Capitolul 1 INTRODUCERE

1.1 CE ESTE DE FAPT UN MICROCONTROLER?

La modul general un controler ("controller" - un termen de origine anglo-saxonă, cu un domeniu de cuprindere foarte larg) este, actualmente, o structură electronică destinată controlului (destul de evident!) unui proces sau, mai general, unei interacțiuni caracteristice cu mediul exterior, fără să fie necesară intervenția operatorului uman. Primele controlere au fost realizate în tehnologii pur analogice, folosind componente electronice discrete şi/sau componente electromecanice (de exemplu relee). Cele care fac apel la tehnica numerică modernă au fost realizate inițial pe baza logicii cablate (cu circuite integrate numerice standard SSI şi MSI) şi a unei electronici analogice uneori complexe, motiv pentru care "străluceau" prin dimensiuni mari, consum energetic pe măsură şi, nu de puține ori, o fiabilitate care lăsa de dorit.

Apariția şi utilizarea microprocesoarelor de uz general a dus la o reducere consistentă a costurilor, dimensiunilor, consumului şi o îmbunătățire a fiabilității. Există şi la ora actuală o serie de astfel de controlere de calitate, realizate în jurul unor microprocesoare de uz general cum ar fi Z80 (Zilog), 8086/8088 (Intel), 6809 (Motorola), etc.

Pe măsură ce procesul de miniaturizare a continuat, a fost posibil ca majoritatea componentelor necesare realizării unei astfel de structuri să fie încorporate (integrate) la nivelul unui singur microcircuit (cip). Astfel că un microcontroler ar putea fi descris ca fiind şi o soluție (nu în sens exhaustiv !) a problemei controlului cu ajutorul a (aproape) unui singur circuit.

Legat de denumiri şi acronime utilizate, aşa cum un microprocesor de uz general este desemnat prin **MPU** (MicroProcessor Unit), un microcontroler este, de regulă, desemnat ca **MCU,** deşi semnificația inițială a acestui acronim este MicroComputer Unit.

O definiție, cu un sens foarte larg de cuprindere, ar fi aceea că un microcontroler este un microcircuit care incorporează o unitate centrală (CPU) şi o memorie împreună cu resurse care-i permit interacțiunea cu mediul exterior.

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

a. o unitate centrală (CPU), cu un oscilator intern pentru ceasul de sistem
b. o memorie locală tip ROM/PROM/EPROM/FLASH şi eventual una de tip RAM
c. un sistem de întreruperi
d. I/O - intrări/ieşiri numerice (de tip port paralel)
e. un port serial de tip asincron şi/sau sincron, programabil
f. 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:

g. un sistem de conversie analog numerică(una sau mai multe intrari analogice)
h. un sistem de conversie numeric analogic şi/sau ieşiri PWM (cu modulare în durată)
i. un comparator analogic
j. o memorie de date nevolatilă de tip EEPROM
k. facilități suplimentare pentru sistemul de temporizare/numărare (captare şi comparare)
l. un ceas de gardă (timer de tip watchdog)
m. 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 şi uşor la intrări/ieşiri şi un mecanism de prelucrare

a întreruperilor rapid şi eficient.

Utilizarea unui microcontroler constituie o soluție prin care se poate reduce dramatic numărul componentelor electronice precum şi costul proiectării şi 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).

1.2 UNDE SUNT UTILIZATE MICROCONTROLERELE?

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

Pentru ca utilizarea lor este de foarte ori sinonimă cu ideea de control microcontrolerele sunt utilizate masiv în robotică şi mecatronică. **Conceptul de mecatronică este pană la urmă indisolubil legat de utilizarea microcontrolerelor**.

Automatizarea procesului de fabricație-producție este un alt mare beneficiar: CNC Computerised Numerical Controls-comenzi numerice pentru maşinile unelte, automate programabile -PLC, linii flexibile de fabricație, etc.). Indiferent de natura procesului automatizat sarcinile specifice pot fi eventual distribuite la un mare număr de microcontrolere integrate într-un sistem unic prin intermediul uneia sau mai multor magistrale.

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 aşa zisa electronică de consum (sisteme audio, televizoare, camere video şi videocasetofoane, telefonie mobilă, GPS-uri, jocuri electronice, etc.), în aparatura electrocasnică (maşini de spălat, frigidere, cuptoare cu microunde, aspiratoare), în controlul mediului şi climatizare (sere, locuințe, hale industriale), în industria aerospațială, în mijloacele moderne de măsurare - instrumentație (aparate de măsură, senzori şi traductoare inteligente), la realizarea de periferice pentru calculatoare, în medicină.

„Johnnie" (figura 1.1) un robot umanoid destul de simplu, construit la Universitatea Tehnică din Munchen în 1998, utilizează 5 microcontrolere, conectate prin intermediul unei magistrale CAN la un calculator PC. „Alpha" un alt robot umanoid (fotbalist ca destinație) dezvoltat la Universitatea din Freiburg utilizează, intr-o variantă a sa, 11 microcontrolere conectate similar. Un număr foarte mare de microcontrolere sunt folosite şi de aşa zisele jucării inteligente, din care „capetele de serie" cele mai cunoscute sunt cei doi roboți, unul canin şi altul umanoid: AIBO (Sony, figura 1.2) şi ASIMO (Honda, figura 1.5). ASIMO foloseşte 26 de microcontrolere numai pentru controlul individual al celor 26 de elemente de acționare inteligente (motoare). Tot în categoria roboților umanoizi intra şi QRIO (Sony, figura 1.4) sau HOAP-1 (Fujitsu, figura 1.3). Roboții respectivi sunt produşi în serie, unii dintre ei chiar la un preț „accesibil".

Ca un exemplu din industria de automobile (automotive industry), unde numai la nivelul anului 1999, un BMW seria 7 utiliza 65 de microcontrolere, iar un Mercedes din clasa S utiliza 63 de microcontrolere.

Practic, deşi am prezentat ca exemple concrete numai sisteme robotice şi mecatronice, este foarte greu de găsit un domeniu de aplicații în care să nu se utilizeze microcontrolerele.

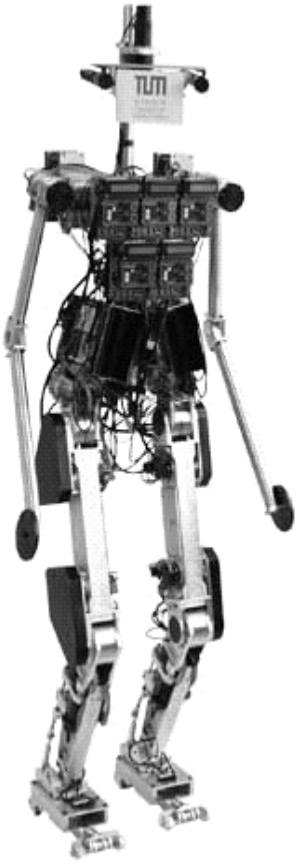

Figura 1.1 Johnnie

Figura 1.2 AIBO

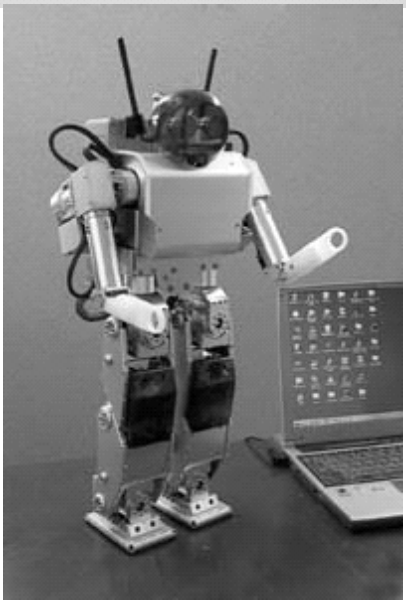

Figura 1.3 Hoap-1

Figura 1.4 QRIO

Figura 1.5 ASIMO

1.3 CLASIFICĂRI ŞI VARIANTE CONSTRUCTIVE

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 şi de alte caracteristici se pot alege variante având dimensiunea cuvântului de date de 4, 8,16 sau 32 de biți (există chiar şi variante de 64 de biți!). Nu este obligatoriu ca dimensiunea cuvântului de date să fie egală cu dimensiunea unui cuvânt maşină (cuvânt program). Există şi multe variante zise dedicate, neprogramabile de utilizator la nivel de cod maşină, strict specializate pe o anumită aplicație, prin intermediul codului preprogramat şi al resurselor hardware, utilizate pentru comunicații, controlul tastaturilor, controlul aparaturii audio/video, prelucrarea numerică a semnalului, etc.

1.4 TEHNOLOGIILE DE FABRICAȚIE UTILIZATE

Practic, toate microcontrolerele se realizează la ora actuală în tehnologie CMOS (tehnologii similare celor utilizate la seriile standard CMOS de circuite numerice: HC, AC, ALV, etc.). Se pot realiza astfel structuri cu o mare densitate de integrare, cu un consum redus (care va depinde de frecvența de lucru), permițând eventual alimentarea de la baterie. Logica internă este statică (total sau în cea mai mare parte) permițând astfel, în anumite condiții, micşorarea frecvenței de ceas sau chiar oprirea ceasului în ideea optimizării consumului. Tehnologia este caracterizată şi 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 +85 C).

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

1.5 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 şi 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 şi a datelor ce fac obiectul prelucrării. Există deci o singură magistrală internă (bus) care este folosită pentru preluarea a instrucțiunilor (fetch opcod) şi a datelor; efectuarea celor două operații separate, în mod secvențial, are ca efect, cel puțin principial, încetinirea operațiilor. Este arhitectura standard (cea mai des întâlnită) şi pentru microprocesoarele de uz general.

b. Arhitecturi de tip " Harvard "

La această arhitectură există spații de memorie separate pentru program şi date. În consecință ar trebui să existe şi magistrale separate (de adrese şi date) pentru codul instrucțiunilor şi 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 şi un pipeline. Este arhitectura standard pentru procesoarele numerice de semnal (DSP). Datorită costului mare al implementării unei astfel de arhitecturi, în cazul microcontrolerelor se întâlneşte mai ales **o arhitectură Harvard modificată**, cu spații de memorie separate pentru program şi date, dar cu magistrale comune pentru adrese şi date.

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 şi 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) .

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 şi la realizarea microcontrolerelor. Prin implementarea unui set redus de instrucțiuni care se pot executa foarte rapid şi 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.

1.5 ASPECTE LEGATE DE IMPLEMENTAREA MEMORIEI 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 şi destinată memorării datelor (variabilelor), mai există o serie de aspecte specifice, marea majoritate a acestora fiind legată de **implementarea fizică a memoriei de program** (şi 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 şi 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 ştergere (deci şi mai ieftine); aceste variante pot fi utilizate ca o alternativă pentru o producție limitată, până în momentul testării şi 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 şi cu un număr garantat suficient de mare (x10000) de cicluri de programare (de ştergere/scriere), este caracterizată şi prin modalități mai flexibile de programare; este utilizată numai ca memorie

de program.

c. EEPROM - multe microcontrolere au şi 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 ştergere/scriere mai mare ca FLASH-ul (x100000).

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 şi date nevolatilă; mult mai rapidă decât toate celelalte tipuri şi fără limitări ca număr de cicluri.

e. Programarea "In System" (ISP-In System Programming) - folosirea unor memorii nevolatile de tip FLASH face posibilă şi "programarea" unui astfel de microcontroler 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. Există microcontrolere la care această programare se poate face prin intermediul portului serial asincron sau al interfetei CAN (Controller Area Network). Este posibilă astfel modificarea cu uşurință a codului program sau a unor constante de lucru (local sau de la distanță-remote update). Vezi şi **Bootloader**.

Un lucru foarte important este că la anumite familii **interfața prin intermediul căreia se face programarea poate fi utilizată şi la testarea şi depanarea aplicației (soft)**, permițând realizarea simplă, cu un preț de cost minim, a unor mijloace de testare şi depanare(emulatoare). De exemplu, în acest caz interfața JTAG este specificată ca fiind **JTAG/ICE** (In Circuit Emulation) pentru a arăta că poate fi folosită şi pentru emularea în circuit.

f. Bootloader – multe din microcontrolerele recente la care memoria de program este de tip FLASH au şi facilitatea (au de fapt instrucțiuni dedicate acestui scop) de a putea şi scrie în această memorie de program fără a utiliza un circuit de programare extern. Astfel în microcontroler poate exista permanent (rezident) un cod de mici dimensiuni (denumit şi bootloader) care pur şi 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 şi cel care lansează în execuție programul utilizator după încărcarea acestuia.

g. Protejarea codului - protejarea codului program dintr-o memorie locală nevolatilă împotriva accesului neautorizat (la citire –deoarece pirateria soft există şi aici) este oferită ca o opțiune (ea mai trebuie şi folosită!) la variantele FLASH, EPROM sau OTP. Codul poate protejat atât la citire cat şi la scriere (practic circuitul trebuie şters, 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.

h. Memoria externă de program sau date

Marea majoritate a familiilor de microcontrolere permit si utilizarea de memorie externă de program (tipic ROM) sau date (tipic RAM). Aceasta presupune existenta si utilizarea **unor magistrale externe de adrese si date**. Conexiunile externe necesare pentru acestea sunt disponibile ca funcții alternative ale pinilor. Din păcate, in 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, intr-o variantă cu interfață serială (memorie RAM, FLASH sau EEPROM cu interfață I2C, SPI, etc.) si numai ca memorie de date.

1.6 CONTROLUL PUTERII CONSUMATE ŞI ALIMENTAREA LA TENSIUNI REDUSE

Majoritatea microcontrolerelor pot fi trecute sub control soft în stări speciale cum ar fi: în aşteptare (STAND-BY), inactiv ("IDLE") sau oprit ("HALT", "POWER DOWN"), denumirile acestor stări diferind şi funcție de fabricant. În aceste moduri starea CPU, conținutul RAM-ului intern, starea I/O-urilor poate fi conservată în condițiile unei tensiuni de alimentare reduse (față de cea nominală) şi deci al unui consum mult redus. Flexibilitatea acestui sistem este strans legata de faciltitatile si programabilitatea sistemului de generare a ceasului de lucru (clock system, vezi 1.7).

De exemplu, într-un mod de tip "IDLE" toate activitățile sunt oprite cu excepția circuitului oscilatorului local de ceas şi, dacă acestea există: circuitul "watchdog" (ceasul de gardă), circuitul de monitorizare a oscilatorului de ceas şi eventual un temporizator dedicat ("idle timer"). Puterea consumată este redusă la cca. 30%, iar ieşirea din acest mod se face prin reset (inițializare) sau printr-un stimul exterior (de regulă o întrerupere). Temporizatorul dedicat poate scoate periodic microcontrolerul din această stare, pentru îndeplinirea anumitor sarcini, după care se reintră în starea respectivă.

În modul"HALT" toate activitățile sunt oprite, tensiunea de alimentare poate fi coborâtă sub valoarea nominală, fără alterarea stării (CPU, RAM, I/O), puterea consumată fiind minimă. Ieşirea din această stare se face prin reset sau ca urmare a unei cereri de întrerupere. Toate aceste moduri de lucru sunt de regulă valorificate în aplicații în care alimentarea sistemului se face de la o sursă de tip baterie (în funcționarea normală sau numai în anumite situații).

Se mai oferă uneori ca facilitate o protecție la scăderea accidentală (în anumite limite) a tensiunii de alimentare ("brownout protection"). La scăderea tensiunii de alimentare sub o anumită limită ("brownout voltage") microcontrolerul este inițializat (resetat) şi ținut în această stare atâta timp cât condiția respectivă persistă.

Tensiunea de alimentare standard pentru microcontrolere a fost mult timp, din considerente şi istorice(vezi TTL-ul), de Vcc = 5V (cu o anumită toleranță). La ora actuală se oferă, pentru multe din ele, şi variante cu tensiune de alimentare redusă (Low Voltage) cu Vcc = 1.8 .. 3.3V, destinate unor aplicații unde consumul este un parametru critic. Există variante cu plajă mare a tensiunii de alimentare, de exemplu Vcc = 2..6V, funcționarea la limita inferioară implicând doar o micşorare a frecvenței maxime de ceas.

1.7 SISTEMUL DE CEAS

Orice microcontroler este caracterizat cel puțin de existența circuitelor electronice aferente oscilatorului care generează ceasul de sistem. Astfel este posibilă implementarea simplă a oscilatorului doar prin adăugarea, în exterior, a unui rezonator extern (cuarț sau piezoceramica) pentru stabilizarea frecventei si eventual a unor capacitori. Dacă stabilitatea si precizia frecvenței nu este o cerință importantă, la anumite microcontrolere se poate utiliza doar un circuit RC extern sau există un circuit RC intern, care determină frecvența de oscilație. Există microcontrolere la care configurația oscilatorului este programabilă prin intermediul unor „fuzibile” FLASH (se programează similar memoriei de program): rezonator extern şi tipul acestuia, varianta RC intern sau extern, gama de frecventă, etc.

La familiile evoluate de microcontrolere există si circuite de tip PLL (Phase Locked Loop) si/sau FLL (Frequency Locked Loop) care permit multiplicarea cu uşurință a frecvenței de

bază (cea a rezonatorului extern). Astfel plecând, de exemplu de la o frecvență de 32.768KHz se pot obține frecvențe de lucru până ordinul MHz. La astfel de microcontrolere sistemul de ceas este programabil prin intermediul unor registre speciale oferind un maxim de flexibilitate în sensul putinței de controla compromisul între puterea consumată şi viteza maximă de lucru.

1.8 INTERFAȚA EXTERNĂ (SISTEMUL DE INTRĂRI ŞI IEŞIRI)

Toate microcontrolerele au un număr oarecare de intrări- Inputs / ieşiri numerice- Outputs (de la x1 la x10) organizate sub forma unor porturi I/O; conexiunile exterioare sunt bidirecționale sau unidirecționale, unele sunt multifuncționale (se oferă funcții alternative pe acelaşi pin), unele pot avea o capacitate sporită de a absorbi curent (de exemplu pentru comanda directă a unui LED, cu $I_{OL\ max}$ = - 20mA), etc.

În afară de acest set de intrări/ieşiri de uz general, pentru interfața cu mediul exterior se oferă **o serie de alte facilități importante de intrare/ieşire** cum ar fi:

a. UART (Universal Asynchronous Receiver Transmitter) este un port serial bidirecțional destinat implementării unui protocol clasic de comunicație asincron; **USART** (Universal Synchronous Asynchronous Receiver Transmitter) este un port similar, dar care permite implementarea şi a unui protocol sincron cu obținerea unor viteze mai mari de comunicație; **SCI** (Serial Communications Interface) este un circuit de tip UART îmbunătățit, definit şi utilizate de firma Freescale(Motorola). **LIN** (Local Interconnect Network) reprezintă o implementare particulară a unui protocol de comunicație asincron, utilizată în industria de automobile ca o alternativă de mică viteză dar cu preț scăzut pentru magistrala şi protocolul **CAN** (vezi mai jos).

b. **Porturi seriale sincrone dedicate** - sunt destinate transferului serial de date de mare viteză cu unele periferice specializate (sisteme de afişare, convertoare analog-numerice, etc.) sau care permit conectarea într-o rețea de comunicație. Presupun existenta, împreuna cu datele, a unui semnal de ceas (implicit sau explicit) pentru sincronizarea acestora. Implică şi implementarea unor protocoale mai mult sau mai puțin complexe de transfer al informației, fiind de fapt vorba de o magistrală serială. Există câteva implementări răspândite (sunt prezentate în ordinea crescătoare a complexității):

- **SPI** (Serial Peripheral Interface) este un port serial sincron definit de firma Motorola
- **Microwire / Microwire Plus** este o interfață serială bidirecțională sincronă definită şi utilizată de firma National Semiconductors
- **I²C** (**I**nter **I**ntegrated **C**ircuits bus) este o interfață serială bidirecțională (pe numai 2 fire), dezvoltată de Philips, destinată aplicațiilor de 8 biți. Există şi multe circuite "periferice" care sunt prevăzute cu o astfel de interfață. Este cunoscută şi sub denumirea **TWI** (Two Wire Interface)
- **CAN** (**C**ontroller **A**rea **N**etwork) proprietate intelectuală a firmei Bosch, foarte utilizat în Europa şi Japonia, oarecum similar ca funcționalitate lui **SAE J1850** care este utilizat în America de Nord (SAE -Society of Automotive Engineers), este un standard (o magistrală şi un protocol) de comunicație serială sincronă utilizat în industria de automobile, permițând interconectarea într-o rețea a diverselor componente inteligente (senzori, elemente de execuție, indicatoare, etc.) omniprezente într-un automobil modern. In ultimul timp magistrala CAN a început să fie utilizată şi în alte domenii decât industria de automobile (automatizări industriale, robotică, acționări electrice).

c. Conectivitate Ethernet/Web – implică existența unor resurse care să permită integrarea cu uşurință într-o rețea de tip Ethernet, pentru a face posibilă, în final, implementarea unui protocol TCP/IP (a unei stive TCP/IP). Resursele respective pot fi de natură software (stivă soft) care presupun o viteză de prelucrare (putere de calcul) a CPU suficient de mare pentru a nu afecta vizibil operarea propriu-zisă a controlerului, sau hardware (stivă hardware). Pe baza acestei stive se poate realiza o conectivitate tip HTTP, FTP, STMP, POP3, etc.

d. Conectivitate USB - magistrala serială USB (Universal Serial Bus) a fost creată pentru conectarea cu uşurință a diverselor periferice la un calculator PC (cu rolul de gazdă - host). Conexiunea permite si furnizarea tensiunii de alimentare. Varianta USB 1.1 permite atingerea unei rate de transfer maxime a datelor de 12Mbytes/sec, iar varianta USB 2.0 a unei rate maxime de cca. 480MBytes/sec. La ora actuală există pe piață multe firme care oferă o gamă largă de microcontrolere cu conectivitate USB (majoritatea compatibile USB 1.1), cu un preț de cost minim pentru componentele hardware şi software. Exemple în acest sens ar fi firmele: Atmel, Microchip, Intel, Cypress, ST, Infineon, s.a. Majoritatea sunt destinate realizării unor periferice USB si mai puține realizării unui USB host.

e. Conectivitate Wireless- se referă la existența unor resurse hardware si/sau software care să permită integrarea cu uşurință şi la un preț de cost avantajos într-o rețea de tip wireless, pentru a face posibilă, în final, implementarea unui protocol (a stivei aferente protocolului). Exemplele cele mai cunoscute de astfel de rețele, protocoale şi stive sunt Bluetooth (IEEE 802.15.1) şi Zigbee (IEEE 802.15.4).

f. Convertoarele Analog Numerice (CAN, ADC)

Convertoarele utilizate fac parte de regulă dintr-un sistem de achiziție de date, existând şi un **multiplexor analogic** cu mai multe canale de intrare. Rezoluția disponibilă este tipic de 8, 10 sau 12 biți, uneori cu precizia (rezoluția adevărată) corespunzătoare unui număr mai mic de biți. In marea majoritate a cazurilor ele sunt realizate pentru **mărime de intrare unipolară**. Sursa de referință utilizată este internă sau externă. Timpul minim de conversie este în plaja x μsec la x10 μsec. Există microcontrolere care utilizează tehnici de recalibrare (auto-zero, corecție câştig, etc.) pentru mărirea şi/sau menținerea preciziei.

Tehnicile de conversie cele mai utilizate sunt: aproximații succesive (majoritatea) cu eşantionare implicită (circuit Track-Hold inclus), rampă digitală (mai rar). Există şi subsisteme locale care, în cazul în când sunt prezente, pot fi folosite pentru implementarea unor alte tehnici de conversie bazate pe integrare (cu utilizarea unui număr minim de componente exterioare): numărătoare de impulsuri, circuite comparatoare (analogice, standard), intrări de captare (forțează memorarea - "captarea" valorii unui numărător care numără liber, în momentul activării, permițând astfel măsurarea intervalelor de timp sau frecvențelor), etc.

OBSERVAȚIE În ultimul timp au apărut şi variante de CAN cu rezoluții mari şi foarte mari, realizate în tehnica sigma-delta (σ-δ). Realizările respective sunt mai degrabă un CAN cu microcontroler (firma Analog Device oferă un nucleu de microcontroler 8051 plus un CAN sigma-delta cu rezoluții de până la 24 biți !)

g. Convertoarele Numeric Analogice (CNA, DAC)

Cea mai răspândită tehnică de conversie numeric analogică folosită este bazată pe modulația în factor de umplere (PWM- Pulse Width Modulation). Există unul sau mai multe canale pe care se poate genera un tren de impulsuri cu factor de umplere programabil (de la 0

la 100%). Factorul de umplere este controlat cu o rezoluție de la 8 biți sau 16 biți. Frecvența trenului de impulsuri este şi ea programabilă, în limite largi. La un microcontroler fără un sistem PWM dedicat, în acest scop se poate utiliza, cu o flexibilitate mai scăzută, sistemul de timere/numărătoare şi orice ieşire numerică. Printr-o filtrare exterioară relativ simplă, de tip trece jos (FTJ, Low Pass), se poate obține o tensiune de ieşire proporțională cu factorul de umplere. Convertoare numeric analogice propriu-zise sunt mai rar întâlnite.

g. Interfața pentru sisteme de afişare tip LCD (panou LCD)

În ultimul timp în familiile de microcontrolere deja consacrate sau în familiile noi au apărut variante care posedă un subsistem destinat conectării directe, cu utilizarea unui număr minim de componente exterioare, unui sistem de afişare (un panou) de tip LCD (cu cristale lichide). Interfața respectivă generează toate semnalele necesare pentru comanda panoului LCD. Complexitatea sa este descrisă prin numărul maxim de segmente LCD care pot fi controlate, fiind limitată în primul rând de numărul de conexiuni externe necesare (pentru un număr mare de segmente oricum se folosesc tehnici de multiplexare).

1.9 FAMILII DE MICROCONTROLERE REPREZENTATIVE

În prezentarea care urmează şi care nu se doreşte exhaustivă, accentul este pus pe familiile mai cunoscute, de 8 biți şi 16biți.

a. 8048 (Intel MCS-48 - www.intel.com)

"Bunicul" microcontrolerelor pe 8 biți, mai este încă folosit! Unele din caracteristicile sale arhitecturale se regăsesc la următoarea generație (MCS-51, 8051).

b. 8051 (Intel MCS-51 şi mulți alții: Atmel, Philips- www.semiconductor.philips.com, Infineon, Atmel, Dallas-Maxim - www.maxim-ic.com, Cygnal – www.cygnal.com , etc.)

A doua generație de microcontrolere de 8 biți a firmei Intel care, deşi apărută acum 20 de ani, încă ocupă un segment semnificativ de piață. Cu o arhitectură destul de ciudată, este suficient de puternic şi uşor de programat (odată învățat!). Arhitectura sa are spații de memorie separate pentru program şi date. Poate adresa 64KBytes memorie de program, din care primii 4(8..32)KBytes locali (ROM). Poate adresa 64KBytes memorie de date externă, adresabilă doar indirect. Are 128 (256) octeți de RAM local, plus un număr de registre speciale pentru lucrul cu periferia locală. Are facilități de prelucrare la nivel de bit (un procesor boolean, adresare pe bit). Intel a dezvoltat si un "super 8051" numit generic 80151. Actualmente există zeci de variante produse de diverşi fabricanți (Philips, Infineon, Atmel, Dallas, Temic, etc.) precum şi cantități impresionante de soft comercial sau din categoria freeware/shareware. Au apărut şi dezvoltări ale acestei familii în sensul trecerii la o arhitectură similară (în mare), dar pe organizată pe 16 biți, cu performanțe îmbunătățite ca viteză de prelucrare: familia XA51 eXtended Arhitecture de la Philips şi familia 80C251 (Intel). Din păcate aceste noi variante nu s-au bucurat nici pe departe de succesul „bătrânului" 8051.

c. 80C196 (Intel MCS-96)

Este un microcontroler pe 16 biți făcând parte din generația treia de microcontrolere a firmei Intel. Destinat inițial unor aplicații din industria de automobile, are o arhitectură von Neumann, cu un spațiu de adresare de 64KBytes, o unitate de intrări/ieşiri numerice de mare viteză (destinată inițial controlului injecției la un motor cu ardere internă), ieşiri PWM, convertor analog numeric, timer watchdog. Există multe variante, ultimele cronologic

apărute, fiind mult superioare variantei inițiale. Există şi o dezvoltare recentă sub forma familiei MCS-296 (80C296).

d. 80C186, 80C188 (Intel, AMD, ş.a.)

Derivate din clasicele 8086/88 prin includerea pe acelaşi microcircuit a 2 canale DMA, 2 numărătoare/timere, un sistem de întreruperi şi un controler pentru DRAM. Marele avantaj al acestor cvasi(aproape) microcontrolere (ele nu au memorie integrată!) este legat de utilizarea ca mediu de dezvoltare a unor platforme de calcul tip IBM-PC, compatibile 80x86, cu tot softul aferent.

e. 68HC05 (Freescale - www.freescale.com, ex Motorola)

Un microcontroler de 8 biți derivat din microprocesorul M6800 şi care prezintă multe asemănări cu un alt microprocesor răspândit, la timpul său, 6502. Are un spațiu de memorie unic (64Kbytes) în care sunt plasate şi registrele perifericelor (I/O, timere) cu un indicator de stivă (SP) hard pe 5biți (stivă de maxim 32 octeți !). Există variante cu memorie EEPROM, CAN, port serial, etc. Este unul din cele mai răspândite microcontrolere (comparabil cu 8051). Varianta evoluată a acestei familii este seria **68HC08** bazată pe o nouă unitate centrală de 8 biți numită **CPU08**, cu cea mai recentă dezvoltare sub forma seriei **68HCS08** destinată în mod special unor aplicații din industria automobilului.

f. 68HC11, 68HC12, 68HC16 (Freescale)

68HC11 a fost unul din cele mai puternice microcontrolere pe 8 biți, foarte răspândit în ciuda faptului că Motorola a fost un timp singurul producător. Are un set de instrucțiuni asemănător cu alte produse ale firmei (6801, 6805, 6809). Are un spațiu de adrese unic de 64K. Nenumărate variante ca resurse: EEPROM, OTP, CAN, PWM, etc. Prezintă ca particularitate existența unui program de încărcare rezident (bootstrap loader în ROM intern) cu care, la reset, un segment din memoria RAM externă poate fi încărcat cu cod program prin intermediul portului serial.

Variantele evoluate sunt de fapt microcontrolere de 16 biți:

- un "super 68HC11", numit **68HC12** bazat pe o nouă unitate centrală numită **CPU12**, care reprezintă extensia la 16 biți a arhitecturii HC11
- un **68HC16**, mai puțin răspândit, bazat pe o unitate centrală numită **CPU16**

g. 683xxx (Freescale)

Microcontrolere pe 32 de biți construite în jurul unui CPU analog microprocesorului M68020 (**CPU32**), denumite şi "procesoare integrate". Putere de calcul comparabilă sau mai mare ca a lui Intel 80386.

h. PIC (Microchip- www.microchip.com)

Primul microcontroler din această familie (PIC1650) a apărut acum mai bine de 20 de ani pe vremea când firma era proprietatea General Instruments. Este o familie de microcontrolere care, în ultimii ani, a cunoscut o dezvoltare explozivă. Sunt disponibile actualmente sub forma a 6 serii: PIC10, PIC12, PIC14, PIC16, PIC17 şi PIC18. In seriile respective există variante cu memorie de program de tip OTP(C) sau FLASH(F). Au fost primele microcontrolere de 8 biți cu arhitectură RISC: PIC16C5x avea un set de doar 33 instrucțiuni (Intel 8048 avea 90). Arhitectura este de tip Harvard şi, ca o particularitate, dimensiunea cuvântului pentru program este de 12, 14 sau 16 biți, cuvântul de date fiind tot de 8 biți. Există foarte multe variante pentru cele sase serii, unele din ele fiind caracterizate

printr-un număr mic de conexiuni exterioare (pini) şi în consecinţă dimensiuni mici, consum foarte mic, ideea de bază fiind costul redus.

Cronologic, ultimul produs al firmei Microchip este seria dsPIC30F, de fapt un procesor numeric de semnal, de 16 biti, cu o periferie specifică optimizată pentru controlul actionărilor electrice (motoare electrice).

Firma Ubicom (ex Scenix, www.ubicom.com) produce nişte clone ale familiei PIC, mult mai rapide decât originalele. Modulele Basic Stamp ale firmei Parallax (www.parallax.com) sunt bazate si pe astfel de microcontrolere (sunt foarte utilizate, şi nu numai de hobbistii din robotică!).

i. AVR (Atmel- www.atmel.com)

Un concurent puternic al seriei PIC este familia numită AVR, a firmei ATMEL, familie apărută în ultimii ani, care oferă variante de microcontrolere oarecum asemănătoare ca resurse cu familia PIC, la performanţe similare sau mai bune. Sunt bazate pe o arhitectură diferită, dar unitatea centrală este tot de tip RISC, cu cuvântul de date de 8 biţi. La fel ca la PIC dimensiunea cuvântului de program este mai mare, fiind de 16 biţi. Există cel puţin 3 sub familii mari, în ordinea complexităţii resurselor, acestea fiind: AT Tiny, AT90 şi ATMega.

j. COP4(00) şi **COP8**(00) (NS -National Semiconductors - www.national.com)

COP4 este un microcontroler pe 4 biţi, categorie de microcontrolere care, în general, departe de a fi învechite, ocupă un segment relativ important al pieţii. Printre caracteristici: până la 2K ROM local, 32x4 până la 160x4 RAM local, Microwire, numărătoare/timere, tensiune de alimentare 2.3-6V, număr mic de pini.

COP8 reprezintă o serie de microcontrolere pe 8 biţi, versatilă, cu preţ scăzut, disponibilă în multe variante. Arhitectura este similară lui 8051, dar setul de instrucţiuni este similar lui Z80.

k. **Z8** (Zilog, - www.zilog.com)

Un derivat al microprocesorului Z80, reprezintă un compozit al mai multor arhitecturi diferite. Nu este compatibil cu setul de instrucţiuni şi nici cu perifericele standard Z80. Are trei spaţii de adrese: program, date şi un masiv de registre. Resurse locale tipice: UART, timere, DMA, sistem de întreruperi cu până la 37 de surse. Există o variantă cu un interpreter Tiny Basic în ROM-ul local (analog 8052AH Basic de la Intel) precum şi o variantă cu resurse îmbunătăţite numită **Super-8**.

l. Z180(Zilog), **Rabbit** (Rabbit Semiconductors- www.rabbitsemiconductor.com)

Z180 -ul firmei Zilog are un CPU similar cu Z80 dar îmbunătăţit, cu resurse locale cum ar fi: management de memorie (memorie paginată de maxim 1MB), USART (numit SIO), 2 canale DMA, timere, sistem de întreruperi, eventual PIO. Instrucţiuni suplimentare faţă de setul standard Z80, printre care şi înmulţirea. Diversele sale variante **nu includ memorie locală**. Rabbit 2000 sau 3000 este un microcontroler bazat pe un nucleu de Z180, deosebit de versatil ca resurse periferice disponibile şi foarte uşor de integrat în aplicaţii. Sunt disponibile module realizate pe baza acestui microcontroler, module care adaugă şi memorie de tip ROM FLASH şi RAM. Utilizarea unui mediu de programare foarte productiv numit Dynamic C precum şi a facilitaţilor de programare şi depanare In-System au făcut ca acest microcontroler sa cunoască o răspândire destul de larga..

m. TMS370 (Texas Instruments- www.ti.com)

Microcontrolerul standard pe 8 biți al firmei TI realizat în multe variante (de ordinul zecilor), prezintă unele asemănări cu 8051 (memoria de date locală, stiva, modurile de adresare). O varietate extrem de mare a resurselor locale.

n. **80386EX** (Intel)

Un 80386 destinat aplicațiilor de tip controler. Resurse locale: I/O seriale, timere/numărătoare, DMA, optimizarea consumului, controler de întreruperi, controler pentru RAM dinamic. Nu au memorie locală.

Marele avantaj al unui astfel de microcontroler este că se poate utiliza ca platformă de dezvoltare un sistem de tip IBM PC împreună cu tot mediul de programare aferent.

o. **SC 3/4/5xx**, **Elan** (AMD- www.amd.com)

O serie de microcontrolere deosebit de performante realizate în jurul unei unități centrale de tip 386/486. Permit practic, doar prin adăugarea de memorie externă, obținerea unui sistem de calcul compatibil PC, destinat unor aplicații de control încapsulate-integrate ("embedded PC").

p. **80C16x** (Infineon, ex Siemens www.infineon.com)

Unul din microcontrolerele de 16 biți foarte utilizat în Europa. Arhitectură deosebit de performantă a CPU, de tip RISC, are diverse variante, cu resurse complexe: 80C165, 80C166, 80C167, etc.

q. **MSP430** (Texas Instruments)

Firma TI oferă şi o familie de microcontrolere de 16 biți cu arhitectura RISC, cu posibilitatea controlului compromisului **viteză de calcul/consum propriu**, destinată aplicațiilor portabile (şi nu numai), denumită MSP 430. Cu un spațiu de adresare de 64KBytes, are diverse variante de realizare a memoriei interne de program (OTP, FLASH), resurse diverse (printre care şi o interfața pentru un sistem de afişare LCD).

r. Alte familii de microcontrolere

Fujitsu Microelectronics (www.fme.fujitsu.com**)** - oferă familii deosebit de puternice de microcontrolere pe 8 biți (FMC-8), 16 biți (FMC-16) sau 32 de biți (FR). În fiecare familie există zeci de variante. Multe dintre ele sunt orientate pe aplicațiile din industria de automobile sau din electronica de consum(audio, video, electrocasnice).

Renesas (ex Hitachi, www.renesas.com) - oferă de asemenea o gamă largă de microcontrolere organizate în familii de 4, 8, 16 şi 32 de biți. Există un număr foarte mare de variante constructive în fiecare familie.

ARM (Advanced RISC Machine, www.arm.com **) -** este de fapt o unitate centrală de 32 de biți (sau de 16/32 biți) care face parte din categoria structurilor IP (“Intelectual Property”). Consorțiul ARM a oferit licențe ale acestei micro arhitecturi (nucleu ARM) pentru numeroşi producători de circuite (Atmel, Philips, TI, OKI – www.okisemi.com , etc.). Pe baza acestor licențe se realizează şi microcontrolere de mare performanță. Cele mai cunoscute şi răspândite variante de nuclee sunt ARM7 şi ARM9, cu implementările lor simplificate numite ARM7T, ARM9T (T-“Thumb”).

MPC500 (Freescale)- este o familie de microcontrolere bazată pe o unitate centrală de 32 de biți compatibilă cu arhitectura (şi cu setul de instrucțiuni) **Power PC**. Include şi o unitate de prelucrare în virgulă mobilă.

OBSERVAȚIE Codurile prezentate sunt generice, identificarea completă a unui microcontroler făcându-se şi cu utilizarea unor prefixe/sufixe alfanumerice prin intermediul cărora se precizează resursele disponibile şi eventual alte caracteristici ale variantei constructive (frecvența maximă de ceas, tipul de capsulă, gama de temperatură, etc.).

1.10 LIMBAJE DE PROGRAMARE

1.6 Limbajul maşină şi de cel de asamblare

Limbajul maşină (instrucțiunile maşină) este singura formă de reprezentare a informației pe care un microcontroler o "înțelege" (ca de altfel orice alt sistem de calcul !). 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 (o mnemonică cu operanzii aferenți) are drept corespondent o instrucțiune în limbaj maşină (excepție fac macroinstrucțiunile disponibile la unele asambloare).

Un program în limbaj de asamblare este rapid şi compact. Aceasta nu înseamnă că un astfel de program, prost scris, nu poate fi lent şi de mari dimensiuni, programatorul având controlul total (şi responsabilitatea !) pentru execuția programului şi gestiunea resurselor.

Limbajul de asamblare este primul care trebuie învățat, chiar sumar, atunci când dorim să proiectăm o aplicație hard/soft cu un anume microcontroler (familie), el permițând înțelegerea arhitecturii acestuia şi utilizarea ei eficientă.

Utilizarea numai a limbajului de asamblare pentru dezvoltarea unei aplicații complexe este neproductivă de multe ori, deoarece există şi familii de microcontrolere cu CPU de tip CISC care au un număr foarte mare de instrucțiuni (x100) combinate cu moduri de adresare numeroase şi complicate.

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

b. Interpretere

Un interpreter este o implementare a unui limbaj de nivel înalt, mai apropiat de limbajul natural. Este de fapt un program rezident care, în acest caz, rulează pe o platformă de calcul de tip microcontroler. Caracteristic pentru execuția unui program interpretat, este **citirea şi executarea secvențială a instrucțiunilor (instrucțiune cu instrucțiune)**. De fapt fiecare instrucțiune de nivel înalt este interpretată într-o secvență de instrucțiuni maşină care se execută imediat.

Cele mai răspândite interpretere sunt cele pentru limbajele BASIC şi FORTH. Limbajul BASIC este remarcabil prin simplitatea şi accesibilitatea codului, dar (în varianta interpretată) şi prin viteza mai mică de execuție, acesta fiind de altfel prețul plătit pentru utilizarea oricărui interpreter. Un exemplu de astfel de interpreter foarte răspândit şi utilizat este PBASIC al firmei Parallax utilizat pentru programarea modulelor Basic Stamp. Este foarte uşor de învățat şi poate fi utilizat suficient de productiv chiar de indivizi care au o experiența minima în domeniul programării.

Limbajul FORTH este popular datorită vitezei de execuție (apropiată de cea oferită de limbajul de asamblare) şi posibilității construirii aplicațiilor din părți reutilizabile. Este un limbaj mult diferit de limbajele clasice, codul este destul de greu de scris şi de mai ales de citit (codul este greu lizibil). Totuşi, odată stăpânit foarte bine (în timp!), poate fi foarte

productiv în aplicații cum ar fi cele de control, în robotică, etc.

Marele avantaj al utilizării unui interpreter este dezvoltarea **interactivă şi 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.

OBSERVAȚIE. Există şi variante interpretate ale limbajului C care constituie o implementare aproximativă a standardului ANSI C. Un astfel de exemplu este Interactive C (Newton Labs) care generează cod Motorola 68HC11 şi este destul de mult utilizat în robotică.

c. Compilatoare

Un compilator combină uşurința în programare oferită de un interpreter (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 maşină sau în limbaj de asamblare (urmând a fi apoi asamblat). Codul maşină rezultat are dimensiuni relativ mari (dar mai mici decât cel interpretat) şi este executat direct, ca un tot, de microcontroler. De regulă codul generat poate fi optimizat fie ca dimensiune, fie ca timp de execuție.

Se pot enumera compilatoare pentru limbajele: C, BASIC, Pascal, PL/M (Intel), Forth.

Cele mai populare şi utilizate sunt cele pentru limbajul C, un limbaj universal folosit atât pentru super computere cum ar fi Cray-ul, cât şi de microcontrolerele de 4 biți. Este un limbaj puternic şi flexibil, care deşi de nivel înalt, poate permite şi accesul direct la resursele sistemului de calcul. Un program bine scris generează un cod rapid şi compact. Totuşi, de multe ori, porțiuni critice din punct de vedere al vitezei de execuție, trebuie încă scrise în limbaj de asamblare. Există numeroase implementări, pentru majoritatea familiilor de microcontrolere. Cu anumite limitări legate de arhitectură şi mai ales resursele microcontrolerului, asigură **portabilitatea** unei aplicații scrisă pentru un anumit tip (familie) de microcontroler la un alt tip (familie).

Pentru unele familii noi şi foarte puternice de microcontrolere, datorită complexității setului de instrucțiuni şi al numeroaselor moduri de adresare, este descurajată în mod explicit utilizarea limbajului de asamblare în momentul în care se programează aplicații performante. **Unitatea centrală a acestor noi microcontrolere a fost proiectată şi optimizată pentru utilizarea unor limbaje de nivel înalt**.

Funcție şi de familia de microcontrolere în cauză, prețul unor astfel de compilatoare (de C) poate fi destul de ridicat, începând cu x100USD şi ajungând la x1000USD. Există insa şi variante freeware de compilatoare de C, cum ar fi **gcc** care este o portare a compilatorului omonim din Linux în lumea microcontrolerelor. Există implementări diferite ale acestui compilator care generează cod pentru familii diferite de microcontrolere (de exemplu AVR, MSP430, 68HC11, etc.).

OBSERVAȚII

1. Pentru a. şi c. 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 sau Motorola S) şi, nu în ultimul rând, **un simulator şi/sau depanator** ("debugger").

2. Codul astfel obținut trebuie încărcat în memoria de program a maşinii țintă unde va rula,

fiind de fapt programat într-o memorie de tip (EP)ROM/FLASH sau încărcat direct (up-loaded) într-o memorie de tip RAM.

1.11 DEZVOLTAREA ŞI TESTAREA APLICAȚIILOR

Cele mai răspândite mijloace hardware/software utilizate în dezvoltarea şi 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 -host (cum ar fi un PC). Programul se poate executa pas cu pas, conținutul variabilelor şi registrelor poate fi vizualizat şi modificat. Reprezintă un punct de plecare atunci când se abordează un microcontroler, pentru familiarizarea cu resursele lui şi cu limbajul de asamblare. Nu permite simularea în timp real a întreruperilor şi, de regulă, programul rulează mai încet decât pe maşina reală. De regulă există mijloace pentru evaluarea vitezei de execuție a codului simulat (ca număr de cicluri maşină sau de stări). Ideal, un simulator ar trebui să permită şi simularea completă a interacțiunii, cel putin din punct de vedere logic, cu toate perifericele disponibile.

b. **Programele de depanare ("debbugers") rezidente**

Sunt programe (denumite uneori în română şi programe "monitor") care rulează -sunt rezidente- pe maşina țintă (microcontrolerul) oferind facilități de depanare similare simulatorului. Interfața cu utilizatorul este realizată prin intermediul unui sistem gazdă (PC) şi/sau a unui terminal alfanumeric, conectate prin intermediul unui port serial. Utilizează o parte din resursele microcontrolerului: memorie de program pentru el însuşi (de tip ROM) şi memorie de date (RAM) pentru variabile proprii, memorie de program (de multe ori memorie externa de tip RAM, pentru a se putea incarca şi modifica cu usurinta codul!) 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 şi depanarea aplicațiilor uzuale.

c. Emulatoarele In Circuit (ICE-In Circuit Emulators**)**

Sunt cele mai eficiente mijloace de testare şi dezvoltare şi au fost mult timp cele mai complexe şi mai costisitoare. Presupun existența unui hardware dedicat (şi complicat!) care înlocuieşte practic microcontrolerul (se conectează în locul acestuia în sistemul pentru care se dezvoltă aplicația), în acelaşi timp fiind disponibile toate facilitățile descrise anterior şi altele suplimentare. Permit un control total al maşinii țintă (în timp real), fără a folosi nimic din resursele acesteia (la variantele cele mai costisitoare). Ele sunt realizate de cele mai multe ori ca un mijloc de testare şi depanare de sine stătător, conectat la un PC prin intermediul unui port paralel, serial sau USB. Cele mai ieftine sunt disponibile în gama x100$, iar cele mai scumpe x1000$.

Variantele mai noi folosesc interfețele specializate de programare şi depanare integrate la nivelul microcontrolerului, când ele există. 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 cat mai complete a sistemului şi a aplicației în ansamblu, cu alte cuvinte a microcontrolerului împreuna cu o dispozitivele hardware externe. Ele integrează de regula şi un simulator SPICE.

Cele mai cunoscute sunt Proteus VSM (Virtual System Modelling) al firmei Labcenter Electronics (http://www.labcenter.co.uk/) şi UMPS (Universal Microprocessor Program Simulator) al firmei Virtual Microdesign (www.vmdesign.com).

Un astfel de simulator permite rularea aplicației (codului), în mod continuu sau pas cu pas şi evaluarea în detaliu a modului cum aceasta (si microcontrolerul) interacționează cu hardware-ul extern. El permite ceea ce se numeşte co-simularea (Co-simulation): interacțiunea dintre software-ul microcontrolerului şi dispozitivele electronice analogice sau numerice conectate cu acesta. Sunt bazate pe utilizarea unor modele avansate ale unor familii de microcontrolere precum şi pe modelele SPICE ale dispozitivelor electronice.

In cazul lui Proteus VSM există (versiunea 6.8 SP1) modele ale microcontrolerelor: ARM (LPC2000-Philips), PIC10, PIC12, PIC16, PIC18, AVR, 8051, 68HC11 şi Basic Stamp.

In cazul UMPS există modele ale microcontrolerelor: PIC12, PIC16, 8051, 68HC05, 68HC11, ST6200 (SGS) şi COP8. Pentru fiecare model de microcontroler există un asamblor şi un editor de legături integrat astfel că se poate face, în anumite limite, şi dezvoltarea codului în asamblare. Pe lângă aceasta ele au asigurate şi 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 (la Proteus VSM există peste 6000 de modele), circuite de memorie sau periferice, în categoria dispozitivelor externe se mai pot menționa şi sisteme de afişare (LED, LCD), tastaturi matriciale sau butoane, relee, etc.

Ambele simulatoare menționate sunt produse comerciale, dar există şi versiuni de evaluare, utilizabile cu limitările de rigoare.

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

Pe piața de software pentru microcontrolere exista şi componente numite nuclee de timp real sau sisteme de operare în timp real (RTOS). Un astfel program de sistem de nivel profesional este o componenta software scumpă sau foarte scumpă, funcție de complexitatea lui, de accesibilitatea surselor programului, de familia de microprocesoare căreia ii este adresat, de modul in care va fi distribuit împreuna cu aplicația. Există insă şi variante de RTOS, de mai mică complexitate, din categoria freeware sau shareware, care pot fi utilizate cu performante mulțumitoare.

Un sistem de operare în timp real facilitează crearea aplicațiilor aşa zise de timp real, dar nu garantează şi faptul ca 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 şi 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 uşurate deoarece comportarea sistemului poate specificată complet încă din faza de proiectare. Din aceasta cauza, în cazul multora din aplicații, multe probleme se pot rezolva în timp real şi fără să se utilizeze un RTOS. Esențială este pană la urmă calitatea şi competența celui care programează aplicația !

1.12 CRITERII DE ALEGERE A UNUI MICROCONTROLER

În momentul în care se doreşte alegerea unui microcontroler (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 şi 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 microcontroler?
Care sunt costurile permise pentru mediul de programare şi dezvoltare?

b. Timpul de dezvoltare al aplicației

Ce limbaj de programare să aleg?
Ce limbaje de programare cunosc bine şi ce medii de dezvoltare am utilizat?
Ce modalitate de testare şi 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 şi respectiv date?
Va fi necesară si o memorie externă?
Ce fel de alimentare este disponibilă şi care sunt limitările acesteia?
De câte intrări şi/sau ieşiri am nevoie?
Ce fel de intrări şi/sau ieşiri sunt necesare: intrări/ieşiri analogice, ieşiri numerice de curent mai mare?
Care sunt resursele necesare în materie de temporizare/numărare si care ar fi caracteristicile lor cele mai importante (rezoluție, frecventa maxima de numărare) ?
Ce tip de capsulă, ce dimensiuni fizice şi 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 şi 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 şi dezvoltarea ulterioară

Cu ce tipuri de circuite se poate interfața cât mai simplu: sisteme de afişare, senzori, elemente de comandă şi 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 şi cât sunt de accesibili pentru mine?
Care este suportul oferit de fabricant sau distribuitor şicare este baza de cunostințe existentă: site-uri web, documentație on-line sau pe CD-uri, note de aplicații, exemple de proiectare (reference designs), software din categoria freeware/shareware si, 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 şi cele de timp de dezvoltare. Principial, utilizarea unui limbaj de nivel înalt împreună cu un emulator pentru testare şi depanare poate duce la scurtarea consistentă a timpului de dezvoltare. Dar prețul unui compilator este întotdeauna mai mare decât cel al unui asamblor (nimic nu e mai ieftin decât ceva ce poate fi gratis!), iar prețul unui emulator este şi el mai mare decât cel al unor mijloace mai simple de testare şi depanare.

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