

La modul general **un controler** este, actualmente, o structură electronică destinată controlului unui proces sau, mai general, unei interacțiuni caracteristice cu mediul exterior, fără să fie necesară intervenția operatorului uman.

Resursele integrate la nivelul microcircuitului **ar trebui să includă**, cel puțin, următoarele componente:

1. o unitate centrală (CPU), cu un oscilator intern pentru ceasul de sistem
2. o memorie locală tip ROM/PROM/EPROM/FLASH și eventual una de tip RAM
3. un sistem de întreruperi
4. I/O - intrări/ieșiri numerice
5. un port serial de tip asincron și/sau sincron, programabil
6. un sistem de timere-temporizatoare/numărătoare programabile

Este posibil ca la acestea să fie adăugate, la un preț de cost avantajos, caracteristici specifice sarcinii de control care trebuie îndeplinite:

7. un sistem de conversie analog numerică (una sau mai multe intrări analogice)
8. un sistem de conversie numeric analogic și/sau ieșiri PWM (cu modulare în durată)
9. un comparator analogic
10. o memorie de date nevolatilă de tip EEPROM
11. facilități suplimentare pentru sistemul de temporizare/numărare (captare și comparare)
12. un ceas de gardă (timer de tip watchdog)
13. un RTC (Real Time Clock – ceas, data, calendar)
14. alte protocoale de transmisie serială sau paralelă (I2C , SPI , 1WIRE , CAN...)
15. facilități pentru optimizarea consumului propriu

Un microcontroler tipic mai are, la nivelul unității centrale, facilități de prelucrare a informației la nivel de bit, de acces direct și ușor la intrări/ieșiri și un mecanism de prelucrare a întreruperilor rapid și eficient.

Utilizarea unui microcontroler constituie o soluție prin care se poate reduce dramatic numărul componentelor electronice precum și costul proiectării și al dezvoltării unui produs.

OBSERVAȚIE Utilizarea unui microcontroler, oricât de evoluat, nu elimină unele componente ale interfeței cu mediul exterior (atunci când ele sunt chiar necesare): subsisteme de prelucrare analogică (amplificare, redresare, filtrare, protecție-limitare), elemente pentru realizarea izolării galvanice (optocuploare, transformatoare), elemente de comutație de putere (tranzistoare de putere, relee electromecanice sau statice).

EXEMPLU : ATMEGA162

Capsula cu specificatii pentru fiecare PIN,

OBS : multi pini ai microcontrollerului au mai multe utilizari si resursele respective se folosesc numai pentru o anumita functionalitate

(OC0/T0) PB0	1	40	VCC
(OC2/T1) PB1	2	39	PA0 (AD0/PCINT0)
(RXD1/AIN0) PB2	3	38	PA1 (AD1/PCINT1)
(TXD1/AIN1) PB3	4	37	PA2 (AD2/PCINT2)
(<u>SS</u> /OC3B) PB4	5	36	PA3 (AD3/PCINT3)
(MOSI) PB5	6	35	PA4 (AD4/PCINT4)
(MISO) PB6	7	34	PA5 (AD5/PCINT5)
(SCK) PB7	8	33	PA6 (AD6/PCINT6)
<u>RESET</u>	9	32	PA7 (AD7/PCINT7)
(RXD0) PD0	10	31	PE0 (ICP1/INT2)
(TXD0) PD1	11	30	PE1 (ALE)
(INT0/XCK1) PD2	12	29	PE2 (OC1B)
(INT1/ICP3) PD3	13	28	PC7 (A15/TDI/PCINT15)
(TOSC1/XCK0/OC3A) PD4	14	27	PC6 (A14/TDO/PCINT14)
(OC1A/TOSC2) PD5	15	26	PC5 (A13/TMS/PCINT13)
(<u>WR</u>) PD6	16	25	PC4 (A12/TCK/PCINT12)
(<u>RD</u>) PD7	17	24	PC3 (A11/PCINT11)
XTAL2	18	23	PC2 (A10/PCINT10)
XTAL1	19	22	PC1 (A9/PCINT9)
GND	20	21	PC0 (A8/PCINT8)



UNDE SUNT UTILIZATE MICROCONTROLERELE?

Toate aplicațiile în care se utilizează microcontrolere fac parte din categoria așa ziselor sisteme încapsulate-integrate (“embedded systems”), la care existența unui sistem de calcul incorporat este (aproape) transparentă pentru utilizator.

Printre multele domenii unde utilizarea lor este practic un standard industrial se pot menționa: în industria de automobile (controlul aprinderii/motorului, climatizare, diagnoză, sisteme de alarmă, etc.), în așa zisa electronică de consum (sisteme audio, televizoare, camere video și videocasetofoane, telefonie mobilă, GPS-uri, jocuri electronice, etc.), în aparatura electrocasnică (mașini de spălat, frigidere, cuptoare cu microunde, aspiratoare), în controlul mediului și climatizare (sere, locuințe, hale industriale), în industria aerospațială, în mijloacele moderne de măsurare - instrumentație (aparate de măsură, senzori și traductoare inteligente), la realizarea de periferice pentru calculatoare, în medicină.

Există la ora actuală un număr extrem de mare de tipuri constructive de microcontrolere. Un criteriu de clasificare care se poate aplica întotdeauna este lungimea (dimensiunea) cuvântului de date. Funcție de puterea de calcul dorită și de alte caracteristici se pot alege variante având dimensiunea cuvântului de date de 4, 8, 16 sau 32 de biți (există chiar și variante de 64 de biți!). Nu este obligatoriu ca dimensiunea cuvântului de date să fie egală cu dimensiunea unui cuvânt mașină (cuvânt program). Există și multe variante zise dedicate, neprogramabile de utilizator la nivel de cod mașină, strict specializate pe o anumită aplicație, prin intermediul codului preprogramat și al resurselor hardware, utilizate pentru comunicații, controlul tastaturilor, controlul aparaturii audio/video, prelucrarea numerică a semnalului, etc.

Practic, toate microcontrolerele se realizează la ora actuală în tehnologie CMOS. Se pot realiza astfel structuri cu o mare densitate de integrare, cu un consum redus (care va depinde de frecvența de lucru). Logica internă este statică (total sau în cea mai mare parte) permițând astfel, în anumite condiții, micșorarea frecvenței de ceas sau chiar oprirea ceasului în ideea optimizării consumului. Tehnologia este caracterizată și de o imunitate mai mare la perturbații, esențială într-un mare număr de aplicații specifice. Se realizează variante pentru domeniu extins al temperaturii de funcționare (de ex. - 40 la +135 C).

Există foarte multe variante de încapsulare (capsule de plastic și mai rar de ceramică), multe din ele destinate montării pe suprafață (SMD): SOIC, PLCC, PQFP, TQFP (x100pini), etc., dar și variante clasice cu pini tip DIP/DIL (tipic de la 8 la 68 pini).

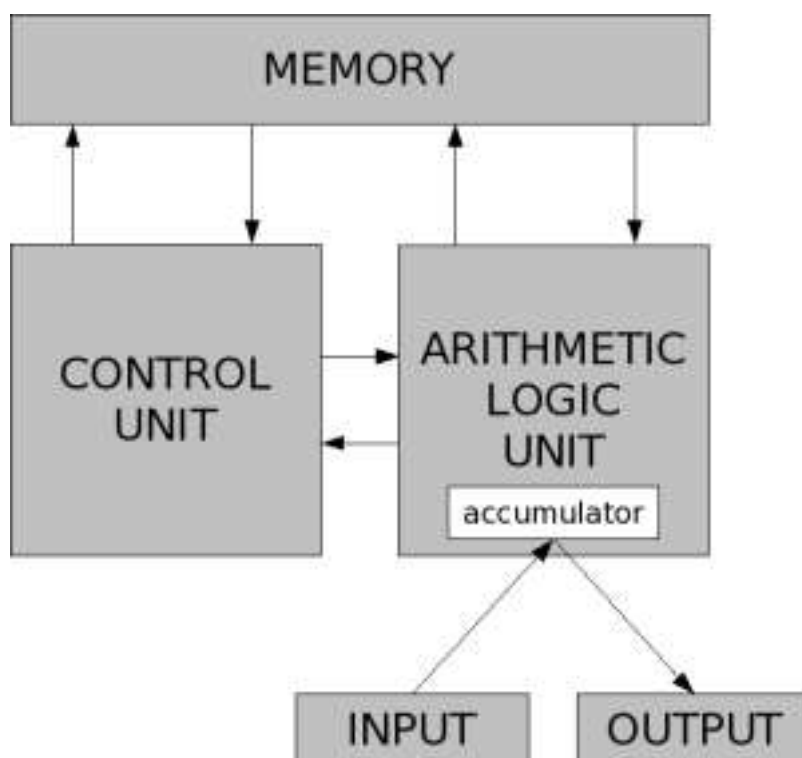
CARACTERISTICI ARHITECTURALE ALE UNITĂȚII CENTRALE

Arhitectura unității centrale de calcul (CPU) este unul din elementele cele mai importante care trebuie avut în vedere în analiza oricărui sistem de calcul. Principalele concepte luate în considerare și întâlnite aici sunt următoarele:

a. Arhitecturi de tip " von Neumann "

Cele mai multe microcontrolere sunt realizate pe baza acestei arhitecturi de sistem. Microcontrolerele bazate pe această arhitectură au o unitate centrală (CPU) caracterizată de existența unui singur spațiu de memorie utilizat pentru memorarea atât a codului instrucțiunilor cât și a datelor ce fac obiectul prelucrării. Există deci o singură magistrală internă (bus) care este folosită pentru preluarea a instrucțiunilor (fetch opcode) și a datelor; efectuarea celor două operații separate, în mod secvențial, are ca efect, cel puțin principal, încetinirea operațiilor. Este arhitectura standard (cea mai des întâlnită) și pentru microprocesoarele de uz general.

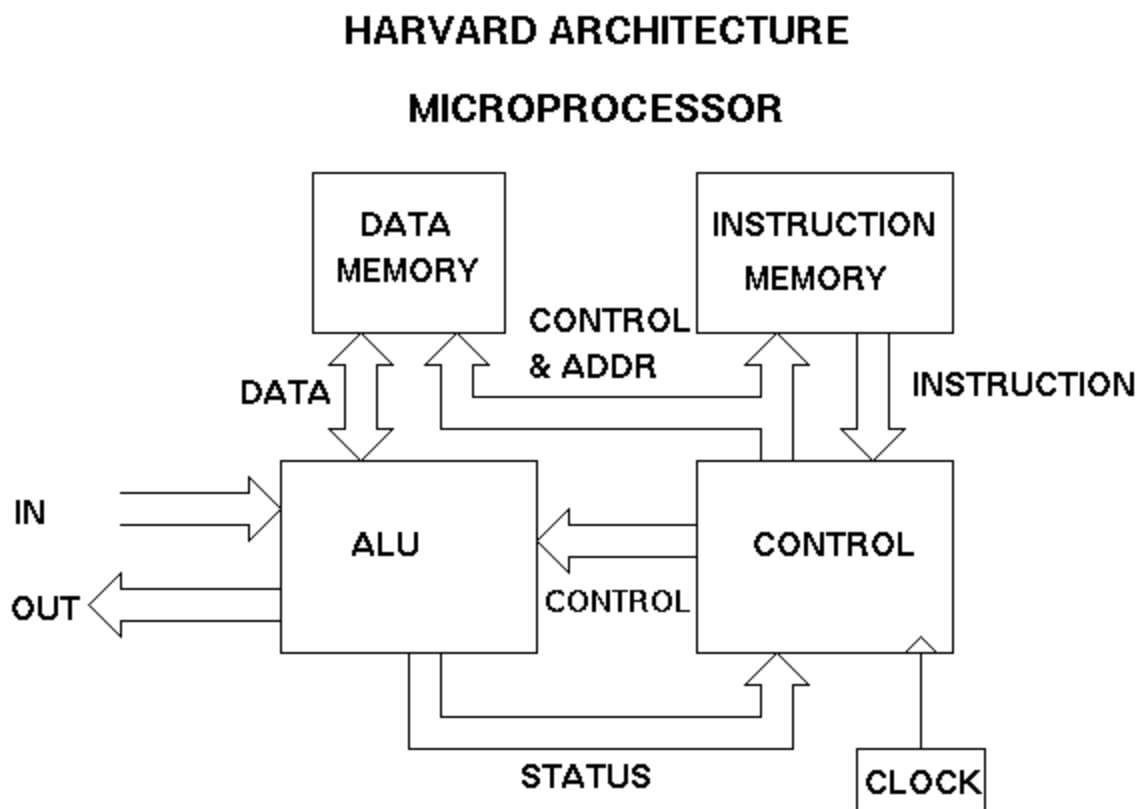
http://en.wikipedia.org/wiki/Von_Neumann_bottleneck



b. Arhitecturi de tip " Harvard "

La această arhitectură există spații de memorie separate pentru program și date. În consecință ar trebui să existe și magistrale separate (de adrese și date) pentru codul instrucțiunilor și respectiv pentru date. Principial există astfel posibilitatea execuției cvasiparalele (suprapunerii) a celor două operații menționate anterior. Codul unei instrucțiuni poate fi preluat din memorie în timp ce se execută operațiile cu datele aferente instrucțiunii anterioare. Este posibilă (cel puțin teoretic) o execuție mai rapidă, pe seama unei complexități sporite a microcircuitului, mai ales atunci când există și un pipeline. Este arhitectura standard pentru procesoarele numerice de semnal (DSP). Datorită costului mare al implementării unei astfel de arhitecturi, în cazul microcontrolerelor se întâlnește mai ales o arhitectură Harvard modificată, cu spații de memorie separate pentru program și date, dar cu magistrale comune pentru adrese și date.

http://en.wikipedia.org/wiki/Harvard_architecture



c. CISC

Aproape toate microcontrolerele au la baza realizării CPU conceptul CISC (*Complex Instruction Set Computer*). Aceasta înseamnă un set uzual de peste 80 instrucțiuni, multe din ele foarte puternice și specializate. De obicei multe din aceste instrucțiuni sunt foarte diferite între ele: unele operează numai cu anumite spații de adrese sau registre, altele permit numai anumite moduri de adresare, etc. Pentru programatorul în limbaj de asamblare există unele avantaje prin utilizarea unei singure instrucțiuni complexe în locul mai multor instrucțiuni simple (analog macroinstrucțiunilor clasice dintr-un limbaj de asamblare) .

<http://en.wikipedia.org/wiki/CISC>

d. RISC

RISC (*Reduced Instruction Set Computer*) este un concept de realizare a CPU care a început să fie utilizat cu succes de ceva timp și la realizarea microcontrolerelor. Prin implementarea unui set redus de instrucțiuni care se pot executa foarte rapid și eficient, se obține o reducere a complexității microcircuitului, suprafața disponibilizată putând fi utilizată în alte scopuri. Printre caracteristicile asociate de obicei unui CPU RISC se pot menționa:

- arhitectură Harvard modificată sau von Neumann
- viteză sporită de execuție prin implementarea unui pipeline pentru instrucțiuni
- set de instrucțiuni ortogonal (simetric): orice instrucțiune operează cu orice spațiu de adrese (de memorie) sau orice registru, instrucțiunile nu prezintă combinații speciale, excepții, restricții sau efecte colaterale.

<http://en.wikipedia.org/wiki/RISC>

Comparare între CISC și RISC:

<http://en.wikipedia.org/wiki/RISCvsCISC>

un vot interesant :

<http://www.xtrempc.ro/forum/viewtopic.php?t=32813&view=next&sid=112fe7674683f0ce676c6d511336d9ea>

ASPECTE LEGATE DE MEMORIA MICROCONTROLERELOR

În afară de memoria locală de tip RAM, de dimensiuni relativ reduse (de la x10 octeți la x1k), implementată ca atare sau existentă sub forma unui set de registre și destinată memorării datelor (variabilelor), mai există o serie de aspecte specifice, marea majoritate a acestora fiind legată de implementarea fizică a memoriei de program (și eventual a unei părți a memoriei de date) cu ajutorul unor memorii nevolatile. Clasic, memoria de program era implementată într-o variantă de tip ROM : EPROM pentru dezvoltare și producție pe scară mică/medie sau mask-ROM pentru producția de masă.

Principalele concepte noi apărute de a lungul timpului în legătură cu implementarea memoriei de program sau date sunt enumerate în continuare.

a. OTP - majoritatea producătorilor oferă variante de microcontrolere la care memoria locală de program este de tip OTP (One Time Programmable), practic o memorie PROM identică intern cu varianta EPROM, dar fără fereastra de cuarț pentru ștergere (deci și mai ieftine); aceste variante pot fi utilizate ca o alternativă pentru o producție limitată, până în momentul testării și validării finale a codului, moment în care pot fi comandate variantele (mask) ROM propriu-zise, cele mai economice pentru o producție de masă

b. FLASH EPROM - este o soluție mai bună decât EPROM-ul propriu-zis atunci când este necesar un volum mare de memorie program (nevolatilă); mai rapidă și cu un număr garantat suficient de mare (x10000) de cicluri de programare (de ștergere/scriere), este caracterizată și prin modalități mai flexibile de programare; este utilizată numai ca memorie de program.

c. EEPROM - multe microcontrolere au și o memorie de acest tip, de dimensiune limitată (de la x10 octeți la x K octeți), destinată memorării unui număr limitat de parametrii (**memorie de date**) care eventual trebuie modificați din timp în timp; este o memorie relativ lentă (la scriere), dar cu un număr de cicluri de ștergere/scriere mai mare ca FLASH-ul

d. NOVRAM (RAM nevolatil) - realizat prin alimentarea locală (baterie, acumulator) a unui masiv RAM CMOS atunci când este necesar un volum mare de memorie de program și date nevolatilă; mult mai rapidă decât toate celelalte tipuri și fără limitări ca număr de cicluri.

e. Memoria externă de program sau date. Marea majoritate a familiilor de microcontrolere permit și utilizarea de memorie externă de program (tipic ROM) sau date (tipic RAM). Aceasta presupune existența și utilizarea **unor magistrale externe de adrese și date**. Conexiunile externe necesare pentru acestea sunt disponibile ca funcții alternative ale pinilor. Din păcate, în această situație numărul de conexiuni exterioare disponibile pentru interfața cu exteriorul se reduce dramatic, reducând mult din versatilitatea microcontrolerului. Mai mult la variantele constructive cu un număr mic de pini (conexiuni externe) nu este posibilă utilizarea de memorie externă, decât, eventual, într-o variantă cu interfață serială (memorie RAM, FLASH sau EEPROM cu interfață I2C, SPI, etc.) și numai ca memorie de date.

LIMBAJE DE PROGRAMARE

Limbajul mașină și de cel de asamblare

Limbajul mașină (instrucțiunile mașină) este singura formă de reprezentare a informației pe care un microcontroler o "înțelege". Din păcate această formă de reprezentare a informației este total nepractică pentru un programator, care va utiliza cel puțin un limbaj de asamblare, în care o instrucțiune are drept corespondent o instrucțiune în limbaj mașină.

Un program în limbaj de asamblare este rapid și compact. Aceasta nu înseamnă că un astfel de program, prost scris, nu poate fi lent și de mari dimensiuni, programatorul având controlul total pentru execuția programului și gestiunea resurselor. Utilizarea numai a limbajului de asamblare pentru dezvoltarea unei aplicații complexe este neproductivă de multe ori, deoarece există și familii de microcontrolere cu CPU de tip CISC care au un număr foarte mare de instrucțiuni (x100) combinate cu moduri de adresare numeroase și complicate.

Totuși, nu trebuie uitat că la ora actuală mulți din producătorii mari de microcontrolere oferă medii de dezvoltare software gratuite care includ programe de asamblare gratuite. De asemenea, comunitatea utilizatorilor diverselor familii de microcontrolere a dezvoltat și ea, în timp, multe astfel de asamblatoare, care sunt disponibile ca freeware.

Interpretare

Un interpretor este o implementare a unui limbaj de nivel înalt, mai apropiat de limbajul natural. Caracteristic pentru execuția unui program interpretat, este citirea și executarea secvențială a instrucțiunilor (instrucțiune cu instrucțiune). De fapt fiecare instrucțiune de nivel înalt este interpretată într-o secvență de instrucțiuni mașină care se execută imediat.

Cele mai răspândite interpretare sunt cele pentru limbajele C++, BASIC și JAVA .

Marele avantaj al utilizării unui interpretor este dezvoltarea **interactivă și incrementală** a aplicației: se scrie o porțiune de cod care poate fi testată imediat, instrucțiune cu instrucțiune; dacă rezultatele sunt satisfăcătoare se poate continua cu adăugarea de astfel de porțiuni până la finalizarea aplicației.

Compilatoare

Un compilator combină ușurința în programare oferită de un interpret (de fapt de limbajul de nivel înalt) cu o viteză mai mare de execuție a codului. Pentru aceasta programul, în limbaj de nivel înalt, este translatat (tradus) direct în limbaj mașină sau în limbaj de asamblare. Codul mașină rezultat are dimensiuni relativ mari (dar mai mici decât cel interpretat) și este executat direct, ca un tot, de microcontroler. De regulă codul generat poate fi optimizat fie ca dimensiune, fie ca timp de execuție.

Cele mai populare și utilizate sunt cele pentru limbajul C.

Deci codul este obținut cu ajutorul unui **mediu integrat de dezvoltare a programelor (IDE-Integrated Development Environment)** care conține în mod tipic următoarele componente software:

- un editor specializat (orientat pe codul sursă),
- un asamblor/compilator,
- un editor de legături/locator ("link-editor/locator"),
- programe de gestiune a unor biblioteci de cod ("librarians"),
- programe de conversie a formatelor de reprezentare a codului (de exemplu din binar în format Intel HEX)
- un simulator și/sau depanator ("debugger")
- posibilitatea de atasare a unui programator ICSP

DEZVOLTAREA ȘI TESTAREA APLICAȚIILOR

Cele mai răspândite mijloace hardware/software utilizate în dezvoltarea și testarea aplicațiilor sunt enumerate în continuare.

a. Simulatoarele .

Un simulator este un program care rulează programul microcontrolerului - implementează un microcontroler virtual - folosind un sistem de calcul gazdă un PC. Programul se poate executa pas cu pas, conținutul variabilelor și registrelor poate fi vizualizat și modificat. Reprezintă un punct de plecare atunci când se abordează un microcontroler, pentru familiarizarea cu resursele lui și cu limbajul de asamblare. Nu permite simularea în timp real a întreruperilor și, de regulă, programul rulează mai încet decât pe mașina reală. De regulă există mijloace pentru evaluarea vitezei de execuție a codului simulat (ca număr de cicluri mașină sau de stări). Ideal, un simulator ar trebui să permită și simularea completă a interacțiunii, cel puțin din punct de vedere logic, cu toate perifericele disponibile.

b. Programele de depanare ("debuggers").

Sunt programe "monitor" care pe mașina țintă (microcontrolerul) oferind facilități de depanare similare simulatorului. Interfața cu utilizatorul este realizată prin intermediul unui sistem gazdă (PC) și/sau a unui terminal alfanumeric, conectate prin intermediul unui port serial. Utilizează o parte din resursele microcontrolerului : memorie de program pentru el însuși (de tip ROM) și memorie de date (RAM) pentru variabile proprii, memorie pentru programul ce se depanează, un port serial pentru comunicația cu sistemul gazdă, eventual întreruperi, etc. Se utilizează de regulă împreună cu un sistem de dezvoltare (sau evaluare), care este un sistem minimal realizat în jurul microcontrolerului pe care rulează depanatorul, dar având resurse suficiente pentru a permite testarea și depanarea aplicațiilor uzuale.

b. Emulatoarele In Circuit (ICE-In Circuit Emulators).

Sunt cele mai eficiente mijloace de testare și dezvoltare și au fost mult timp cele mai complexe și mai costisitoare. Presupune existența unui hardware dedicat care înlocuiește practic microcontrolerul (se conectează în locul acestuia în sistemul pentru care se dezvoltă aplicația), în același timp fiind disponibile toate facilitățile descrise anterior și altele suplimentare. Permit un control total al mașinii țintă (în timp real), fără a folosi nimic din resursele. Ele sunt realizate de cele mai multe ori ca un mijloc de testare și depanare de sine stătător, conectat la un PC prin intermediul unui port paralel, serial sau USB. Nu este necesară înlocuirea microcontrolerului de pe sistemul țintă, conectarea cu acesta făcându-se printr-un număr minim de interconexiuni. Exemple de astfel de interfețe ar fi: JTAG/ICE – In Circuit Emulation - la multe familii de microcontrolere, BDM (Background Debug Monitor) – pentru Freescale/Motorola. Existența acestui tip de interfețe face posibilă realizarea de emulatoare cu un preț de cost mult mai mic decât cele clasice.

d. Simulatoarele de sistem

Reprezintă o categorie aparte de simulatoare destinate simulării cât mai complete a sistemului și a aplicației în ansamblu, cu alte cuvinte a microcontrolerului împreună cu o dispozitivele hardware externe. Ele integrează de regulă și un simulator SPICE. Cele mai cunoscute sunt Proteus VSM (Virtual System Modelling) al firmei Labcenter Electronics . Un astfel de simulator permite rularea aplicației (codului), în mod continuu sau pas cu pas și evaluarea în detaliu a modului cum aceasta (și microcontrolerul) interacționează cu hardware-ul extern. El permite ceea ce se numește co-simularea (Co-simulation): interacțiunea dintre software-ul microcontrolerului și dispozitivele electronice analogice sau

numerice conectate cu acesta. Sunt bazate pe utilizarea unor modele avansate ale unor familii de microcontrolere precum și pe modelele SPICE ale dispozitivelor electronice. Pentru fiecare model de microcontroler există un asamblor și un editor de legături integrat astfel că se poate face, în anumite limite, și dezvoltarea codului în asamblare. Pe lângă aceasta ele au asigurate și interfețe corespunzătoare pentru a se putea dezvolta codul cu medii de programare consacrate pentru familia respectivă de microcontrolere, folosind de exemplu un compilator C. Facilitățile de simulare a codului sunt similare celor întâlnite la simulatoarele deja menționate. Pe lângă numeroasele dispozitive electronice discrete, circuite integrate analogice sau numerice, circuite de memorie sau periferice, în categoria dispozitivelor externe se mai pot menționa și sisteme de afișare (LED, LCD), tastaturi matriciale sau butoane, relee, etc.

e. Nucleele (sistemele de operare) de timp real (Real Time kernel, Real Time Operating System-RTOS)

Pe piața de software pentru microcontrolere există și componente numite nuclee de timp real sau sisteme de operare în timp real (RTOS). Un astfel de program de sistem de nivel profesional este o componentă software scumpă sau foarte scumpă, funcție de complexitatea lui, de accesibilitatea surselor programului, de familia de microprocesoare căreia îi este adresat, de modul în care va fi distribuit împreună cu aplicația. Există însă și variante de RTOS, de mai mică complexitate, din categoria freeware sau shareware, care pot fi utilizate cu performanțe mulțumitoare.

Un sistem de operare în timp real facilitează crearea aplicațiilor așa zise de timp real, dar nu garantează și faptul că ele chiar se vor executa în timp real, aceasta depinzând de modul în care este utilizat acest software la nivel de sistem.

Spre deosebire de un calculator cum este PC-ul, un sistem integrat (embedded system) este proiectat întotdeauna într-un anumit scop și are un cod care se execută aproape întotdeauna dintr-o memorie ROM, fiind de presupus că nu se modifică pe parcursul execuției aplicației. Astfel lucrurile sunt ușurate deoarece comportarea sistemului poate specificată complet încă din faza de proiectare. Din aceasta cauză, în cazul multora din aplicații, multe probleme se pot rezolva în timp real și fără să se utilizeze un RTOS. Esențială este până la urmă calitatea și competența celui care programează aplicația !

CRITERII DE ALEGERE A UNUI MICROCONTROLLER

În momentul în care se dorește alegerea unui microcontroller (sau mai bine zis a unei familii de microcontrolere) pentru dezvoltarea unei aplicații de tip “embedded system” există mai multe criterii care trebuie luate în considerare, ținând cont de implicațiile multiple ale acestei alegeri. Vom încerca să grupăm aceste criterii după cerințele impuse aplicației și să prezentăm câteva din întrebările rezultate, la care trebuie dat un răspuns.

a. Costurile aplicației

Care va fi scara de producție: prototip, producție mică/medie sau de masă?

Care sunt costurile permise pentru microcontroller?

Care sunt costurile permise pentru mediul de programare și dezvoltare?

b. Timpul de dezvoltare al aplicației

Ce limbaj de programare să aleg?

Ce limbaje de programare cunosc bine și ce medii de dezvoltare am utilizat?

Ce modalitate de testare și depanare folosesc: simulator, sistem de dezvoltare, emulator?

c. Caracteristicile fizice

Care este viteza de prelucrare (de calcul) necesară?

De câtă memorie am nevoie pentru program și respectiv date?

Va fi necesară și o memorie externă?

Ce fel de alimentare este disponibilă și care sunt limitările acesteia?

De câte intrări și/sau ieșiri am nevoie?

Ce fel de intrări și/sau ieșiri sunt necesare: intrări/ieșiri analogice, ieșiri numerice de curent mai mare?

Care sunt resursele necesare în materie de temporizare/numărare și care ar fi caracteristicile lor cele mai importante (rezoluție, frecvența maximă de numărare) ?

Ce tip de capsulă, ce dimensiuni fizice și număr de pini ar trebui să aibă?

Care este gama temperaturilor de lucru necesare?

Aplicația va funcționa într-un mediu cu caracteristici speciale, de exemplu în care există perturbații electromagnetice puternice?

d. Conectivitatea

Care sunt resursele de comunicație necesare: câte porturi seriale asincrone și cu ce caracteristici, ce tipuri de magistrale seriale sincrone sunt disponibile?

Este necesară o conectivitate Ethernet (o stivă TCP/IP), USB sau wireless (stive Bluetooth, Zigbee, etc.)?

e. Compatibilitate, scalabilitate și dezvoltarea ulterioară

Cu ce tipuri de circuite se poate interfața cât mai simplu: sisteme de afișare, senzori, elemente de comandă și execuție (relee, motoare de cc, motoare pas cu pas, etc.)?

Cum se poate realiza extinderea ulterioară, atunci când este necesară?

Există mai multe variante în familia respectivă de microcontrolere, care să acopere eventualele cerințe suplimentare în materie de viteză de lucru, resurse periferice sau de memorie?

f. Alte aspecte

Ce distribuitori există și cât sunt de accesibili pentru mine?

Care este suportul oferit de fabricant sau distribuitor și care este baza de cunoștințe existentă: site-uri web, documentație on-line sau pe CD-uri, note de aplicații, exemple de proiectare (reference designs), software din categoria freeware/shareware și, nu în ultimul rând, forumuri de discuții pentru utilizatori?

Din păcate răspunsurile la multe din aceste întrebări sunt corelate între ele. Un exemplu este legătura care există între criteriile de cost și cele de timp de dezvoltare. Principial, utilizarea unui limbaj de nivel înalt împreună cu un emulator pentru testare și depanare poate duce la scurtarea consistentă a timpului de dezvoltare. Dar prețul unui compilator este întotdeauna mai mare decât cel al unui asamblor, iar prețul unui emulator este și el mai mare decât cel al unor mijloace mai simple de testare și depanare.

În practică, de cele mai multe ori, **alegerea unui microcontroler pentru a anumită aplicație este și trebuie să fie rezultatul unui compromis.**

CU CE SI CUM SE POATE PROGRAMA UN MICROCONTROLLER?

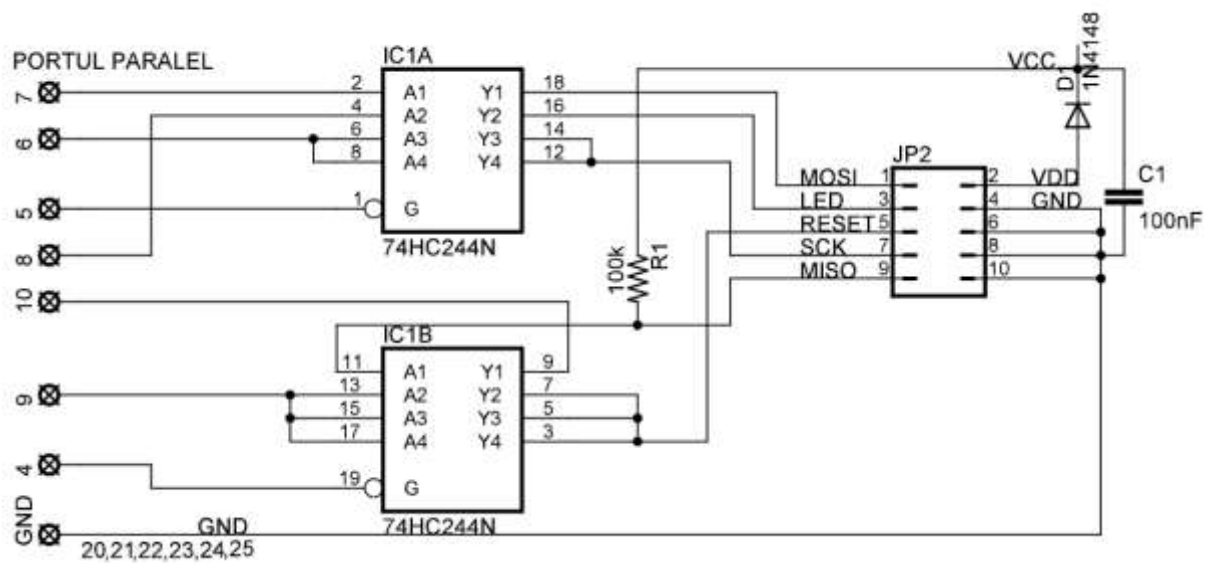
a. Programator specializat extern.(ex.STK200)- este un circuit electronic care conecteaza un port al PC-ului (serial,paralel sau USB) cu pinii de programare ai microcontrollerului. Programatoarele acestea au de obicei un soclu ZIP in care se pot conecta mai multe tipuri de microcontrolere in diferite pozitii, pentru ca pinii de programare diferă ca pozitie de la familie la familie . Unele programatoare pot programa chiar si diferite familii de microcontrolere.

b. Programarea "In System" (ISP-In System Programming) - folosirea unor memorii nevolatile de tip FLASH face posibilă și "programarea" unui astfel de microcontroller fără a-l scoate din sistemul în care este încorporat (programare on-line, In System Programming) ; programarea se face de regulă prin intermediul unei interfețe seriale dedicate de tip ISP (poate avea nume diferite) sau a unei interfețe standard JTAG.

c. Bootloader – multe din microcontrolerele recente la care memoria de program este de tip FLASH au și facilitatea (au de fapt instrucțiuni dedicate acestui scop) de a putea și scrie în această memorie de program fără a utiliza un circuit de programare extern. Astfel în microcontroller poate exista permanent (rezident) un cod de mici dimensiuni (denumit și bootloader) care pur și simplu va încărca prin intermediul portului serial (este doar un exemplu) codul utilizator sau constantele pe care acesta vrea eventual să le actualizeze. Bootloader-ul este și cel care lansează în execuție programul utilizator după încărcarea acestuia.

ATENȚIE !!! Protejarea codului - protejarea codului program dintr-o memorie locală nevolatilă împotriva accesului neautorizat (la citire –deoarece pirateria soft există și aici) este oferită ca o opțiune (ea mai trebuie și folosită!) la variantele FLASH, EPROM sau OTP. Codul poate protejat atât la citire cat și la scriere (practic circuitul trebuie șters, înainte de a se mai putea scrie ceva în el). Este eliminată astfel posibilitatea de a se realiza, în acest caz, de patch-uri (alterări cu un anumit scop) ale codului original. La variantele mask-ROM propriu-zis protecția este de cele mai multe ori implicită.

Schema electrica pentru programatorul ICSP : STK200



Programator pe SPI sau JTAG



Programator UNIVERSAL

