števca je določena s številom bitov časovnika. Perioda časovnika je shranjena v ARR-registru. ARR register je 'Avto Reload register'. Perioda števca določa število preštetih enot znotraj dolžine celotnega ARR-registra. To pomeni, da števec ne šteje samo do maksimalne dolžine ARR-ja ampak samo do vrednosti periode, ki je programsko nastavljiva.

Števci imajo še dva načina izvajanja. Ta dva načina poznamo, kot vhodno zajemanje 'Input Capture' ter izhodno primerjanje 'Output Compare'. Vhodno zajemanje se

uporablja takrat, ko želimo šteti določene dogodke na vhodnih pinih krmilnika. Vsak dogodek spremembe logičen '0' ali '1' na točno določenem vhodnem pinu se iz povečanega števca prenese v 'Input capture' register. Ta register je možno brati tekom programske kode. Način zajemanje vhoda se pogosto uporablja pri branji različnih kodirnikov in enkoderjev. Način izhodnega primerjanja je podobno, kot vhodno zajemanje le da tukaj spremljajo izhoden pine. Spremembe na izhodnih pinih povečajo ali zmanjšajo stanje števca. Način izhodnega primerjanja lahko prav tako proži prekinitev, ko je števec dosegel želeno število inkrementov.

Mikrokrmilniki ob splošno uporabnih števcih vsebujejo tudi poseben števce, ki niso namenja zgoraj opisanemu. Takšne števce imenujemo 'Watchdog timers'-WT in služijo za spremljanje delovanja krmilnika ter potek programske kode. WT-časovnik ima svojo ločeno uro, ki ni vezana na CPU. Delovanje WT-je relativno preprosto. WT nastavimo na časovno okno, ki je lahko v rangu milisekund ali sekund. Ob zagonu števec, ta odšteva proti nič. Vsako opravilo, funkcija, prekinitvena rutina resetira WT števec, kar pomeni, da je ob vsakem resetu časovnika na začetni vrednosti. Če števec ne uspemo resetiramo, ko ta prešteje do nič, ta sproži strojni reset krmilnika ('Kick the dog'). Krmilnik se zažene ponovno. Z WT-časovnikom preprečimo, da bi se krmilnik zaciklal v rutini ali da bi postal neodziven.

8.4.5. Komunikacijski vmesniki

Komunikacijski vmesniki mikrokrmilniku omogočajo komunikacijo z drugimi zunanjimi napravami, kot so drugi krmilniki, PC-ji, senzorji itd.. V tem delu bomo obravnavali le vmesnike, ki so direktno implementirani v čip krmilnika. Komunikacijske vmesnike ločimo glede na različne karakteristike, kot so: paralelni ali serijski vmesnik, sinhronska ali asinhronska komunikacija, točka do točke ali mrežni način, polnodupleksni ('Half-duplex') ali poldupleksni način ('Full-duplex'), žični brez žični. V tem delu se bomo osredotočili le na žične vmesnike.

Serijski vmesnik pošilja en bit v enem urinem taktu. Serijski vmesnik je tako odvisen od hitrosti ure. Glede na frekvenco ure določim število prenesenih bitov na enoto časa (bit/s), kar z drugimi besedami imenujemo hitrost komunikacije. Serijski komunikacijski vmesnik tako potrebuje manj fizičnih podatkovnih linij. Paralelni vmesnik prenaša vsak bit besede po fizično ločenem vodilu. Če prenašamo 8bitni podatek potrebujemo najmanj 8 fizičnih povezav med obema vmesnikoma. Paralelna povezava je hitrejša od serijske. Slabost paralelnega vmesnika je, da zahtev več fizičnih vodil in več pinov krmilna. Paralelna komunikacija se ponavadi uporablja za zelo kratke razdalje, LCD zasloni, branje hitrih ADC-jev in DAC-jev, različni senzorji itd.. Serijska povezava se zaradi fizične preprostosti uporablja tako za kratke kakor tudi za daljše razdalje.

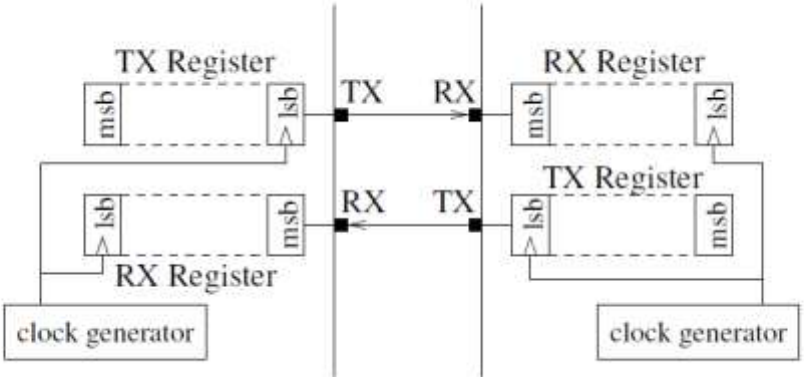
V mnogih primerih je komunikacija med dvema sistemoma dvosmerna. To pomeni, da obe napravi lahko prejemata ali pošiljata podatke. Naslednje vprašanje je, kdaj lahko naprava pošilja ali prejema podatke. Če uporabljamo polnodupleksni način potem obe napravi lahko hkrati prejemat in oddajta. Fizično pomeni, da imata ločeni liniji za oddajanje-Tx in prejemanje-Rx. Pri serijski povezavi to pomeni, da imamo dve žici ena je za pošiljanje druga za prejemanje. Pri 8bitnem paralelnem načinu bi pomenilo 8 žic za pošiljanje in 8 dodatnih žic za oddajanje. Če uporabljamo poldupleksni način

potem napravi prejemata in oddajat po isti liniji, kar pomni, da sočasnost ni dovoljena. Sočasnost bi pomenila kolizijo in izgubo podatkov. Takšen način zahteva še manj fizičnih povezav, toda poveča kompleksnost komunikacijskega protokola.

Pri komunikacijskih vmesnikih se pogost srečamo s terminom gospodar-suženj ('Master-Salve'). Takšen način je pogosto uporabljen pri serijskem načinu. Naprav gospodar določa, kdaj lahko suženj dostopa do vodila in kaj lahko počne na vodilu (prejema ali pošilja). Če se naprave nahajajo v mreži, potem je dopustno, da je le ena naprava gospodar ostale so tej podrejene.

Najpogostejši komunikacijski vmesniki mikrokrmilniki so; USART,SPI,I2C. Nekateri naprednejši krmilniki imajo integrirane tudi kompleksnejše vmesnike, kot so CAN, USB in TCP/IP.

- **USART:** Je serijska povezava (USART- 'Universal Synchronous Asynchronous Receiver Transmitter'), ki za komunikacijo uporablja le dve liniji. Ena linija je za pošiljanje Tx (Tx-transmitt), druga za prejemanje Rx (Rx-'receive'). Razlika med USART-om in UART-om je le v tem, da USART potrebuje ločeno linijo za uro. UART ima prednastavljeno ur, s katero so seznanjene vse naprave na komunikacijski poti. Pri USART-u prejemna naprava uporabi uro pošiljatelja. Slika 13. prikazuje UART komunikacijski vmesnik.



Slika 13: UART komunikacijski vmesnik

Nastavljivi parametri USART serijskega vmesnika so:

- ***Število podatkovnih bitov:*** Število podatkovnih bitov določa koliko bitov bo zajemal poslan podatek. Možno je izbrati od 5-9bitne podatke. Najpogosteje se uporabljajo 8 bitni podatki.
- ***Paritetni bit:*** Uporabnik lahko določi če bo uporabljena pariteta ali ne. Pariteta je lahko soda bit '1' ali liha bit '0'. Pariteta se uporablja, kot preprosti nadzor nad pravilnostjo komunikacije. Primer uporabe sode in lihe paritete pri 7bitnem podatku slika 14.

7 bits of data	(count of 1-bits)	8 bits including parity	
		even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110

Slika 14: Soda in liha pariteta

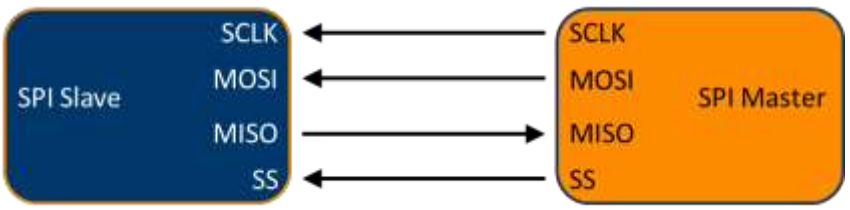
- ***Stop bit:*** Stop bit naznani, kdaj je prenos končan. Stop bit je lahko dol 1 ali 2 bita.
- ***Hitrost prenosa ('Baud Rate'):*** Hitrost prenosa določa uro takta. Višja je ura hitrejši je prenos. Najpogosteje se uporabljene naslednje hitrosti povezave: 9.6kb/s, 38kp/s, 115.2kb/s, 1Mb/s,10Mp/s.

Serijska povezava je poznana tudi pod standardom RS232, RS422, RS485 itd. RS232 je povezava med dvema točkama, kjer je napetosti nivo pozitivne fronte določen med 3-15V. Obe napravi povezani z RS232 morat biti na enakem potencialu. Morata imeti skupni GND. Za uporabo RS232 in UART-a na mikrokrmilniku se uporabljajo vmesniki, ki dvignejo ali spustijo napetostni nivo za standard RS232. Pogosto uporabljen vmesnik je MX232.

RS422 je enaka komunikacija, kot RS232 le da uporablja diferenčne napetostne nivoje. Diferenčni napetostni nivoji omogočajo prenos na daljše razdalje. Napravi ne potrebujeta skupnega GND-ja.

RS485 je podoben standardu RS422 le, da dopušča 32 naprav na istem vodilu, kjer RS232 in RS422 dopuščata le en par. Pri obeh standardih RS422 in RS485 hitrost povezave pada z dolžino povezave.

- **SPI:** SPI je serijska povezava (SPI-'Serial Peripheral Interface'), ki je namenjena za komunikacijo med bližnjimi napravami. Dopuščene razdalje so le nekaj centimetrov do največ enega metra. Komunikacija zagotavlja polnodupleksni način med gospodarjem in sužnjem. Gospodar določa hitrost prenosa. Slika 15. prikazuje SPI komunikacijo v 4-žičnem načinu med gospodarjem in sužnjem. Vmesnik prav tako podpira 3,2,1 žični način.

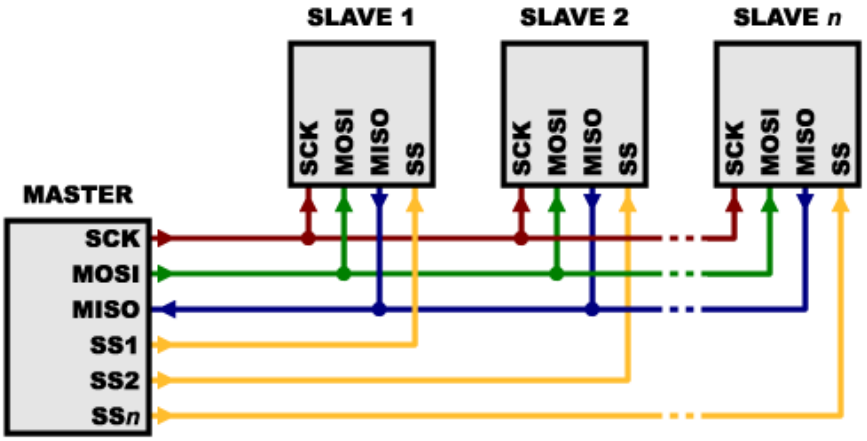


Slika 15: SPI komunikacija.

SPI komunikacija v polni razsežnosti uporablja štiri linije:

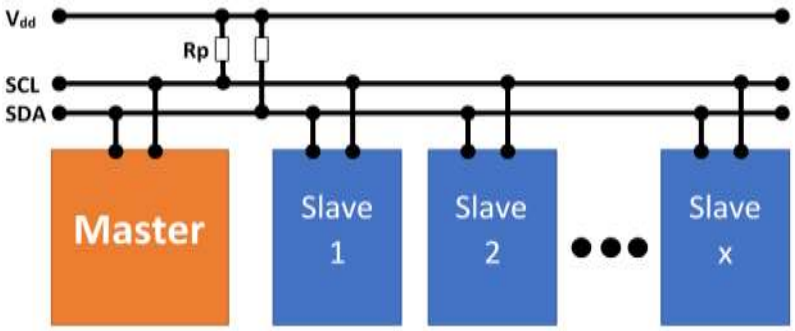
MOSI: ('Master Out Slave In')Povezava za pošiljanje podatkov gospodarja k sužnju.
MISO: ('Master In Slave Out') Povezava za prejemanje podatkov gospodarja od sužnja.
SCK: ('System clock') Generirana ura-hitrost komunikacije.
SS ali CS: ('Slave select' ali 'Chip select') Naslavljanje naprave iz strani gospodarja.

SPI komunikacija omogoča priključitev več naprav na isto SPI vodilo MOSI, MISO, SCK. Vsaka naprava ima le ločen SS/CS pin, slika 16. SPI komunikacija pošilja pakete po 8 bitov. Hitrost komunikacije je vezana na uro CPU-ja gospodarja ter hitrost ure sužnja.



Slika 16: Več naprav na SPI vodilu.

- **IIC (I^2C):** Je serijska komunikacija, ki omogoča polduppleksni način. Prav tako uporablja princip gospodar-suženj. Enako kot SPI je tudi IIC namenjena za komunikacijo med bližnjimi napravami. Hitrost IIC vodila je določena s tremi načini. Počasen način 100kbit/s, hiter način 400kbit/s in najvišja hitrost 3.4Mbit/s. Vmesnik za komunikacijo potrebuje le dve liniji. Ena linija je podatkovna-SDA in druga linija je generirana ura SCL iz strani gospodarja. Vodilo omogoča 10 ali 7 bitni način prenosa podatka. Naprava na IIC vodilu ima 7bitno adreso. Na začetki komunikacije gospodar naslavlja napravo s adreso potem se začne prejemanje in oddajanje podatkov. Vmesnik omogoča več naprav na istem vodilu, slika 17.



Slika 17: IIC komunikacijski vmesnik.

SPI komunikacija je hitrejša od I2C komunikacije ter fizično porabi več pinov, kot I2C vodilo. SPI komunikacija se pogosto uporablja za vodenje grafičnih prikazovalnikov za spominske kartice (SD-karitce) ter zajemanje slike iz optičnega čipa. I2C je pogosto uporabljen vmesnik, kjer ni strogo določenih časovnih omejitev.

Mikrokrmilniki pogosto vsebujejo več ločenih komunikacijskih vmesnikov. Za primer vzemimo mikrokrmilnik STM32F407, kjer imamo 6 USART vmesnikov, 3 SPI vmesnike in 3 IIC vmesnikov.

8.3. Napotki za nizko energetsko rabo krmilnika ter ekološki vidiki

Vsak krmilnik je možno konfigurirati tako, da je poraba čim manjša. Prvi pristop ekološkega snovanj zahteva izbiro krmilnika, glede na njegove karakteristike ter velikost čipa. Velikost čipa je direktno povezana z uporabljenim materialom čipa, kot so plastika ohišja, kovinski pini ter silicij. Uporabljen material na koncu življenjskega cikla vpliva tudi na recikliranje. Pri izbiri krmilnika ne bomo izbrali 144 pinski čip, pri tem pa uporabili le nekaj vhodnih in izhodnih pinov. Za ta namen je smiselno uporabiti manjšo različico iste družine. Drugi primer je uporaba visoko zmogljivih krmilnikov. Visoko zmogljivi krmilniki porabijo več energije, kot preprostejši krmilniki. Za vsako napravo ali aplikacijo je smiselno oceniti, kakšen krmilnik bomo uporabili. Na trgu se pojavljajo krmilniki, ki so namenjeni za nizko porabo in čim daljšo avtonomijo.

Kadar izberemo krmilnik je smiselno preučiti naslednje stvari:

- **Ura krmilnika:** Nastavitev ure krmilnika je tesno povezana s porabo energije in tako z avtonomijo sistema. Frekvenca je sorazmerno povezana z energijo. Pri vsakem krmilniku je možno nastaviti tak CPU-ju v določenih mejah, ki jih predpiše proizvajalec. Če naloge krmilnika nimajo visokih časovnih zahtev, lahko takt krmilnika znižamo do te mere, da zadostimo časovnim kriterijem opravil. Večina krmilnikov ima tudi programsko nastavljivo uro, kar pomeni, da tekom izvajanja zahtevnih nalog povišamo takt ure in tako zgotovimo hitrejše izvajanje. Kadar krmilnik opravlja preproste in časovno nezahtevne naloge uro znižamo do te mere, da opravila tečejo normalno.
- **Napajalna napetost:** Večina krmilnikov ima dopuščeno nastavljivo napajalno napetost v določenem območju. Napajalna napetost je prav tako sorazmerna s porabo energije. V večini primer je napajalna napetost povezana tudi z učinkovitostjo krmilnika in taktom ure CPU-ja. Pogosto se srečamo z omejitvijo, da pri nižji napetosti ne moremo nastaviti hitrejšega takta ure. Pri izbiri napajalne napetosti je nekako potrebno narediti kompromis med zmogljivostjo ter porabo energije.
- **Izklapljanje neuporabljenih modulov:** Mnogokrat se soočimo s snovanjem programske kode, kjer ne uporabljamo veliko perifernih naprav. Krmilniki omogočajo, da se periferen naprave izključijo, kar pomeni, da ne trošijo dodatne energije, ko niso v uporabi. Prav tako je možno uporabo perifernih naprav voditi tekom programske kode.

- **Optimalni dizajn kode:** Pri snovanju kode je potrebno paziti na izvajanje. Čim krajšo kodo spišemo tem manj inštrukcij krmilnik opravi, kar posledično pomeni, da krmilnik lahko dalj časa ostane v času pripravljenosti. Pri snovanju kode je potrebno paziti tudi na tip izbranih spremenljivk ter števila spremenljivk. Pri daljših programih se lahko hitro zgodi, da spremenljivko deklariramo in jo nato le redko uporabimo ali jo sploh ne uporabimo.
- **Način pripravljenosti:** V primeru, da krmilnik omogoča različen načine pripravljenosti je te smiselno uporabiti, še posebej če med izvajanjem nalog imamo daljše premore.

Viri:

[1] Günther Gridling, Bettina Weiss, 'Introduction to Microcontrollers', Vienn University of Technology press, 2007.

[2] http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/raspberry-pi/gpio-pin-electrical-specifications