Ecodesign-ul dispozitivelor electronice

UNIT 8: Sisteme de microcontroler partea 1

Autor: Andrej Sarjaš

8.1. Introducere în sistemele de microcontrolere 2
8.2. Structura microcontrolerului 4
8.2.1. Termeni generali 6
8.3. Structura microcontrolerului 7
8.3.1. Miezul procesorului 7
8.3.2. Memoria 9
8.4. Unități periferice 13
8.4.1. Intrări și ieșiri digitale 13
8.4.2. Intrări și ieșiri analogice 15
8.4.3. Subprograme de întrerupere 18
8.4.4. Contoare și cronometre 20
8.4.5. Interfețe de comunicare 22
8.5. Orientări pentru consumul redus de energie al microcontrolerelor și aspectele ecologice 26

Rezumatul capitolului:

–Sisteme în timp real
–Componente ale timpului real
–Proiectarea programelor de sisteme în timp real

Funded by the Erasmus+ Programme of the European Union

Competențe Ecoinnovation pentru designeri europeni, numărul proiectului: 562573-PPE-1-2015-1-SI-EPPKA2-SSA. Acest proiect a fost finanțat cu sprijinul din partea Uniunii Europene. Această publicație (comunicare) este responsabilitatea exclusivă a autorului său. Comisia nu este responsabilă pentru eventuala utilizare a informațiilor pe care le conține

8.1. Introducere în sistemele de microcontrolere

Când Intel a prezentat primul microcontroler 4004, a început epoca dezvoltării microcontrolerului. Structura modernă a microcontrolerului TMS1802 de la Texas Instruments a fost dezvoltată pentru a fi utilizată în calculatoare și până în 1971 a fost utilizată în multe sisteme în timp real, precum ceasuri, aparate de măsură, case de marcat etc. Cu dezvoltarea noii serii TMS100, care a început în 1974, microcontrolerul a conținut unități periferice care sunt componente de bază și în microcontrolerele moderne (RAM, ROM, I / O). TMS100 a fost apoi cunoscut sub porecla de microcomputer. Primul controler care a avut un real succes și a fost utilizat pe scară largă a fost Intel 8048, care avea tastatură integrată. Epoca aceea a continuat cu modelul Intel 8051, precum și cu Motorola 68HCxx.

Producția de microcontrolere de astăzi atinge miliarde de piese pe an cu numeroși producători diferiți. Sistemele electronice de astăzi sunt greu de imaginat fără un microcontroler, iar utilizarea lor este în continuă creștere. Iată câteva grupuri de dispozitive care utilizează sisteme de microcontroler:

- Aparate de uz casnic (microunde, mașini de spălat, mașini de cafea etc.).
- Telecomunicații (telefoane, smartphone-uri, modemuri, routere etc.).
- Electronice de consum (televizoare, playere de muzică și video, console de jocuri etc.)
- Întreprinderi/fabrici (gestionarea și controlul dispozitivelor și proceselor de producție).
- Industria de automobile (controlul motorului, sistemul de conducere de siguranță etc.).
- Tehnologie spațială.

Sistemele de microcontroler s-au dezvoltat foarte mult și reprezintă o mare substituție pentru sistemele analogice și circuitele. Odată cu includerea sistemelor de microcontrolere în proiectarea dispozitivului, putem reduce în mod semnificativ dimensiunea dispozitivului, crește eficiența, fiabilitatea și modernizarea. Din perspectiva ecologică, dispozitivele cu sistem de microcontroler sunt semnificativ mai economice, utilizarea materialelor este mai mică, iar reciclarea este mai ușoară. Microcontrolerele moderne conțin unități periferice de bază (RAM, FLASH, I / O, module de comunicație) și unități separate pe care le putem utiliza pentru a permite consumul redus de energie atunci când nu folosim sistemul sau este inactiv (modul de așteptare, modul de trezire etc.). Cu o bună înțelegere a structurii microcontrolerului și a conceptului de program în sistem, putem dezvolta un sistem sau un dispozitiv eficient care să fie mai ecologic. Proiectarea ecologică are un rol important în sistemele de microcontrolere, deoarece există multe opțiuni de asigurare a funcționării fiabile, a consumului redus și a utilizării reduse a materialelor.

Deci, ce este microcontrolerul? Care este diferența dintre microcontroler și procesor? De ce avem nevoie de un microcontroler și la ce ajută? De exemplu, să

examinăm mai atent un dispozitiv pentru încălzirea apei cu temperatură reglabilă. Principiul de funcționare al acestui dispozitiv este următorul:

- Măsurarea periodică a temperaturii.
- Gestionarea încălzitorului în funcție de temperatura curentă și stabilită (ON / OFF).
- Interfețe pentru gestionarea dispozitivelor (butoane, tastatură).
- Afișarea temperaturii.

În primul rând, procesul începe cu proiectarea plăcii cu circuite imprimate PCB și se folosește procesorul Z80 de la Zilog. Se adaugă, de asemenea, două unități de intrare-ieșire - PIO, interfață serială SIO, temporizator, SRAM, FLASH, EEPROM. Placa circuitului imprimat final este prezentată în imaginea 1.

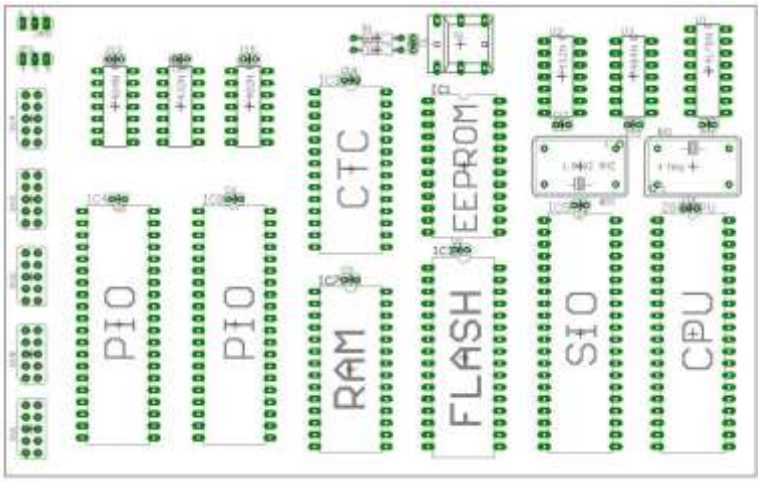

IMAGE 1: PRINTED CIRCUIT BOARD WITH PROCESSOR Z80.

Problema poate fi rezolvată cu ajutorul microcontrolerului ATmega16. Proiectul tipărit pentru același sistem este prezentat în imaginea 2.

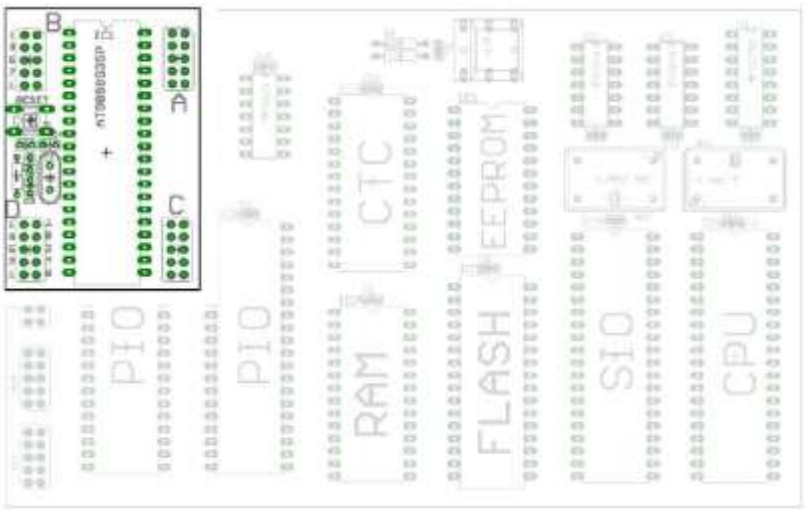

IMAGE 2: PRINTED MATTER WITH CONTROLLER ATMEGA16.

Din ambele aspecte tipărite, este vizibil faptul că materialul imprimat din imaginea 2 este mai mic decât cel din imaginea 1. Pe scurt, microcontrolerul este un microprocesor cu unități periferice într-un cip. De exemplu, am ales Z80 și ATmega16 deoarece microcontrolerul ATmega16 a fost dezvoltat pe baza Z80. Aceasta înseamnă că include Z80 și are unități periferice integrate, cum ar fi interfața serial și I / O, ADC, SRAM, ROM, FLASH și EEPROM. În acest caz, putem vedea și utilitatea și avantajele principale ale microcontrolerului.

În dezvoltarea dispozitivelor, avem multe opțiuni, de exemplu în alegerea categoriei producătorului și a microcontrolerului. Categoria microcontrolerului este de obicei asociată cu eficiența unității logice aritmetice (8, 16, 32, 64, 128 biți). Pentru aplicația dată, dimensiunea cipurilor trebuie determinată. Microcontrolerele din aceeași categorie sunt diferențiate în funcție de mărimea cipului sau de pini liberi. Aceasta conduce la carcase de 48, 100, 144 sau 176 de pini din aceeași familie. Dacă nu avem nevoie de multe interfețe externe, este bine să alegeți cipuri cu mai puțini pini, ceea ce duce la economii în prețul dispozitivului și suprafața imprimată.

8.2. Structura microcontrolerului

Dezvoltarea sistemelor de microcontrolere este extremă. În prezent, cele mai des folosite sunt sistemele cu 16-32 biți, cu viteză de la 10 MHz la 300 MHz. Structura internă a controlerului este aceeași. Imaginea 3 prezintă structura de bază a microcontrolerului.

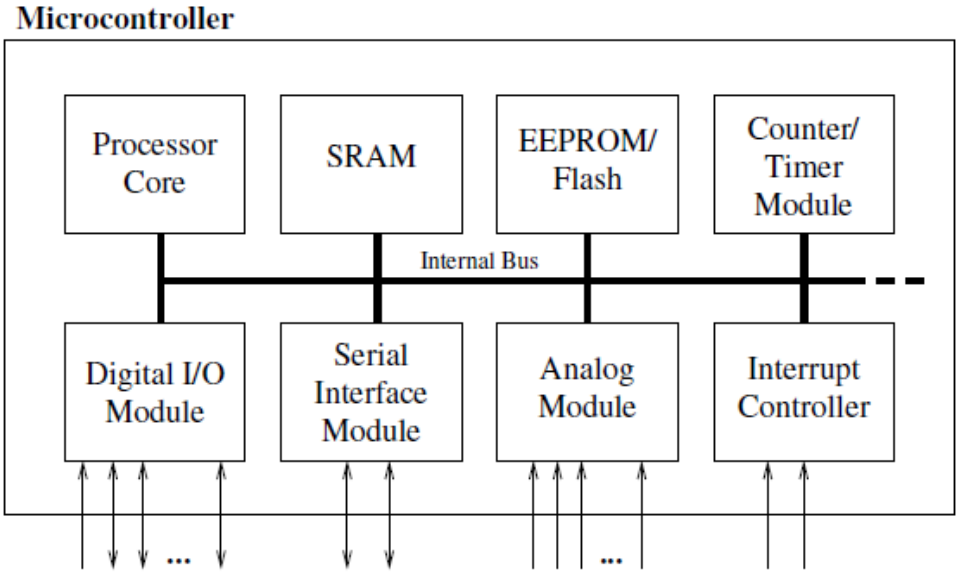

IMAGE 3: MICROCONTROLLER STRUCTURE

Toate unitățile periferice ale microcontrolerului, așa cum se vede în imaginea 3, sunt interconectate cu "magistrala internă" centrală. Modulele tipice care sunt comune în majoritatea microcontrolerelor sunt:

- **Procesor CPU:** procesor sau CPU (unitate centrală de procesare) este o unitate logică aritmetică care este capabilă de operații matematice (adunare și scădere) și conduce registre interne (registru de memorie, registru de programe, baterie etc.).
- **Memorie:** memoria este utilizată pentru salvarea datelor și variabilelor finale sau intermediare. Memoria microcontrolerului este împărțită în program și în partea de date. Printre microcontrolerele mai puternice cunoaștem DMA (acces direct la memorie), care permite unităților periferice să comunice direct cu memoria și să nu întrerupă operațiile procesorului.
- **Regulator de întrerupere:** întreruperile sunt esențiale pentru funcționarea eficientă a programului în timp real. Acestea întrerup funcțiile programului de bază în cazul evenimentelor din unitățile periferice. Întreruperile sunt, de asemenea, utile atunci când dorim să scădem consumul de energie în modul repaus.
- **Contoare și cronometre:** Majoritatea controlorilor au cel puțin două contoare care sunt destinate timpului de numărare, numărând evenimentele externe și intervalele. Multe cronometre au modul de codare de numărare și modularea PWM (modulația lățimii impulsurilor). Modularea PWM este adesea folosită ca DAC (conversie digitală la analogică). Rezoluția temporizatorului / contorului depinde de ceasul sistemului care este determinat de oscilatorul extern sau intern și de lungimea registratorului de numărare. Registratorul, mărimea este determinată de numărul de biți ai registratorului. Prin urmare, putem avea contoare de 8, 16 sau 32 de biți.
- **Intrări / ieșiri digitale**: adesea le vedem etichetate ca GPIO (intrare și ieșire cu scop general). Numărul de intrări și ieșiri este limitat de tipul controlerului și de versiunea acestuia.
- **Intrări / ieșiri analogice:** Printre controlerii mai mici, cele mai multe au convertoare analogice ADC (conversie analogică la cea digitală). Seria mai puternică de microcontrolere are mai multe ADC-uri, până la patru unități separate. Aceste unități sunt conectate la pinii externi prin intermediul multiplexorului. În practică, aceasta înseamnă că putem folosi mai multe intrări multiplexate analogice. Numărul de intrări analogice fizice multiplexate pe pinii controlerului este, prin urmare, mai mare decât numărul de unități ACD din controler. Unitățile ACD sunt diferențiate în funcție de rezoluția de conversie și știm de la convertoarele de la 6 la 12 biți. Rezoluția ACD este determinată de registrul de control al unității ADC care este configurat prin codul programului. În cea mai recentă serie de controlere, tendința este de a integra și unitățile DAC care sunt cheia pentru procesarea semnalelor de ieșire.

- **Interfețe de comunicare:** Pentru dispozitivele periferice din schemă, care nu fac parte din cip, se utilizează cel mai frecvent trei tipuri de standarde de comunicare. Acestea sunt conexiuni seriale USART, I2C și SPI. Toate aceste conexiuni au în comun că distanța dintre dispozitive este între câțiva centimetri până la maximum un metru.
- **Timer "Watchdog":** Timerul Watchdog este un cronometru special care declanșează independent funcționarea regulatorului. Acest cronometru asigură că, în caz de eroare de funcționare, controlerul este resetat.
- **Unitate de depanare:** La proiectarea unui program pentru controler este practică utilizarea interfeței de depanare. Această interfață poate accesa registrul controlerului intern în timp ce programul este în desfășurare. În acest fel, este mai ușor să căutați erori în codul programului și să rezolvați posibilele probleme în implementarea programului. Adesea se numește JTAG (grup de acțiune comun de testare).

Pentru a rezuma: microcontrolerul este un procesor limitat care a integrat unitățile periferice enumerate anterior. Cel mai mare avantaj al microcontrolerului este prețul. Putem economisi bani, diminua dimensiunea dispozitivului și adăugăm o anumită putere de calcul sistemului. Dezavantajul microcontrolerului este că nu poate atinge viteza circuitelor analogice. În sistemele cu un timp de reacție foarte scurt, trebuie să înlocuim o anumită parte a sistemului cu un circuit analogic. În majoritatea cazurilor, timpul de reacție nu este esențial pentru funcționarea dispozitivului. Noile microcontrolere ajung deja la timpul de reacție în intervalul de 10 microsecunde.

8.2.1. Termeni generali

Adesea întâlnim termeni diferiți care induc în eroare sau sunt utilizați incorect.

- **Microprocesor:** acesta este procesorul normal care poate fi găsit pe computerele personale. Comunicarea între dispozitivele periferice se face prin magistrala principală. Toate unitățile externe (memorie, USB, cronometre) sunt conectate la magistrala principală. Microprocesorul nu poate funcționa singur, deoarece are nevoie de dispozitive periferice, cum ar fi memoria, unitățile de intrare / ieșire etc. Microprocesorul nu este un microcontroler.
- **Microcontroler:** include toate unitățile periferice și poate funcționa ca atare. Acesta este conceput pentru managementul sistemelor. În funcție de microprocesor, microcontrolerul are unități periferice integrate propriu-zis în cip.
- **Controler de semnal mixt:** controlerul de semnal mixt poate procesa semnale analogice și digitale.
- **Sistem încorporat:** Zona importantă a microcontrolerului sunt sistemele încorporate. Termenul de sistem încorporat înseamnă că controlerul este integrat în sistem sau dispozitiv. Astfel de exemple sunt telefoanele, roboții, mașinile etc. Acestea sunt sisteme controlate de acești controlori.

• **Sistem în timp real:** înseamnă că sistemul este controlat și gestionat la ora definită. Reacția sistemului este executată în intervale de timp definite - în timp real. Intervalul de timp este adesea evaluat ca timp de sistem care este determinat de microcontroler sau procesor.
• **Procesor digital de semnal DSP:** Acestia sunt procesoare care sunt în afară de o unitate logică aritmetică de adunare și scădere capabilă să execute multiplicarea și împărțirea într-un singur ciclu. Astfel de procesoare sunt destinate procesării semnalelor digitale (transformare Fourier, calcul de corelare, autocorelații și convoluție). Producătorii de top ai controlorilor DSP sunt:
Texas Instruments, STMicroelectronics, Freescale, Microchip and Nxp Semiconductors, etc.

8.3. Structura microcontrolerului

În acest capitol vom analiza structura și funcționarea principalelor elemente de microcontroler.

8.3.1. Miezul procesorului

Miezul procesorului CPU este partea principală a fiecărui microcontroler. Imaginea 4 prezintă structura procesorului de bază.

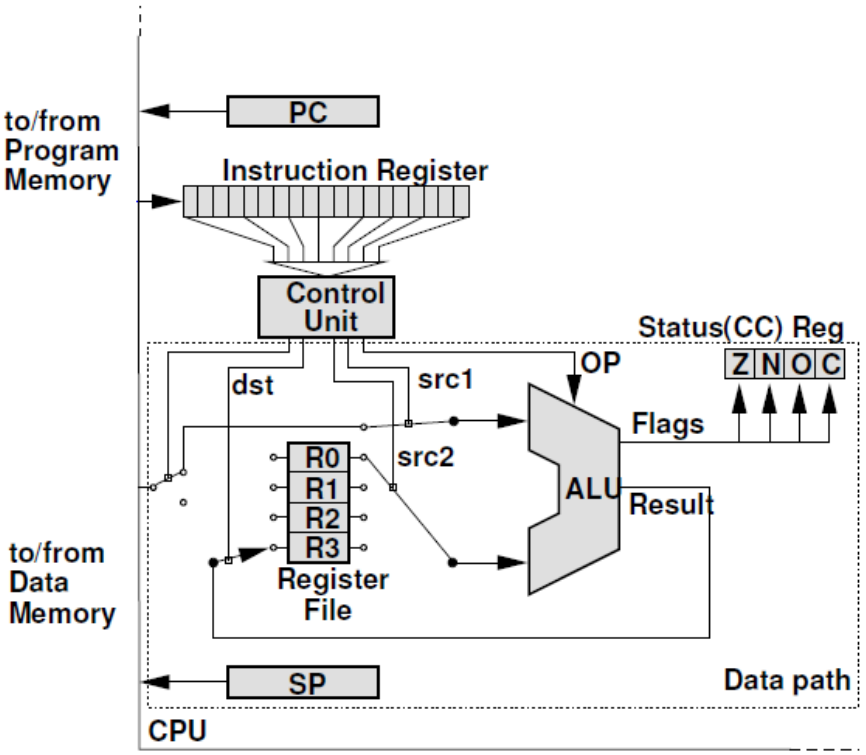

IMAGE 4: PROCESSOR STRUCTURE

Miezul CPU constă dintr-o cale de date care execută instrucțiuni de la unitatea de control care controlează calea de date. Miezul procesorului este o unitate logică aritmetică ALU care este capabilă să execute doar operații de bază de calcul, cum ar fi adăugarea, scăderea și completarea bițială. Practic, ALU ia două valori și le readuce la ieșire. ALU salvează, de asemenea, în registrul de stare. Etichetele din registrul de stare înseamnă că rezultatul Z este zero, N este negativ, operația O a provocat o depășire, operația C furnizează date ("curry").

Unitatea de control are sarcina de a executa instrucțiuni și de a determina instrucțiunile care vor fi executate la ora dată. Această unitate salvează, de asemenea, indicele de instrucțiuni din contorul de programe și conținutul acestor instrucțiuni în registrul de instrucțiuni. Instrucțiunea definește ce valoare din registru va fi trimisă către ALU și unde va fi salvată. În funcție de structura unității de comandă, cunoaștem două structuri:

- RISC: Structura calculatoarelor cu instrucțiuni reduse este mai simplă, deoarece executarea durează numai câteva cicluri de oră. Avantajele sistemelor RISC sunt că acestia utilizează o anumită lungime de microcod pentru adresare și instrucțiuni. Ca urmare, instrucțiunile sunt executate rapid, dar sunt destul de mici.
- CISC: Calculatorul complex de instrucțiuni este o structură care este gestionată de instrucțiuni complexe de microcoduri. Aceste instrucțiuni necesită mai multe cicluri de oră pentru execuție. Instrucțiunea are o

lungime variabilă și permite multe instrucțiuni eficiente în comparație cu structura RISC.

Alegerea structurii depinde de aplicația pentru care avem nevoie de controler. Dacă avem nevoie de instrucțiuni mai complexe, atunci trebuie să folosim structura CISC. Viteza cu care se execută instrucțiunile este conectată la ceasul principal din CPU. Miezul microcontrolerului este adesea bazat pe structura RISC, iar procesoarele computerelor se bazează în principal pe structura CISC.

În imaginea 4 putem vedea instrucțiunile și memoria de date ca două subiecte separate. Nu este întotdeauna aceeași situație, dar ambele pot folosi aceeași memorie. În funcție de faptul că memoria este separată sau comună, diferențiăm două tipuri de structură a procesorului:

- **structura Von Neumann**: în această structură, instrucțiunile și memoria de date sunt comune. Calea utilizată pentru a accesa memoria este, de asemenea, obișnuită. Din păcate, în această structură, aceasta poate duce la un conflict între instrucțiuni și date care pot provoca întârzieri nedorite ale sistemului. Acest dezavantaj este, în general, cunoscut sub numele de "strâmtoarea Von Neumann".
- **structura Harvard:** în această structură, memoria, precum și calea, sunt separate. Conflictul dintre date și instrucțiune nu este posibil, ceea ce îmbunătățește în consecință eficiența procesorului. Dezavantajul sistemului este că arhitectura are nevoie de mai multe componente, cum ar fi memoria dublă, cale dublă și o unitate care permite accesul simultan la memorie de date și instrucțiuni.

Viteza de execuție a sarcinii în CPU depinde de mai mulți factori. Este influențată în principal de structura procesorului, ceea ce înseamnă dacă sistemul este RISC sau CISC. Viteza de execuție a sarcinii depinde, de asemenea, de lungimea instrucțiunilor (8, 16, 32, 64 de biți). Să aruncăm o privire la instrucțiunea de 32 de biți, care este executată doar într-un singur ciclu cu procesor de 32 de biți, care este mai rapid decât dacă instrucțiunea ar fi executată pe un procesor pe 8 biți. Procesorul de 8 biți are nevoie de patru cicluri pentru executarea instrucțiunilor. În cele din urmă, viteza de execuție a sarcinii depinde de ceasul principal (cristal extern sau intern), ceea ce înseamnă că instrucțiunile sunt executate mai rapid cu ceasul de 160 MHz decât cu ceasul de 10 MHz.

8.3.2. Memoria

În capitolul anterior, am prezentat structura procesoarelor care au nevoie de memorie pentru funcționarea lor. În funcție de sarcina și structura memoriei, cunoaștem trei tipuri de memorii:

- **Registrul:** Registrul memoriei este o memorie relativ mică încorporată în CPU. CPU-ul are nevoie de memorie pentru a înregistra starea procesorului și setările unităților periferice, cum ar fi ADC, DAY, I2C etc. Această

memorie este numită și memorie pe termen scurt deoarece toate valorile din memorie sunt pierdute atunci când sursa de încărcare este deconectată.

- **Memorie de date:** În memoria de date, putem salva pe termen lung, și de obicei este adăugată la CPU ca unitate periferică. Memoria de date este semnificativ mai mare decât registrul.
- **Memorie de instrucțiuni:** Este relativ mare, la fel ca memoria de date și, prin urmare, este implementată pe lângă CPU ca dispozitiv periferic. În structura Von Neuman, instrucțiunile și memoria de date sunt un dispozitiv periferic cu o cale comună. În sistemele de microcontrolere, memoria este implementată în memoria cu un singur cip, dar este împărțită în sectoare.

Anterior au fost menționate cele mai comune tipuri de memorie utilizate pentru memorie CPU. Există și alte tipuri de memorie și registre, cum ar fi registrele de conducte, memoria ascunsă și diferite amortizoare. În funcție de structura microcontrolerului, dimensiunea memoriei este integrată în cip și este fixă. Memoria nu poate fi adăugată sau mărită. Din acest motiv, microcontrolerele sunt destinate doar executării unor sarcini mai simple.

Până acum, am tratat memoria ca o unitate care face anumite sarcini ale procesorului. Din punct de vedere al programării, memoria poate fi caracterizată ca memorie nevolatilă și volatilă. Memoria volatilă poate fi clasificată ca dinamică sau statică. Pentru memoria nevolatilă, folosim abrevierile în funcție de structura și tipul de funcționare: ROM, EPROM, EEPROM, FLASH. Clasificarea memoriei este prezentată în imaginea 5.

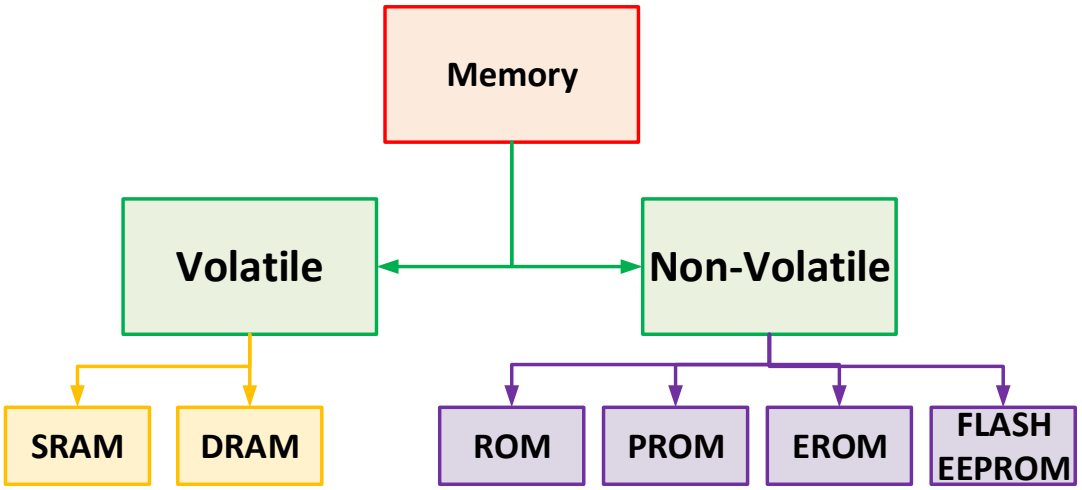

IMAGE 5: MEMORY TYPE

Memoria volatilă este tipul de memorie care salvează datele până când este deconectat sau resetat sistemul. Deci, de ce nu folosim memoria nevolatilă? Răspunsul este simplu. Memoria nevolatilă este mai lentă decât memoria volatilă și necesită mai mult timp operațional pentru salvare, ștergere și citire. Dacă facem o comparație dificilă:

citirea pe memorie volatilă este măsurată în nanosecunde și citirea pe memorie nevolatilă este măsurată în milisecunde.

• **RAM-SRAM static:** Word RAM provine din memoria de acces aleatoriu, ceea ce înseamnă că putem accesa locațiile de memorie aleatoriu. Fiecare locație este accesibilă cu adresa de memorie. Contrar RAM, știm registrele de deplasare unde datele pot fi scrise / citite doar secvențial. O astfel de memorie este denumită de regiștrii de deplasare tip FIFO (primul în primul out), FILO, LIFO (ultima în prima ieșire), LILO etc. Memoria SRAM pentru salvare folosește celulele D-flip-flop. D-flip-flop-ul este compus din cel puțin șase tranzistoare. Fiecare celulă D-flip-flop poate salva un bit (0 sau 1) și fiecare celulă are adresa. Imaginea 6 arată memoria matrice pe 16 biți cu jantele D. A0,1 în A2,3 înseamnă rândul / coloana.

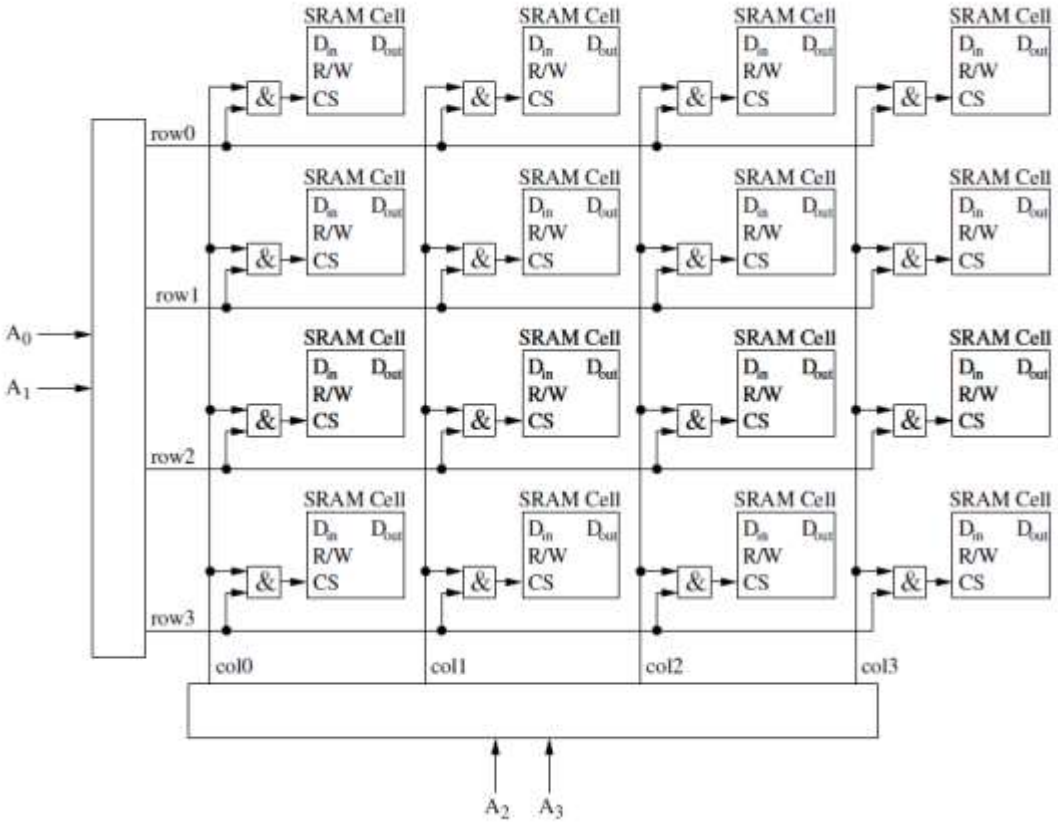

IMAGE 6: 16-BIT RAM MEMORY

• **Dynamic RAM - DRAM:** În comparație cu SRAM, RAM dinamică atinge capacități semnificativ mai mari. Acum știm că SRAM are nevoie de cel puțin șase tranzistoare pentru a salva 1 biți. Pentru capacitatea mai mare de SRAM, avem nevoie de mai multe celule care cresc semnificativ dimensiunea fizică a chip-ului și prețul. Dacă memoria poate fi redusă, putem folosi o modalitate simplă de a salva valoarea biților. DRAM salvează datele în stocarea încărcării electrice (condensator). În acest fel, numărul

de elemente pentru salvarea unui bit este redus și, prin urmare, capacitatea de memorie este mărită la o dimensiune semnificativ mai mică a chipului. Dacă vrem să salvăm valoarea logică în memoria DRAM, trebuie să încărcăm condensatorul la o anumită locație. Utilizează adrese pentru adresarea fiecărei celule, la fel ca SRAM și DRAM. Datorită structurii mai simple și a prețului mai mic, memoriile DRAM au unele dezavantaje în comparație cu memoria SRAM. DRAM este, de exemplu, mai lent. Pentru întreținere, este necesară reîmprospătarea memoriei deoarece condensatorul este epuizat în timp.

Contrar SRAM și DRAM, folosim și memorie nevolatilă. Aceasta este folosită atunci când dorim să salvăm date pentru o perioadă mai lungă de timp chiar și atunci când sistemul nu este conectat la sursa de alimentare.

- **ROM:** ROM-ul este memorie numai pentru citire. Numai pentru citire înseamnă că nu putem salva niciun fel de date. ROM-ul conține, de obicei, date salvate de producător și sunt cheia funcționării dispozitivului. În ROM se salvează BIOS-ul pentru computerele personale sau se înregistrează valori pentru setările microcontrolerului.
- **PROM:** ROM-ul este utilizat numai în producția de masă deoarece producția sa pentru cantități mai mici sau pentru cercetarea pilot este prea scumpă. PROM (memorie programabilă numai pentru citire) este o alternativă pentru ROM. Acesta poate fi programat o singură dată deoarece are o siguranță care împiedică scrierea repetată. PROM este, de asemenea, cunoscut sub numele de OTP (memorie programabilă o singură dată). Nu este potrivit pentru dezvoltare când conținutul de memorie și programul se schimbă adesea.
- **EPROM:** EPROM este o memorie programabilă pentru citire care poate fi șters. Programarea EPROM necesită o procedură complexă. Prin urmare, de obicei nu poate fi șters de microcontroler. Valoarea EPROM este salvată în FET (tranzistori cu efect de câmp) care este gestionată de pin-poarta. Înalta tensiune în pin-poarta închide tranzistorul. Când aceste porți sunt închise, ele rămân închise până când nu mai sunt sub tensiune. În acest fel putem salva valori digitale 1 sau 0. Datorită problemelor din FET, porțile se pot deschide în timp. Producătorii precizează de obicei cât de mult pot fi salvate datele în EPROM. Această perioadă este de obicei zece ani - fiecare EPROM are o fereastră care permite ștergerea. Aceasta se execută prin fereastră cu lumină ultravioletă. Lampa UV deschide FET, ceea ce înseamnă că memoria este ștearsă. Ștergerea EPROM durează aproximativ 30-40 de minute.
- **EEPROM:** EEPROM este o memorie ce se poate șterge și ROM programabil din punct de vedere electric, ceea ce înseamnă că este o memorie programabilă electronic. În esență, EEPROM funcționează la fel ca EPROM; singura diferență este că pentru ștergerea memoriei nu avem nevoie de tensiuni externe speciale și de lumină UV. Tensiunea înaltă pentru trezirea FET este încorporată în interiorul cipului și este cunoscută sub denumirea

de pompă de încărcare. În general, EEPROM are, ca EPROM, un ciclu de viață care este adesea limitat la 100.000 de cicluri de ștergere.

- **FLASH:** FLASH este o versiune limitată a memoriei EEPROM și funcționează la fel ca EEPROM. Diferența dintre FLASH și EEPROM este că nu putem șterge fiecare celulă separat, ci doar memoria completă a unui anumit sector. Motivul pentru implementarea memoriei FLASH este prețul ridicat al memoriei EEPROM. FLASH are, de asemenea, un număr limitat de semnale, la fel ca EEPROM.
- **NVRAM:** NVRAM (RAM nevolatil) este o combinație de memorie volatilă și nevolatilă. Această memorie are același principiu de funcționare ca SRAM, dar a adăugat o baterie de alimentare. De asemenea, cunoaștem o versiune a memoriei, în care memoria EEPROM și SRAM sunt îmbinate în același chip.

Când operăm cu date din memorie, este important să recunoaștem tipul de scriere. Pentru scrierea datelor, știm două abordări:

• **Big Endian:** Este un tip de scriere sau citire a datelor în cazul în care octeții mai mari sunt salvați la începutul locului de memorie. De exemplu, dacă salvăm datele "0x7799" pe adresa de memorie "0x00" și "0x01". În Big Endian, valoarea '0x77' este salvată la adresa '0x00' iar valoarea '0x99' este salvată la adresa '0x01'.

• **Little Endian:** Este un tip de scriere în cazul în care octeții înalți sunt salvați în locații superioare. Dacă vom continua cu exemplul anterior: din valoarea "0x7799", partea "0x77" este salvată la locația "0x01" și "0x99" este salvată la "0x00".

8.4. Unități periferice

În capitolele anterioare, am prezentat diferența dintre microcontroler și procesor. În acest capitol vom prezenta unități periferice, care cel mai frecvent sunt integrate într-un cip al unui microcontroler modern.

8.4.1. Intrări și ieșiri digitale

Intrările și ieșirile digitale - I / O sunt utilizate pentru controlul și gestionarea dispozitivelor externe și sunt principalele unități din microcontroler. În consecință, fiecare microcontroler are cel puțin 1 până la 2 pini I / O digitale care pot fi conectați la dispozitive externe. În general, microcontrolerele au mai mult de 32 intrări / ieșiri care pot fi utilizate în diverse scopuri. La conectarea intrărilor / ieșirilor este necesar să fiți atenți la nivelurile de tensiune admise care sunt permise de un anumit microcontroler. Cele mai comune tensiuni sunt de 5 și 3.3 volți.

Intrările și ieșirile digitale sunt adesea grupate în porturi. Fiecare port poate conține 8, 16 sau 32 de intrări / ieșiri. Gruparea unui număr de pini în porturile individuale depinde de producător și de lungimea registrelor de microcontrolere de control. Cel mai adesea toți pinii I / O sunt în ambele sensuri, adică pot fi utilizați ca intrări sau ieșiri. Cei mai mulți pini de I / O au, de asemenea, funcționalități suplimentare, cum ar fi contoare, întreruperi externe, o interfață de comunicație și conversie analog-digital ADC sau digital-analog DAC. Toate funcționalitățile portului I / O pot fi configurate în program prin intermediul registrelor de control ale fiecărui port.

Intrările / ieșirile digitale sunt numite în acest fel, deoarece potențialul de tensiune din intrarea pinului este convertit la o valoare logică "1" sau "0". Nivelurile de tensiune ale valorilor logice sunt prezentate în imaginea 7. În general, putem spune că valoarea zero a logicului este potențial de tensiune sub 0-1V. Logic "1" este potențial de tensiune mai mare de 2.4V-Vcc. Vcc este tensiunea de alimentare a microcontrolerului. Voltajul variază în funcție de producători și de tipul pinilor I / O și, de asemenea, dacă este utilizat ca intrare sau ieșire [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]

IMAGE 7: VOLTAGE POTENTIALS I/O PINS BY DIFFERENT MANUFACTURERS

Microcontrolerele folosesc cel mai adesea logica pozitivă, ceea ce înseamnă că logicul "1" este potențial mai mare de 2,4V, iar logicul "0" este sub 1V. În logica negativă logică "1" este sub 1V și logică "0" este de peste 2.4V. Sistemele mixte sunt adesea folosite atunci când dorim să conectăm microcontrolere cu tensiuni diferite de alimentare. De exemplu, avem un sistem alimentat de 5V și 3.3V. Deseori, producătorii de microcontrolere cu tensiune de alimentare mai mică, producI / O care sunt tolerante

la 5V. Caracteristicile pinilor și ale caracteristicilor electrice ale pinilor sunt descrise separat în fișele tehnice pentru fiecare grup și producător.

De multe ori putem întâlni termenii "trage-în sus", "trage-în jos" sau "variabil". Termenul de tragere înseamnă că determinăm potențialul de tensiune implicit la pinii I / O atunci când nu avem potențial extern pe pin. În modul "trage-în sus", potențialul implicit este tensiunea de alimentare, iar în "trage-în jos" potențialul implicit este GND. Termenul "variabil" înseamnă că lăsăm pinul plutitor (nu avem potențial implicit). În acest caz, pinul este supus unor efecte externe, care pot provoca întreruperi nedorite în sistem dacă pinul este utilizat discret ca intrare sau ieșire. Modul "variabil" este adesea folosit în conversia ADC sau DAC. Imaginea 8 prezintă modurile de "trage-în sus", "trage-în jos" sau "variabil".

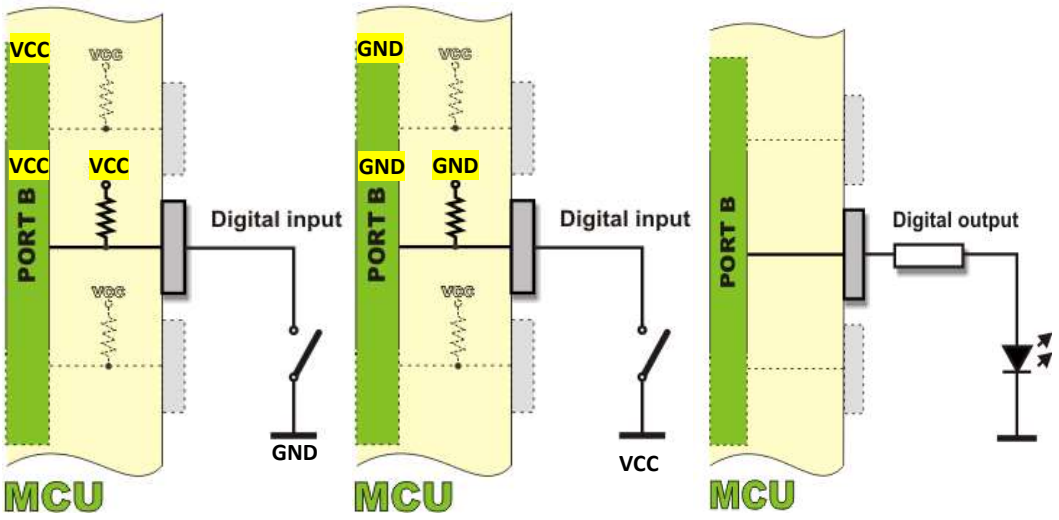

IMAGE 8: I/O PIN MODES; A) PULL-UP, B) PULL-DOWN, C) FLOATING.

8.4.2. Intrări și ieșiri analogice

În comparație cu intrările și ieșirile digitale, este posibilă citirea sau procesarea diferitelor niveluri de tensiune cu intrări / ieșiri analogice. Citirea valorilor analogice este legată de unitatea periferică ADC din controler. Datele de bază ale convertorului ADC sunt rezoluția, viteza de conversie și tipul de conversie. Rezoluția ADC este dată cu numărul de biți. Din numărul de biți, putem determina câte niveluri de tensiune sunt împărțite în domeniul de tensiune al ADC. Domeniul de tensiune prestabilit al ADC este de obicei tensiunea de alimentare a ADC. Majoritatea microcontrolerelor au un domeniu de tensiune reglabil care poate proveni de la un circuit de referință extern. Microcontrolerele au, de asemenea, rezoluție ADC reglabilă. Aceasta înseamnă că putem alege între rezoluția convertorului de 6, 8, 10 sau 12 biți. Să aruncăm o privire la ADC de 12 biți și la tensiunea de alimentare de 3.3V. Cea mai mică tensiune care poate fi măsurată prin ADC este dată de următoarea formulă:

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

unde V_{REF} este domeniul de tensiune al ADC, N este numărul de biți și V_{RES} este rezoluția tensiunii convertorului. Pentru exemplul dat (12 biți, 3.3V), rezoluția tensiunii convertorului este V_{RES} = 0.805mV. Multe convertoare permit setarea celei mai ridicate și cele mai scăzute referințe de tensiune.

Aceasta înseamnă că rezoluția completă a convertorului poate fi utilizată între două potențiale. De exemplu, aruncați o privire la măsurătorile potențialelor de tensiune între 1.5V și 2.2V. cu setarea V_{REF} - = 1.5V și V_{REF} + = 2.2V putem distribui rezoluția pe 12 biți numai între 0.7V (V_{REF} + - V_{REF} -). Rezoluția convertorului între potențialele 1.5-2.2.V este acum $V_{RES} = \frac{V_{REF+} - V_{REF-}}{2^N - 1} = \frac{2.2 - 1.5}{2^{12} - 1} = 0.171mV$.

Următoarele date cheie sunt viteza de conversie care este dată de obicei în numărul de probe pe secundă. Viteza de conversie în microcontrolere variază, de obicei, între 100kpps și mai multe 3MPS, în funcție de producător și de categoria controlorilor. 3Msps înseamnă că ADC creează 3 milioane de probe într-o secundă. Viteza eșantioanelor este un element cheie în procesarea semnalelor digitale.

Captarea semnalului ADC este împărțită în trei faze. La început, folosim suportul de semnal (eșantionare și reținere). Suportul de semnal deține semnalul atâta timp cât are loc conversia. În imaginea 9, suportul de semnal este prezentat ca întrerupător și condensator în care salvăm tensiunea semnalului eșantionat.

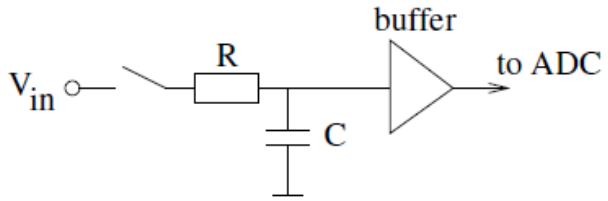

IMAGE 9: SAMPLE AND HOLD SCHEME

A doua fază de conversie este cuantificarea. Acesta este un proces în care divizăm nivelurile de tensiune în intervalele de măsurare. Un cuantic ADC prezintă rezoluția ADC care a fost prezentată cu formula de mai sus. Procesul de cuantificare este rotunjirea valorii analoge reale; aceasta înseamnă că introducem eroarea de măsurare. Această eroare se numește eroare de cuantificare a convertorului ADC. Ultimul pas în conversie este codarea în care fiecare nivel de cuantificare a atribuit o valoare binară. În imaginea 10 este prezentat procesul de cuantificare pentru 3 biți DC cu eșantioane de timp τs.

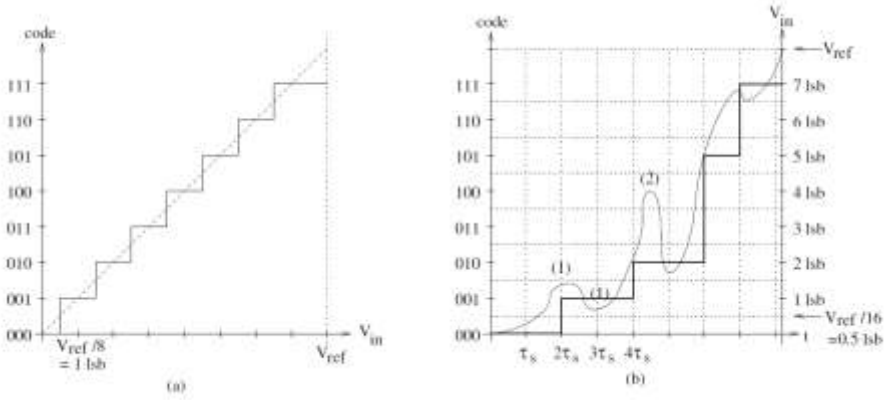

IMAGE 10: QUANTIFICATION PROCESS: A) VOLTAGE LEVELS OF ADC *0-V_{REF}*, B) SIGNAL QUANTIFICATION

Într-un microcontroler, unitatea ADC este conectată la pinii controlerului I / O. Acest lucru ne permite să citim mai multe semnale analogice cu doar unul sau două ADC-uri. Intrările în ADC sunt multiplexate. Pinii multiplexați sunt, de asemenea, numiți canale ADC. Microcontrolerele avansate au 3-4 ADC-uri separate, în timp ce cele mai simple au doar 1 ADC. Imaginea 11 prezintă multiplexarea pinilor de intrare în ADC.

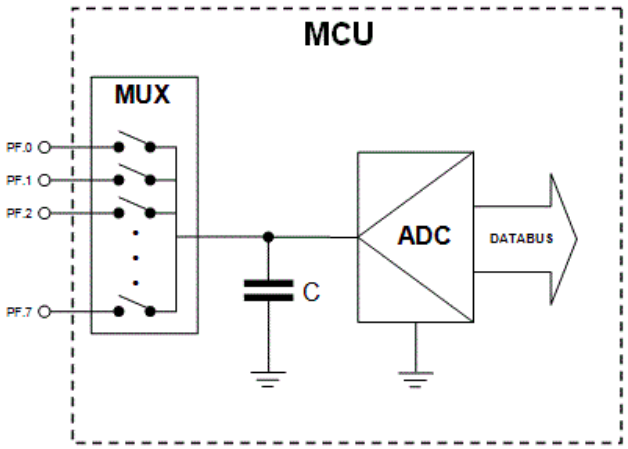

IMAGE 11: MULTIPLEXED INPUT PINS IN ADC

Funcția convertorului digital-analog DAC este de a converti valoarea binară la semnalul analogic. DAC este cel mai adesea folosit pentru procesarea oricăror semnale de ieșire de la controler. DAC are caracteristici similare cu ADC. Rezoluția DAC este definită prin rezoluție (numărul de biți) și eșantioane pe viteză secundă. Nivelul de tensiune al semnalului de ieșire este determinat de referința de tensiune, care este în mod prestabilit setată la tensiunea de alimentare a controlerului. Unitățile DAC se găsesc numai în cele mai avansate controale, iar numărul de unități este, în general, mai mic decât numărul de ADC-uri. Dacă controlerul nu are unitate DAC, semnalul analogic poate fi procesat cu modul de lățime a impulsului (PWM). Semnalul PWM este generat cu o frecvență fixă și un ciclu de funcționare reglabil. Cu ciclul de funcționare, generăm

potențialul de tensiune dorit pe pinul de ieșire. Valoarea medie a unei perioade de semnal PWM prezintă valoarea analogică a ieșirii.

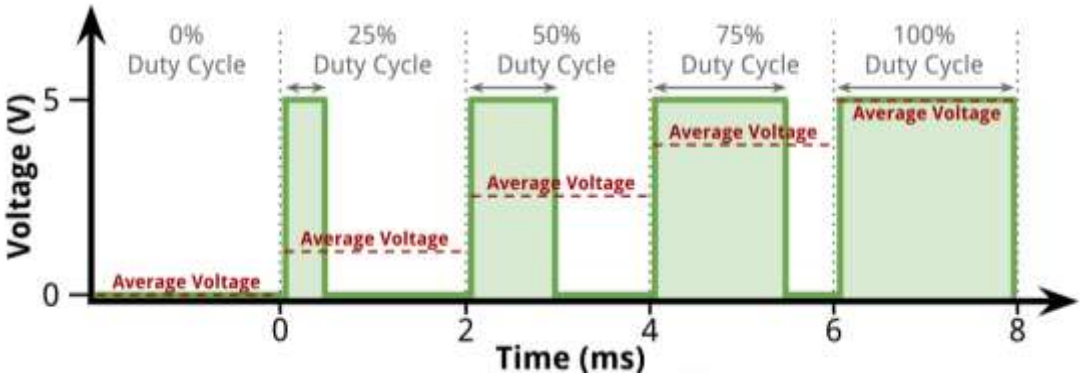

IMAGE 12: PULSE WIDTH MODULATION

8.4.3 Subprograme de întrerupere

Programele care sunt executate pe sistemele de microcontrolere trebuie să reacționeze la evenimentele date. Evenimentele diferă în funcție de durată, indiferent dacă se repetă și de complexitate. Controlorii trebuie să reacționeze la fiecare schimbare potențială a pinului de intrare, precum și să primească sau să transmită date prin interfețe de comunicație.

Subprogramele de întrerupere se referă la faptul că controlerul întrerupe executarea programului principal și este dedicat funcției de întrerupere. Dacă folosim multiple întreruperi, trebuie stabilite priorități. Prioritățile determină ce subprogram are avantajul în execuție. La proiectarea unui program, este bine să clasificați subprogramele de întrerupere după priorități și modul în care acestea vor fi executate. Dacă folosim subprograme, care trebuie executate la un anumit moment, este logic să setăm o prioritate ridicată. Mai jos vom prezenta câteva dintre cele mai comune subprograme de întrerupere:

- **Întrerupere de timp:** Întreruperea temporală este legată de cronometru și este executată în conformitate cu contorul temporizatorului. Intervalul de timp este executat la ora exactă. Întreruperea poate fi periodică sau controlată prin registrul de stare al întreruperii. Controlul întreruperii înseamnă că îl permitem atunci când este necesar. Intervalele de timp sunt adesea folosite pentru măsurarea timpului și sincronizarea diferitelor sarcini periodice.
- **Întreruperea contorului:** Întreruperea contorului este în esență aceeași cu întreruperea timpului, dar nu este strict legată de timp. Întreruperea este executată atunci când contorul atinge o anumită valoare. Numărătoarea contorului este în general declanșată de dispozitive externe prin intrări discrete. Întreruperea contorului este de obicei folosită pentru contorizarea impulsurilor.

- **Întrerupere externă:** Întreruperea externă este declanșată când un potențial de tensiune este modificat la intrare. Întreruperea externă citește doar valorile logice 0 sau 1. Întreruperea detectează frontul pozitiv sau negativ al intrării discrete.
- **Primirea întreruperii:** Această întrerupere este utilizată pentru sincronizarea cu expeditorul. Aceasta înseamnă că subprogramul detectează când datele sunt transmise. Subprogramele de întrerupere sunt adesea folosite în recepția asincronă a datelor. Recepționarea datelor asincrone înseamnă că transmiterea datelor nu este periodică, ci la întâmplare. Întreruperea garantează că nu rataţi primirea datelor.
- **Transmisia întreruperii:** Întreruperea la transmisia de date permite ca acel pachet de date să fie trimis în momentul dat. Întreruperea permite ca controlerul să nu efectueze alte sarcini decât transmiterea de date.
- **Întreruperea ADC:** Întreruperea ADC este timpul necesar ADC pentru citirea valorii analogice. În timpul întreruperii, controlerul așteaptă până la conversia ADC și apoi continuă cu alte sarcini. Întreruperea ADC este utilă atunci când avem nevoie de date din ADC pentru sarcinile următoare.
- **Întreruperea CAD:** La fel ca în cazul întreruperii ADC, întreruperea CAD are un rol similar. DAC declanșează întreruperea pentru o perioadă scurtă necesară conversiei valorii binare la ieșirea analogică.
- **Întreruperea modului repaus:** Întreruperea modului repaus este utilă atunci când controlerul nu execută operațiile în mod constant. Apoi putem seta controlerul să intre în modul sleep. În acest mod, are un consum redus de energie. Se folosește întreruperea modului repaus, astfel încât controlerul utilizează majoritatea energiei pentru monitorizarea unui anumit eveniment. Toate celelalte dispozitive periferice sunt oprite. La evenimentul setat, întreruperea trezește controlerul care funcționează cu toate unitățile periferice necesare.

Subprogramele de întrerupere sunt o activitate de cod de program care este executată la întrerupere. În aceast subprogram, este important să ștergem indicația de întrerupere. Indicația de întrerupere este un bit în registrul de stare care este setat la valoarea "1" atunci când are loc întreruperea. Dacă dorim ca întreruperea să fie executată din nou, indicația trebuie să fie resetată la "0". Când scriem programul pentru subprogramele de întrerupere, trebuie să fim foarte precauți, deoarece codul nu funcționează mai mult decât se poate repeta întreruperea. Acest lucru este deosebit de important la întreruperile care sunt executate periodic și rapid.

Iată câțiva factori care influențează atunci când este logic să folosiți subprograme de întrerupere:

- Sarcinile trebuie executate în mod aleatoriu.
- Intervale lungi între două sarcini.
- Schimbarea condiției este importantă.
- Monitorizarea impulsurilor scurte.

• Pentru funcționarea completă a controlerului, trebuie executate doar câteva sarcini.
• Evenimentele sunt generate de ambele dispozitive. Nu există semnale de vârf și nici un efect mecanic de bouncing.

Câteva linii directoare când nu avem nevoie de subprograme de întrerupere și când sarcinile sunt executate în programul principal:

- Operatorul este om.
- Executarea evenimentului nu este critică în timp.
- Impulsurile de intrare sunt lungi.
- Semnalul este acoperit de zgomot.
- Programul principal nu este lung.

Atunci când proiectați un program pentru microcontroler, este important să analizați comprehensiv execuția programului. Trebuie să se studieze ceea ce este executat în subprogramele de întrerupere și ce va fi executat în programul principal. Atunci când se utilizează mai multe întreruperi, trebuie să fim atenți dacă executarea întreruperii necesită utilizarea suplimentară a memoriei RAM. Atunci când sunt declanșate mai multe întreruperi, controlerul poate rămâne fără memorie și putem pierde anumite informații din subprogramele de întrerupere. Corecția unor probleme similare este foarte dificilă și consumatoare de timp.

8.4.4. Contoare și cronometre

Contoarele și temporizatoarele sunt unități periferice cheie ale fiecărui microcontroler care sunt adesea conectate la alte unități. Microcontrolerele au integrate mai multe contoare care diferă în funcție de numărul de biți (8, 16, 32 de biți). Timerele sunt utilizate pentru sarcini diferite, cum ar fi întârzierea, măsurarea timpului, măsurarea frecvenței. Cea mai folosită utilizare a cronometrelor este folosirea contorului. Timerele pot genera diferite evenimente, întreruperi sau semnal PWM modulat de proces.

Fiecare cronometru este un contor. Contorul temporizatorului poate număra în sus sau în jos. Modul de numărare poate fi configurat în interiorul registrului de control. Condiția contorului poate fi citită în registrul de date al contorului. Fiecare contor are propriul control, statut și registru de date. Dimensiunea sa este determinată de lungimea bitului. Lungimea de biți este cel mai mare număr posibil de numărători 2N-1, unde N este numărul de biți. De exemplu, contorul pe 8 biți poate număra între 0 și 255, unde contorul pe 16 biți poate conta de la 0 la 65535. Dacă contorul este conectat la ceas, obținem un cronometru. Cronometrul declanșează numărarea în funcție de modul de introducere (ceas). Numărătoarea cronometrului poate fi declanșată de marginea pozitivă sau negativă a tactului ceasului. Ceasul utilizat în timer poate fi extern sau intern. În cazul tactului intern, folosim ceasul procesorului, deși vine de la un cristal extern. Acest mod de utilizare a ceasului este modul sincron. Tactul ceasului extern înseamnă că oferim un nou ceas special pentru temporizator. Ceasul temporizat extern nu este legat de tactul CPU cunoscut ca modul asincron. Ceasul extern este adesea

folosit pentru calendar (ora, zilele, luna etc.). Pentru calendar, tactul ceasului este exact 32.768kHz. Temporizatorul pentru calendar este cunoscut ca numărător RTC (RTC este ceasul în timp real). Temporizatorul poate fi configurat prin mai multe moduri de numărare și interval de timp pentru numărare. Cresterea timpului cronometrului este timpul creșterii unui contor. Creșterea timpului și durata de timp pot fi setate în registrele temporizatorului.

Parametrii principali ai cronometrului:

- **Ceasul:** Ceasul temporizat este determinat de viteza căii periferice pentru cronometrul dat. Viteza căii periferice este determinată de setările de sistem ale controlerului. Unii controleri simpli nu au căi periferice separate pentru unitățile individuale, iar ceasul este același pentru toate unitățile periferice. De asemenea, trebuie să determinăm sursa ceasului care este modul asincron sau sincron.
- **Factor de scalare (prescaler):** Factorul de scalare determină împărțirea ceasului temporizat. Cu acest factor, determinăm creșterea temporizării și durata de numărare atunci când numărăm până la sfârșitul înregistrării datelor de contoare. Creșterea temporizării este o rezoluție de timp care determină cât de precis măsuram timpul. Valoarea factorului de scalare este salvată în registrul PSC.
- **Perioada:** Perioada: Perioada de timp determină intervalul de timp de numărare. Aceasta înseamnă că cea mai lungă perioadă este egală cu lungimea registrului de date de contor. Datele lungimii contorului sunt determinate de un număr de biți din cronometru. Perioada de timp este salvată în ARR (registrul de reîncărcare automată). Perioada de contracarare determină numărul de unități numărate în lungimea întregului registru ARR. Aceasta înseamnă că contorul nu numără doar până la lungimea maximă a ARR, dar și până la valoarea perioadei care este stabilită în program.

Contoarele au încă două moduri de funcționare. Acestea sunt cunoscute ca și captură de intrare și captură de ieșire. Captura de intrare este utilizată atunci când dorim să numărăm anumite evenimente pe pinii de intrare ai controlerului. Fiecare schimbare (eveniment) a valorilor logice "0" sau "1" pe un pin specific de intrare este transferată de la contor la registrul de captură de intrare. Acest registru poate fi citit în codul programului. Modul de captare a intrărilor este adesea folosit pentru citirea codificatoarelor diferite. Modul de comparație a ieșirii este similar cu captura de intrare, dar aici monitorizăm pinii de ieșire. Modificările de pe pinii de ieșire măresc sau scad contorul. Modul de comparație a ieșirilor poate declanșa și întreruperea atunci când contorul a atins numărul incrementului setat.

În afară de contoarele foarte utile, microcontrolerele conțin și contoare speciale care nu sunt destinate utilizărilor menționate mai sus. Acestea sunt watchdog timer (WT) și sunt destinate monitorizării funcționării controlerului și a fluxului de cod program. Timerul WT are propriul ceas separat care nu este legat de CPU. Funcționarea sa este relativ simplă. WT este setat la un interval de timp care poate fi în intervalul de

milisecunde până la secunde. La început, contorul se reduce la zero. Fiecare sarcină, functie, subprogramele de întrerupere reseteaza contorul WT, adica cronometrul are valoarea de pornire după fiecare resetare. Dacă nu reușim să resetăm contorul înainte de a ajunge la zero, resetarea controlerului hardware va fi declanșată ("kick the dog"). Controlerul este repornit. Cu temporizatorul WT am împiedicat controlerul să rămână blocat în cicluri sau să nu mai răspundă.

8.4.5. Interfețe de comunicare

Interfețele de comunicare permit comunicarea între microcontroler și alte dispozitive externe, cum ar fi alte controlere, PC-uri, senzori etc. În această unitate vom prezenta doar interfețele care sunt implementate direct în chip-ul controlerului. Interfețele de comunicație diferă în funcție de caracteristici diferite, cum ar fi interfața paralelă sau seria, comunicarea sincronă sau asincronă, modul punct-la-punct sau rețea, modul semi-duplex sau full-duplex, modul wire sau wireless. Mai jos vor fi prezentate numai interfețele de rețea.

Interfața în serie intermitentă transmite un bit într-un singur tact de ceas. Aceasta înseamnă că depinde de viteza ceasului. În funcție de frecvența ceasului, determinăm numărul de biți transmiși pe unitate de timp (bit / s), numit viteză de comunicație. Din această cauză, interfața de comunicație serială necesită mai puține linii de date fizice. Interfața paralelă a transmis fiecare bit prin reețeua separată fizic. Dacă transmitem date pe 8 biți, avem nevoie de cel puțin 8 conexiuni fizice între două interfețe. Conexiunea paralelă este mai rapidă decât în serie. Dezavantajul său este acela că necesită prea multe legături fizice și mai mulți pini de controler. Comunicarea paralelă este adesea folosită pentru distanțe foarte scurte, afișaje LCD, citirea ADC-urilor rapide și a DAC-urilor, senzori diferiți etc. Conexiunea în serie este adesea folosită pentru distanțe mai scurte, precum și pe distanțe mai mari datorită simplității acesteia.

În multe cazuri, comunicarea între două sisteme este reciprocă. Aceasta înseamnă că ambele dispozitive pot primi sau transmite date. Următoarea întrebare este când un dispozitiv poate transmite sau primi date. Dacă folosim modul full-duplex, ambele dispozitive pot primi sau transmite date în același timp. Din punct de vedere fizic, aceasta înseamnă că au două linii separate pentru transmiterea (Tx) și primirea (Rx). În conexiune în serie aceasta înseamnă că avem două fire; unul pentru transmitere, unul pentru recepție. În modul paralel pe 8 biți, aceasta înseamnă 8 fire pentru recepție și 8 fire suplimentare pentru transmitere. Dacă folosim modul semi-duplex, ambele dispozitive recepționează și transmit pe aceeași linie, ceea ce înseamnă că nu este permisă concurența. Aceasta ar însemna coliziune și pierdere de date. Acest mod necesită mai puține conexiuni fizice, dar crește complexitatea protocolului de comunicare.

În interfețele de comunicare, adesea întâlnim termenul "master-slave"/comandant-subordonat. Acest mod este adesea utilizat în modul în serie. Managerul de dispozitiv decide când subordonatele poate accesa rețeaua și ce poate face în rețea (recepționează sau transmite date). Dacă dispozitivele se află în rețea, este permis ca numai un singur dispozitiv să fie comandantul, iar altele sunt subordonate.

Cele mai comune interfețe de comunicație pentru microcontrolere sunt USART, SPI, I2C etc. Unii controlori avansați au integrat, de asemenea, interfețe complexe, cum ar fi CAN, USB și TCP / IP.

- ***USART*:** Este o conexiune în serie (transmițător sincron transmițător asincron universal) care utilizează numai două căi de comunicație. O linie este pentru transmiterea Tx și altele pentru primirea Rx. Diferența dintre USART și UART este că USART are nevoie de o linie separată pentru ceas. UART are un ceas prestabilit, cu care sunt familiarizate toate celelalte dispozitive din calea de comunicare. În USART, dispozitivul de recepție utilizează ceasul expeditorului. Imaginea 13 prezintă interfața de comunicare UART.

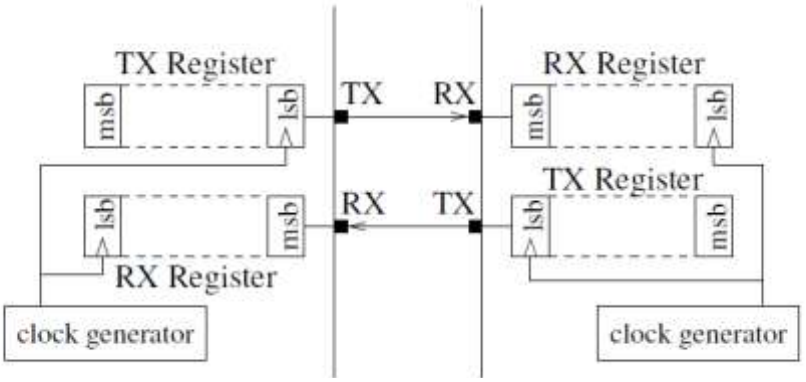

IMAGE 13: UART COMMUNICATION INTERFACE

Parametrii reglabili ai interfeței seriale USART sunt:

- ***Numărul de biți de date:*** Numărul de biți de date determină câte biți vor fi capturați în datele trimise. Acest lucru poate varia de la 5 la 9 biți de date. Cele mai frecvente sunt datele de 8 biți.
- ***Bitul de paritate:*** Utilizatorul poate decide dacă paritatea va fi utilizată sau nu. Paritatea poate fi un bit, par "1" sau impar "0". Este folosit ca un control simplu asupra regularității comunicării. Un exemplu de utilizare a parității parțiale și imparțiale în cazul datelor pe 7 biți este în imaginea 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

IMAGE 14: EVEN AND ODD PARITY

- ***Bitul de stop:*** bitul de anulare anunță când transferul este terminat. Poate fi de 1 sau 2 biți lungi.
- ***Viteza de transfer (unitatea de viteză):*** Viteza de transfer este determinată de tactul ceasului. Cu cât este mai mare tactul ceasului, cu atât este mai rapid transferul. De cele mai multe ori sunt utilizate următoarele viteze de conectare: 9.6kb / s, 38kp / s, 115.2kb / s, 1Mb / s, 10Mp / s.

Conexiunea în serie este cunoscută sub standardul RS232, RS422, RS485 etc. RS232 este o conexiune între două puncte unde nivelul de tensiune al frontului pozitiv este determinat între 3-15 V. Ambele dispozitive conectate la RS232 trebuie să aibă același potențial și au GND comun. Pentru utilizarea RS232 și UART pe microcontroler se utilizează interfețe care măresc sau scad nivelul de tensiune pentru RS232 standard. De multe ori se folosește și interfața MX232.

RS422 este aceeași comunicație ca RS232, dar utilizează nivele de tensiune diferențiale. Acestea permit transferul pe distanțe lungi. Dispozitivele nu au nevoie de GND comune.

RS485 este similar cu standardul RS422, dar permite 32 de dispozitive pe aceeași magistrală, în timp ce RS232 și RS422 permit doar o pereche. În ambele standarde RS422 și RS485 viteza de conectare scade cu lungimea conexiunii.

- **SPI:** SPI (interfață serial periferică) este o conexiune în serie destinată comunicării între dispozitive apropiate. Distanțele permise sunt de la numai câțiva centimetri până la maximum un metru. Comunicarea este asigurată de modul complet duplex între comandant și subordonat. Interfața suportă de asemenea modul 3,2,1 rețea.

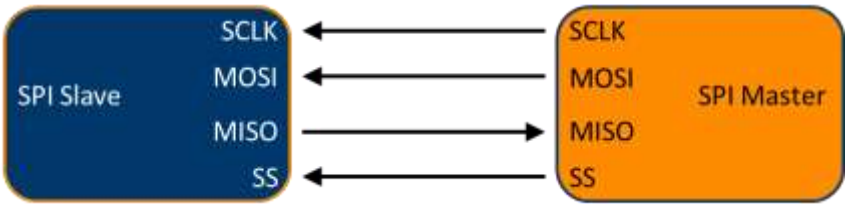

IMAGE 15: SPI COMMUNICATION

Comunicarea SPI utilizează pe deplin patru linii:

MOSI: (master out slave in): Conexiune pentru transmiterea datelor de la master la slave.

MISO: (master in slave out): Conexiune pentru primirea datelor de la slave la master.

SCK: (ceas de sistem): viteza de comunicare generată de ceas.

SS sau CS: (selectare slabă sau selecție cip): Adresarea dispozitivului de către comandant.

Comunicarea SPI permite conectarea mai multor dispozitive la aceeași rețea SPI (MOSI, MISO, SCK). Fiecare dispozitiv are doar un pin SS / CS separat, după cum se vede în imaginea 16. Comunicarea SPI trimite pachete de 8 biți. Viteza de comunicare este legată de ceasul CPU al vitezei master și ceasului slave-ului.

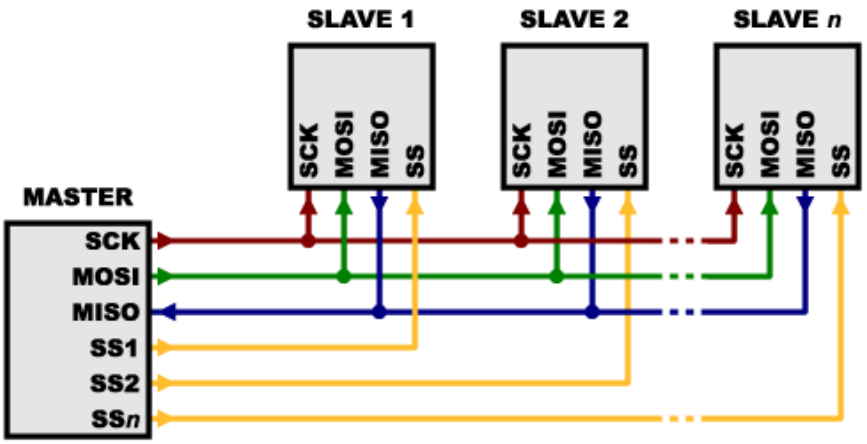

IMAGE 16: SEVERAL DEVICES ON SPI BUS

- **IIC (I^2C):** Este comunicare în seriecare permite modul semiplex. De asemenea, utilizează principiul master-slave. La fel ca SPI, IIC este destinat, de asemenea, pentru comunicarea dintre dispozitive apropiate. Viteza magistralei IIC este determinată de trei factori. Modul lent este de 100 kbit / s, modul rapid de 400 kbit / s și cea mai mare viteză de 3,4 Mbit / s. Interfața de comunicare necesită doar două rânduri. O linie este data SDA, iar cea de-a doua este generată de ceasul SCL de către comandant. Rețeaua permite transferul de date de 10 sau 7 biți. Dispozitivul de pe magistrala IIC are o adresă de 7 biți. La începutul comunicării, comandantul adresează dispozitivului adresa, apoi

începe să primească și să transmită date. Interfața permite mai multe dispozitive pe o magistrală, așa cum se vede în imaginea 17.

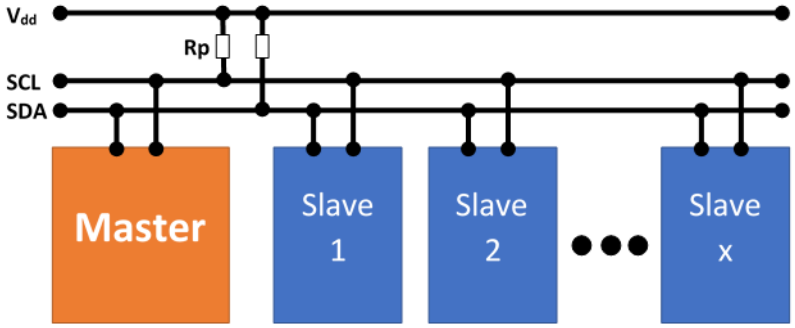

IMAGE 17: IIC COMMUNICATION INTERFACE

Comunicarea SPI este mai rapidă decât comunicarea I2C și are nevoie fizic de mai mulți pini decât rețeaua I2C. Comunicarea SPI este adesea utilizată pentru gestionarea afișărilor grafice pentru carduri SD și pentru captarea imaginilor de pe cipul optic. I2C este adesea utilizat în interfața în care nu există limite de timp definite strict.

Microcontrolerele conțin adesea mai multe interfețe de comunicare. De exemplu, aruncați o privire asupra microcontrolerului STM32F407 unde avem 6 interfețe USART, 3 interfețe SPI și 3 interfețe IIC.

8.5. Linii directoare privind consumul redus de energie al microcontrolerelor și aspectele ecologice

Fiecare controler poate fi configurat pentru a avea cel mai mic consum posibil. Primul pas spre designul ecologic necesită alegerea unui controler în funcție de caracteristicile acestuia și de dimensiunea chipului. Dimensiunea cipului este direct legată de materialele folosite în cip, cum ar fi carcasa din plastic, pinii metalici sau siliciul. Materialul utilizat la sfârșitul ciclului de viață afectează și reciclarea. Când alegem un controler, nu vom alege cipul de 144 pin și vom folosi doar câteva intrări și ieșiri. Pentru aceasta, este logic să alegeți o versiune mai mică din aceeași familie. Un alt exemplu sunt controlerii foarte eficienți. Aceștia utilizează mai multă energie decât controlerii simpli. Pentru fiecare dispozitiv sau aplicație, este important să evaluăm ce controler vom folosi. Pe piață există mai mulți controleri care sunt destinați consumului redus și autonomiei îndelungate.

Atunci când alegeți un controler, este bine să luați în considerare următoarele aspecte:

- **Ceasul controlerului:** Setarea ceasului de comandă este strâns legată de consumul de energie și de autonomia sistemului. Frecvența este proporțională cu energia. În fiecare controler, este posibilă setarea procesorului în anumite limite recomandate de producător. Dacă sarcinile

controlerului nu au nevoie de timp îndelungat, atunci tactul controlerului poate fi redus până când acesta îndeplinește criteriile de timp pentru sarcini. Majoritatea controlorilor au un ceas gestionat de program, ceea ce înseamnă că în timpul executării unor sarcini complexe creștem tactul ceasului și asigurăm rapid executarea. Când controlerul lucrează la sarcini mai simple și mai dificile, ceasul este redus în măsura în care sarcinile funcționează normal.

- **Tensiunea de alimentare:** Majoritatea controlerilor au tensiune de alimentare reglabilă într-o anumită zonă. Tensiunea de alimentare este, de asemenea, proporțională cu consumul de energie. În majoritatea cazurilor, tensiunea de alimentare este legată și de eficiența controlerului și de tactul ceasului CPU. Adesea întâlnim limitarea faptului că la tensiune mai mică nu putem seta tactul ceasului mai mare. La alegerea tensiunii de alimentare, este necesar să se facă un compromis între eficiență și consumul de energie.
- **Dezactivarea modulelor neutilizate:** De multe ori, când proiectăm codul de program, nu folosim multe dispozitive periferice. Controlerul permite ca dispozitivele periferice să fie oprite, ceea ce înseamnă că acestea nu consumă energie suplimentară atunci când nu sunt utilizate. Utilizarea dispozitivelor periferice poate fi, de asemenea, gestionată prin codul programului.
- **Designul optim al codului:** Atunci când proiectați un cod, este important să luați în considerare execuția. Cu cât codul este mai scurt, cu atât mai puține instrucțiuni trebuie executate de controler, adică controlerul poate fi în modul de așteptare pentru mai mult timp. La proiectarea codului trebuie să fim conștienți de tipul de variabile selectate și de numărul acestora. În programele mai mari, se poate întâmpla ca variabila să fie declarată și apoi utilizată rar sau chiar deloc folosită.
- **Modul de așteptare** (**Standby) :** Atunci când controlerul permite diferite moduri de așteptare, este bine să le folosiți, mai ales dacă avem pauze lungi între executarea sarcinilor.

References:

[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