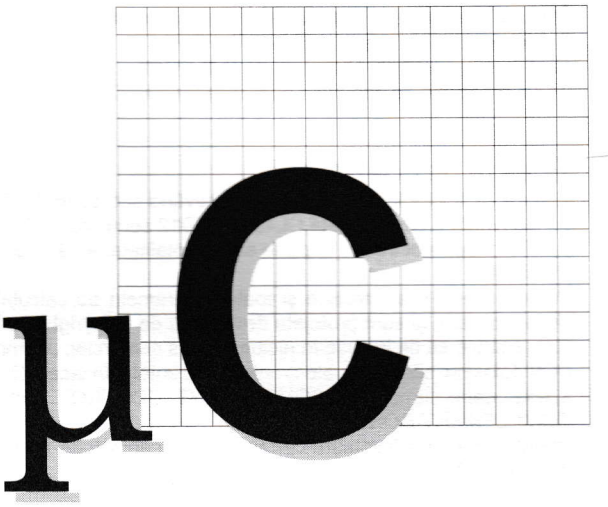


ÎNVAŢĂ HARDWARE FIRMWARE ŞI SOFTWARE DESIGN

TRADUCERE ŞI ADAPTARE PENTRU LIMBA ROMÂNĂ DE

O G POPA

DUPĂ VERSIUNEA ÎN LIMBA ENGLEZĂ (EDIŢIA A 5-A RV0110)

LEARN HARDWARE FIRMWARE AND SOFTWARE DESIGN

O G POPA

EDITURA
COMPLEMENT CONTROL
http://www.corollarytheorems.com

CUPRINS

DESPRE ACTIVITATEA DE R&D

Acest capitol este mai mult *o introducere de ansamblu*, generală, referitoare la subiectele majore prezentate în cartea de față. Acum, cu toate că cei mai mulți dintre cititori consideră (în general) că introducerile sunt plictisitoare și chiar inutile, eu vă sugerez totuși să citiți și să analizați acest capitol cu multă atenție. Abrevierea „R&D" [Research and Development] se traduce prin „Cercetare și Dezvoltare": ea se referă la creația de produse noi, avansate din punct de vedere tehnic, în civilizația noastră, într-o țară, într-o companie, sau pentru o persoană—de exemplu, pentru dumneavoastră.

Activitatea de R&D este cea mai importantă comparativ cu orice altă activitate în civilizația noastră; ca rezultat, cele mai multe națiuni alocă fonduri impresionante pentru dezvoltarea ei. Din nefericire însă, indiferent de valoarea fondurilor alocate, ele nu sunt niciodată suficiente, și sunt deseori direcționate greșit. În principiu, se încearcă încurajarea *celor mai inteligente persoane* să creeze, numai că noțiunea abstractă *„inteligență"* este, în zilele astea, numai un termen general, teoretic, fără prea multe legături cu realitatea de pe stradă. Starea de *inteligență avansată* este o apariție cu totul întâmplătoare, accidentală, printre membrii speciei umane, perfect independentă de gene, rasă, naționalitate, și chiar de educație—cea oficială. Lucrurile stau cam așa: noi fie avem chestia asta numită inteligență, sau nu. Educația (oficială) este obligatorie pentru a se facilita *maturizarea inteligenței*, însă ea nu poate să înlocuiască inteligența.

NOTĂ Specific activității de HFS design este faptul că s-a întâmplat de multe ori că au apărut programatori de excepție din rândul persoanelor care nu aveau nici un fel de vocație tehnică, anterior; de exemplu, muzicieni, doctori, funcționari. Acest lucru se datorează faptului că programarea calculatoarelor este de fapt un exercițiu logic. Pentru a deveni un programator bun, trebuie să vă placă programarea înainte de orice. Însă, dacă vă place programarea, aceasta înseamnă că logica dumneavoastră personală și-a descoperit un câmp nou pentru expansiune, și că ea este suficient de coaptă pentru activitatea (logică) de programare.

Ideea de ansamblu este, societatea trebuie să creeze, să îmbunătățească, și să faciliteze *mijloacele* de care au nevoie persoanele fizice pentru a-și dezvolta puterea lor de creație naturală. Atenție vă rog: la nivel de societate, noi trebuie să încurajăm cetățenii, și companiile foarte mici, nu firmele monstruoase sau instituțiile guvernamentale mastodont—asta numai dacă vrem să avem cu adevărat progres social în viitor. În acest sens, cel mai bun exemplu este USA: în această națiune de excepție, orice persoană are (și a avut) mijloacele necesare, nemaipomenit de ieftine, pentru a inventa produse noi. Vedeți dumneavoastră, pentru a se încuraja creația de invenții, și implicit progresul social, indivizii (printre care există inteligența nativă, naturală) trebuie să aibă acces ușor și ieftin la următoarele *mijloace ale dezvoltării*.

1 ***În primul rând avem nevoie de cărți bune de specialitate.*** *Acest aspect este cel mai important fiindcă, din nefericire, publicarea, promovarea, și distribuția cărților bune a devenit aproape imposibilă în ziua de azi. Lozincile propagandiste de tip „libertate de expresie" și „competiție liberă" sunt numai concepte teoretice pe hârtie, azi, în perfectă opoziție cu realitatea sângeroasă de pe piață.*

Industria cărților, în particular, este păzită cu gelozie de marile edituri împotriva oricăror intruși potențiali. La rândul lor, editurile mari (dar și toată media în general) sunt ținute zdravăn în ghearele marilor capitaluri, iar aceast lucru nu poate ajuta pe nimeni să realizeze ceva bun, și cu atât mai puțin progresul social.

2 ***Distribuţia de componente şi instrumente electronice ieftine.*** *Exceptând America de Nord, Europa de Vest, Australia, şi câteva zone prin Asia de Est, distribuţia de componente şi instrumente pentru dezvoltarea HFS nu este numai foarte dificilă: ea este şi prohibitiv de scumpă. Acest aspect trebuie adresat cât se poate de repede în fiecare ţară care doreşte să atingă un nivel de dezvoltare mai avansat.*

3 ***Compilatorele pentru HFS design trebuie să fie mai ieftine.*** *Există multe compilatoare excepţionale care se vând cu preţuri între 5000 şi 100000 USD, în ciuda faptului ca ele au numai câteva sute de MB cu totul [aceasta pentru că ele folosesc librării de tip *.dll dezvoltate de alte companii, aşa cum este Microsoft]. Totuşi, în multe instanţe preţul lor este pur şi simplu jaf la drumul mare.*

Pentru a forţa preţurile compilatoarelor să scadă, guvernele, companiile, şi chiar dezvoltatorii individuali ar trebui să-şi unească eforturile în crearea de compilatoare ieftine--aceasta nu este foarte dificil. În plus, industria de software are nevoie de puţină standardizare bazată pe limbajului C, numai. Singurul lucru necesar, în plus, este, limbajul C trebuie îmbogăţit cu o interfaţă grafică standard. Din nefericire, (Visual) C++ nu este suficient de bun fiindcă el a devenit o aberaţie OOP [Object Oriented Programming] pretenţioasă şi ultra-exagerată.

Pasul următor, în viitor, ar fi standardizarea sistemului de operare. Noul sistem de operare (numit, să zicem, „Standard OS" [SOS]) trebuie să fie numai scheletul unui sistem de operare, fără nici un fel de grafice decorative, însă complet funcţional.

Ca o notă aici, limbajul de programare C este cel mai rapid, şi cel mai puternic; el este superior tuturor celorlalte limbaje, inclusiv Assembler, C++, Java, Delphi, şi Visual Basic. În plus, C are avantajul că este cel mai simplu şi cel mai logic limbaj de programare dezvoltat vreodată, aşa că se potriveşte perfect pentru a deveni o disciplină obligatorie, standard, de programare în toate cursurile tehnice. Din nou, C are nevoie numai de o interfaţă grafică bună similară celei din Visual Basic 6.

Ieftinătatea mijoacelelor (componente electronice, instrumente, şi compilatoare) pentru activitatea de R&D în HFS reprezintă motivul pentru care USA este azi cea mai avansată naţiune din lume, însă multe alte naţiuni ar putea repeta această experienţă cu succes. Trebuie de-asemenea remarcat şi faptul că, sub presiunea rezervelor de ţiţei care se diminuează, civilizaţia noastră trebuie să încline balanţa energetică în favaoarea producerii şi utilizării energiei electrice. Acest aspect înseamnă că, în curând, vor trebui să fie implementate mii şi zeci de mii de *noi aplicaţii de control folosind microcontrolere*.

DEFINIŢII

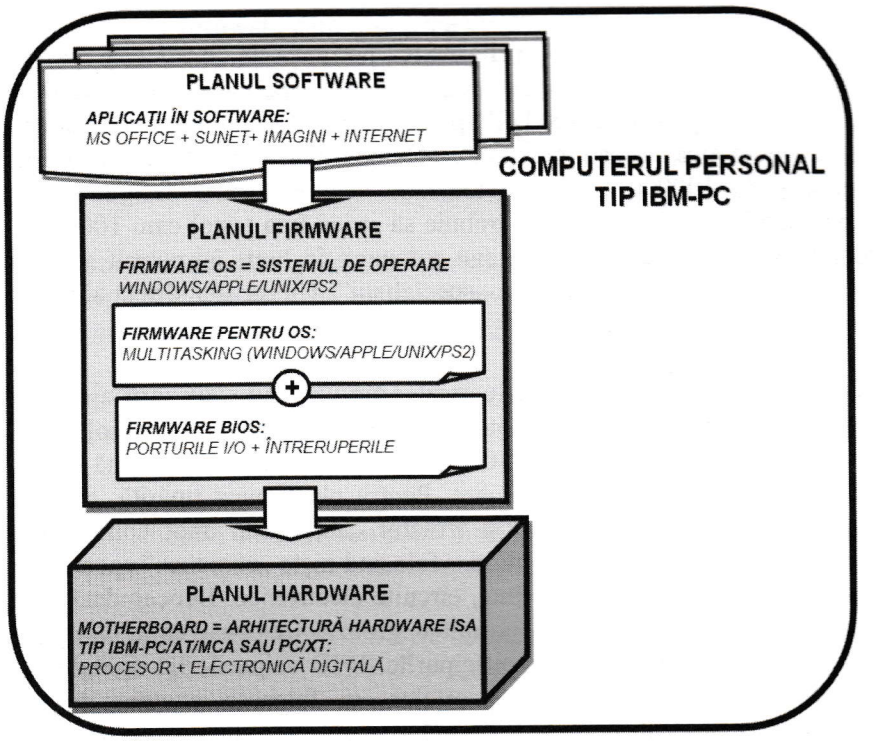
Această secţie explică câţiva termeni de bază fiindcă ei sunt folosiţi cam aiurea prin mai toată industria IT. Chiar şi „la mama lor", în America de Nord, numai puţine persoane ştiu ce înseamnă cu adevărat aceşti termeni. Pentru cetăţeanul obişnuit (vorbitor de engleză), majoritatea termenilor folosiţi în industria IT sunt mai mult decât criptici: ei sunt un fel de ghicitori greu de desluşit!

DEFINIŢIE: *„Hardware" este un termen general care numeşte componentele materialele, sau instrumentele şi materialele fizice folosite la întreţinere, în construcţie, în design, sau în activităţile de birou. În electronică, „hardware" se referă strict la componentele electronice fizice.*

În limba engleză, oamenii folosesc în mod obişnuit noţiunile *„home hardware", „maintenance hardware",*

„construction hardware", *„office hardware"*, *„computer hardware"*, *„electronic hardware"* şi multe altele. Trebuie remarcat că domeniul de aplicaţie al (înţelesului) cuvântului *„hardware"* este nelimitat fiindcă el se refera la „orice fel de componente fizice" în general. Din nefericire, în electronica digitală termenul *„hardware"* creează chiar şi mai multă confuzie, în mod special atunci când el este acompaniat de cuvântul „design".

Fig I1: interacţiunea planurilor de hardware, firmware, şi software într-un PC



Iniţial, noţiunea de *„hardware"* denumea numai *componentele electronice digitale*. Se întâmplă însă că procesul de *hardware design* are o particularitate: el este aproape identic în funcţionalitate cu *firmware design*, în ciuda faptului că ele sunt două domenii diferite. Evoluţia istorică a noţiunii *„hardware"* a fost (aproximativ) după cum urmează. Prima oară au apărut *circuitele electronice de tip analog*; pasul următor au fost *circuitele electronice digitale* (logice); la un anumit punct în timp au apărut componentele programabile folosind *„software"*. În curând, noţiunea de *„software"* a fost despărţită în două categorii: *„firmware"* şi *„software"*; mai departe, oamenii au remarcat că *firmware* este identic în funcţionalitate cu electronica digitală, aşa că ei au re-denumit electronica digitală *„hardware"*. În momentul în care aplicaţiile digitale au început se se „coacă", *hardware*, *firmware*, şi *software* au devenit toate părţi integrale constituente în dezvoltarea *aplicaţiilor digitale (programabile)*.

Necazul este că lucrurile nu s-au oprit acolo. Aproximativ cu 20 de ani în urmă, oamenii au inventat câteva limbaje de programare speciale, aşa cum sunt *HDL* [Hardware Descriptive Language], *Verilog*, *CUPL* [Cornell University Programming Language], plus compilatorele necesare, şi chiar şi nişte maşini extraordinar de complexe care fabricau hardware direct din fişierele HDL: toate aceste sisteme au re-definit întelesul cuvântului hardware. De data asta, lucrurile stau cam aşa: se foloseşte un compilator special (tip CUPL, Verilog, sau HDL) pentru a se scrie *codul de hardware*. Mai departe, codul de hardware este testat virtual, apoi programul compilat este trimis la un set de maşini care vor fabrica un IC [Integrated Circuit] special numit ASIC [Application Specific Integrated Circuit].

Fac o paranteză aici fiindcă trebuie menţionat un alt domeniu paralel foarte important în hardware design, şi anume *programarea hardware folosind PGA/FPGA* [Programmable Gate Arrays/Field Programmable Gate

Arrays]. In principiu, acestea sunt IC-uri speciale care pot fi programate folosind un limbaj de programare hardware (HDL, Verilog, CUPL), iar programul compilat este tradus în configuraţii electronice (digitale, logice); cu alte cuvinte, în *configuraţii de hardware*. Cu toate că acest proces este similar cu programarea controlerelor folosind firmware, diferenţa remarcabilă este, un modul logic programat pe un PGA este executat de zeci sau chiar de sute de ori mai repede decât acelaşi modul programat în *firmware* pe un controler. Aceasta a fost cândva *programarea hardware adevărată*, numai că ea a pierdut teren odată cu apariţia *tehnologiei ASIC*.

Atunci când ne referim la IC-urile ASIC, notiunea de „*hardware designer*" numeşte o persoană capabilă să proiecteze cipuri ASIC. Acestea sunt cele mai bine plătite slujbe din lumea electronicii [un ASIC designer bun poate câştiga uşor 250K USD pe an în mână!]. Din nefericire, să devii un hardware designer, aşa cum a fost explicat aici, nu este pentru toata lumea. Unele din compilatoare HDL menţionate sunt foarte scumpe [ajung şi la 100000 USD] cu totul de neatins pentru designerii obişnuiţi. În plus, pentru a „câştiga experienţă", un ASIC designer trebuie să strice (via teste) cam 10000000 USD (asta este, zece milioane) în costuri de producţie. Numai câteva companii în lume îşi permit aceste tehnologii scumpe, şi acolo este şi locul unde sunt formaţi cei aleşi: „specialiştii"! Un rezultat direct al acestei tehnologii avansate este telefonul mobil pe care-l avem azi.

Poate nu o să credeţi asta, dar lucrurile, din nou, nu s-au oprit aici în hardware design; ele continuă să se dezvolte foarte mult. Pasul următor a fost SOC [System on a Chip], după care a apărut „*printed electronics*" «electronica tipărită». În cazul SOC, un singur IC poate să conţină azi o întreagă placă PCB cu zeci sau sute de microcontrolere/microprocesoare. Pentru electronica tipărită, circuitele de hardware microscopice sunt impriate pe orice suport solid şi izolator cu ajutorul unor compilatoare speciale—cu totul imposibil de procurat pentru designerii obişnuiţi—folosind nişte „cerneluri" speciale care pot forma rezistoare, capacitori, şi semiconductori: cu alte cuvinte, circuite electronice. Deocamdată lucrurile sunt destul de simple, încă în stadiul experimental, dar ele promit o groază. Cea mai comună aplicaţie în electronica tipărită, care se foloseşte deja în ziua de azi, sunt cipurile RFID «cipurile ID de radiofrecvenţă»: ele sunt atât de mici încât sunt incluse în paginile paşapoartelor, şi folosesc energia de care au nevoie direct din undele electromagnetice radio care le scanează—acestă aplicaţie este numai începutul acestei tehnologii.

Merită menţionat aici şi generaţia următoare de semiconductori: *semiconductorii organici!* Eficienţa noii generaţii de semiconductori [tranzistori şi leduri de tip OLED] este oriunde între de trei ori până la de zece ori mai bună decât ceea ce avem noi azi folosind siliconul—poate chiar şi mai mult de atât. Aceste semiconductoare organice aduc o nouă revoluţie în electronica digitală, şi ele sunt aici, acum!

DEFINIŢIE: *„Firmware" este un termen general care numeşte primul nivel de cod programat care este capabil să acţioneze direct toate „porturile I/O", „întreruperile", precum şi toate „registrele de sistem" ale unui procesor/microcontroler.*

DEFINIŢIE: *„Software" este un termen general care numeşte nivelul al doilea, şi celelalte următoare, de cod programat, şi care este situat „deasupra" codului de firmware; cu alte cuvinte, software poate accesa numai codul de firmware, nu şi nivelul hardware, în mod direct.*

Acum hardware, firmware, şi software lucrează împreună, şi ele sunt foarte asemănătoare. De exemplu, Computerul Personal [tip IBM-PC] are o placă PCB fizică numită în mod obişnuit „*motherboard*" «placa-mamă»: ea reprezintă partea (principală) de *hardware*. Trebuie remarcat că partea de *hardware* a PC-ului este (aproape) la fel pentru toate computerele fiindcă toate au o arhitectură de hardware asemănătoare bazată pe modelul standard ISA [Idustry Standard Architecture]: aceasta permite unui program de software (de exemplu, MS Office®) să lucreze la fel pe toate tipurile de PC. Totuşi, există mici diferenţe în hardware cauzate de faptul că există mai mulţi fabricanţi mari care luptă (fără scrupule) pentru acapararea pieţii. În consecinţă, pentru ca un program de software (de exemplu, MS Office) să lucreze la fel pe toate tipurile de PC-uri avem nevoie de programe de firmware scrise special ca să rezolve micile diferenţe în hardware:

aceste programe de firmware sunt numite BIOS [Basic Input/Output Operating System].

Domeniul PC-ului (computerul personal) este grozav de fertil pentru aplicații, dar asta nu este totul. La fel de importante, sau poate chiar și mai importante, sunt *aplicațiile industriale/comerciale de control*. Hai să ne uitam puțin la exemplul nostru, proiectul de HFS design numit LHFSD. Partea de *hardware* este reprezentată de placa PCB [numită LHFSD-HCK] care conține, printre altele, microcontrolerul dsPIC30F4011. Arhitectura plăcii LHFSD-HCK este proiectată special ca să rezolve funcțiile pe care le dorim noi să le controlăm. Pasul următor este să scriem codul de firmware care o să „dea viață" plăcii de *hardware* LHFSD-HCK; cu alte cuvinte, codul de firmware va controla direct toate *porturile I/O*, *registrele de sistem*, precum și *întreruperile*. Mai departe, o să scriem un program de software, folosind computerul personal, care ne permite să controlăm atât partea de firmware cât și cea de hardware folosind, de data asta, toată puterea de calcul oferită de PC, inclusiv grafica lui excepțională, comunicațiile, și chiar și accesul la Internet. Sună destul de interesant, nu-i așa?

DEFINIȚIE: *„Codul binar" este un „sistem matematic" folosit în circuitele digitale logice de hardware, dar și în programare. În codul binar, poziția fiecărui bit reprezintă valoarea exponentului numărului „2". De exemplu, numărul binar „1011 0011" înseamnă „$2^7+0^6+2^5+2^4+0^3+0^2+2^1+2^0$" în sistem decimal.*

DEFINIȚIE: *„Bitul" este unitatea de bază în circuitele logice, și în programare. Un bit poate lua numai o valoare din două posibile, „0" sau „1", care sunt interpretate logic ca adevărat/fals (sau activ/inactiv, sau închis/deschis etc).*

Trebuie remarcat că un bit logic este perfect sinonim cu un întrerupător fizic aflat în una din stările „închis" sau „deschis".

DEFINIȚIE: *„Byte-ul" este o unitate folosită în programare. Un byte conține 8 biți, așa că poate exprima 256 numere: 1 byte = 1111 1111= $2^7+2^6+2^5+2^4$ + $2^3+2^2+2^1+2^0$ =128+64+32+16 + 8+4+2+1=255 (plus numărul „0"). Unitatea „char" folosită în programare este un byte.*

DEFINIȚIE: *„Integer" este o unitate folosită în programare. Un număr de tip integer conține 2 byți, sau 16 biți, așa că poate exprima 65536 numere: 1 integer = 2 byți = 16 biți. Byții componenți într-un număr integer sunt grupați ordonat în „byte-ul inferior" «low byte» și „byte-ul superior" «high byte".*

Trebuie remarcat că datele de tip integer sunt deseori interpretate diferit (din lipsă de standardizare); de exemplu, există și unități integer de 32 de biți. În contexte particulare, numerele de tip integer mai sunt numite și „*word*" «cuvânt».

DEFINIȚIE: *„Array" «vector» este o construcție specifică folosită în HFS care conține un set de elemente identice. Elementele unui array pot fi de tip bit, byte, integer, sau orice altceva; singura condiție este ca toate elementele să fie identice. O altă caracteristică a construcției array este faptul că fiecare element într-un array este accesat folosind un „index". De exemplu, elementul 4 dintr-un array de 10 elemente numit „data[]" este data[3]: „3" este indexul folosit pentru accesarea elementului 4 din data[].*

DEFINIȚIE: *„Structure" «structură» este o construcție specifică folosită în programare care conține un set de elemente diferite; condiția este, fiecare element are un nume propriu. Elementele într-o structură pot fi de tip bit, byte, integer, sau orice altceva. Fiecare element într-o structură este accesat folosind numele lui. De exemplu, elementul „elem5" din structura „set10" este accesat folosind: „set10.elem5".*

Cam asta e. Noțiunile de bază prezentate sumar aici ar trebui să fie suficiente pentru ca începătorii să înțeleagă această carte. Detalii suplimentare ajutătoare vor fi prezentate ori de câte ori aceasta va fi necesar în paginile următoare.

CONSIDERAŢII DESPRE ACTIVITATEA DE CERCETARE ÎN HFS

Aşa cum a fost menţionat, o parte importantă din activitatea de cercetare este deja focusată spre producţia diversificată de electricitate în viitor, dar şi spre folosirea ei în diferite moduri la consumator [de exemplu, în transport]. Chiar mai mult, există o tendinţă serioasă de a se produce electicitate chiar la consumator [de exemplu, case care au acoperişul din elemente fotovoltaice]. Acest aspect, şi multe altele similare, necesită *foarte multe aplicaţii noi de control în viitor*. Cu alte cuvinţe, dezvoltarea aplicaţiilor de HFS folosind microcontrolere are un viitor glorios, strălucitor!

Din păcate, exista şi multe aspecte negative pe care proiectanţii în devenire, şi micile companii care doresc să înceapă dezvoltarea HFS, trebuie să le cunoască. Tendinţa cea mai alarmantă este generată de *complexitatea în continuă creştere a tehnologiilor IT*. Există numai o mână de fabricanţi mari de procesoare şi microcontrolere în lume, iar ţelul lor principal este, natural, să genereze cât mai mult profit. Pentru a-şi atinge scopul, fabricanţii schimbă produsele lor (atât cele de hardware cât şi programele de software) cât pot ei de repede motivând nevoia de „îmbunătăţire". În realitate, ei încearcă să mascheze tendinţa lor permanentă de a creşte preţurile. Pentru noi, această activitate obscură de „îmbunătăţire" se traduce prin complexitate crescândă adăugată zi de zi la produlele de HFS care sunt deja mult prea complicate pentru începători.

Aspectul complexităţii crescânde a determinat şi cartea de faţă să apară în 5 ediţii în numai patru ani de la lansare, pentru a putea ţine pasul cu şirul nesfârşit de schimbări din industria IT. Acesta creează necazuri mari pentru noi, scriitorii, dar generează şi mai multe probleme pentru dumneavoastră, cititorii. Cu fiecare zi care trece devine din ce în ce mai greu pentru începători să „prindă din urmă" HFS design. Sfatul meu este, în momentul în care vă decideţi să începeţi activitatea dumneavoastră de HFS design, ar trebui să „îngheţaţi" compilatoarele pe care le cumpăraţi şi să nu le mai *actualizaţi* «upgrade» decât dacă nu mai aveţi încotro.

NOTĂ Eu am folosit un PC care rula Windows 98 SE pe post de „master" [PC-ul „master" avea ca „slaves" alte PC-uri care rulau Windows 2000 şi XP] din 1998 până în 2007. L-am înlocuit numai fiindcă nu am avut încotro, dar vă mărturisesc sincer că am fost incomparabil mai mulţumit de funcţionalitatea lui decât sunt azi cu actualul „master" care rulează Vista.

Aşa cum este prezentat în această carte, activitatea de HFS design necesită câteva componente obligatorii: „MPLB IDE", „compilatorul C pentru controlerele Microchip", şi „Visual Basic 6". Încercaţi să folosiţi numai versiunile iniţiale ale lui MPLB IDE, ale compilatorului C, şi Visual Basic 6, indiferent de orice actualizări «upgrades», îmbunătăţiri, sau apariţii de compilatoare noi: aceasta înseamnă „îngheţarea" compilatoarelor pentru dezvoltarea HFS.

Eu presupun că veţi putea continua să folosiţi compilatoarelor de HFS iniţiale cam zece ani. Important este să nu vă lăsaţi tentat să „actualizaţi" «upgrade» instrumentele sau programele folosite fiindcă nu o să mai terminaţi niciodată cu asta.

Un domeniu de cercetare extrem de interesant, cu adevărat revoluţionar, sunt *nanotehnologiile*. Lucrurile stau cam aşa: în plus faţă de avantajul miniaturizării, *la nivelul nano*, *efectele fizice sunt diferite*, uneori chiar total diferite faţă de ceea ce ştim noi. De exemplu, *nano-tuburile de carbon* «carbon nano-tubes» sunt superconductive la temperaturi obişnuite! Remarcaţi că superconductivitatea este unul din visurile de aur ale civilizaţiei noastre—şi care este pe cale să devină o realitate implementată în aplicaţii technice revoluţionare cât de curând. Alte tehnologii însă, la fel de importante, sau poate chiar mai importante, ne aşteaptă imediat după colţ—colţul se referă aici numai la modul în care înţelegem/interpretăm noi azi legile fizicii.

Al doilea aspect important este generat de *Revoluţia Calculatorului* actuală. Aşa cum a fost menţionat, activitatea de HFS design pare că este „revoluţionară" azi, însă viitorul poate să fie mult mai grozav. În realitate, numai instrumentele şi compilatoarele pe care le folosim se schimbă întrucâtva revoluţionar, însă nu şi *logica de HFS*, şi nici chiar *principiile de funcţionare ale componentelor/modulelor de HFS*. Adevărul este, hardware, firmware, şi software sunt toate *aplicaţii digitale logice*. Aceasta înseamnă că ele lucrează numai cu două stări: „zero" şi „unu". Indiferent de cât de sofisticate sunt, sau ar putea deveni în viitor componentele procesului de HFS design, stările lor digitale vor rămâne la fel de simple; din nou, numai „zero" şi „unu". Aceasta este frumuseţea adevarată în electonica digitală: simplitate! Cei drept, mai există ceva foarte important în afară de „zero" şi „unu", *inteligenţa* (sau logica), dar o să discutăm despre asta la timpul potrivit (în partea de firmware design).

În concluzie, tendinţa actuală în cercetare, peste tot în lume, motivează azi toate eforturile de studiu în domeniul HFS design: acesta ESTE viitorul!

CONSIDERAŢII DESPRE ACTIVITATEA DE DEZVOLTARE ÎN HFS

Aspectul *dezvoltării* «development» în HFS este legat, în parte, de „*cum să începem*" dezvoltarea aplicaţiilor de HFS. Acum, Microchip este numai o companie, nici măcar cea mai mare din cele existente. Competiţia în domeniul microcontrolerelor include câteva nume care au rezonante remarcabile: *Texas Instruments*®, *Motorola*®, *ATMEL*®, *Cypress*®, *Intel*®, *ST Microelectronics*®, *Samsung*®, *Philips*® etc. Este normal ca unii cititori să-şi pună întrebarea, „***Care este cea mai bună companie pentru a începe cu HFS design?***"

Cel mai important factor care trebuie luat în consideraţie, înainte de a începe HFS design, este *documentaţia produselor* (de fapt, specificaţiile lor DS). Problema este, documentaţia care însoţeşte cele mai multe produse de *hardware* este atât de criptică de parcă ar fi fost scrisă pentru numai un cerc intim de specialişti—care, apropo, cunosc deja acele produse foarte bine, aşa că ei nu au nevoie de respectiva documentaţie. Cu toate că eu am lucrat câţiva ani buni în HFS design, am fost forţat de câteva ori să abandonez dezvoltarea cu unele microcontrolere pe care le dorem fiindcă documentaţia lor însoţitoare era incompletă—din punctul meu de vedere.

Este posibil ca această situaţie să fi fost creată artificial, în mod intenţionat, fiindcă fabricanţii ne sugerează să cumpărăm fel de fel de *kituri* scumpe, construite special, şi care includ, într-adevăr, toată documentaţia necesară pentru a învăţa fiecare microcontroler în parte. Chiar mai rău, fabricanţii vând *cursuri de HFS design* pentru fiecare microcontroler în parte; cu toate că aceste cursuri nu sunt foarte scumpe, este nevoie de mai multe din ele pentru a se reţine cât de cât o idee două folositoare. Pe ansamblu, documentaţia produselor Microchip pare să fie cea mai bună din toate pentru a începe HFS design. Pe de altă parte, aici intervine şi cartea de faţă: ea prezintă întregul proces de HFS design folosind microcontrolere Microchip de la „A" la „Z"!

Al treilea aspect important este *disponibilitatea componentelor electronice*, a microcontrolerelor, şi a instrumentelor/compilatoarelor necesare dezvoltării—aceasta este o problemă serioasă în unele părţi ale globului. Bineînţeles că oricine poate folosi Internetul pentru a comanda piese direct din USA de la Digi-Key®, sau de la alţi distribuitori nord-americani, numai că *taxele de transport* «shipping taxes» sunt prohibitive—ele sunt chiar mai mari decât valoarea comenzii în sine. În concluzie, înainte de orice trebuie să studiaţi posibilitatea de a vă procura componente electronice ieftine (atât microcontrolere, cât şi toate celelalte instrumente/compilatoare necesare).

În final, nu uitaţi că dezvltarea HFS cere *timp*, *bani*, şi *foarte mult studiu*. Pentru a minimaliza variabila „timp" aveţi nevoie de documentaţie bună: această carte şi, posibil, câteva în plus. Aspectul costurilor

aferente dezvoltării poate fi uşor controlat dacă formaţi *o echipă de designeri*, ceea ce duce în mod natural la diviziunea costurilor. Remarcaţi următoarele: costurile necesare dezvoltării HFS sunt o „investiţie" în cariera dumneavoastră viitoare, sau într-o posibilă companie/firmă care vă aparţine—mai mult ca sigur, acesta poate fi cea mai bună investiţie posibilă! Instrumentele şi compilatoarele sugerate în această carte sunt cele mai ieftine dintre cele de tip profesional. Acest aspect înseamnă că, odată ce aţi trecut de faza de *începător*, puteţi dezvolta aplicaţii profesionale folosind exact aceleaşi instrumente/compilatoare folosite în timpul studiului. Nu pierdeţi niciodată din vedere, în timpul studiului, *scopul/ţelul final*, fiindcă acest ţel final este singurul important.

Acum, *timpul dedicat studiului* pe care trebuie să-l alocaţi nu este o glumă, Adevărul este, învăţatul este cea mai dificilă activitate umană din toate. Pe de altă parte, *calitatea învăţatului* este o caracteristică personală: fie o aveţi, fie sunteţi capabil s-o aveţi, sau pur şi simplu nu o aveţi. Singura modalitate prin care această carte vă putea ajuta să reduceţi timpul de studiu este ca ea să fie uşor de citit. În consecinţă, *tonul* cărţii de faţă este în mod intenţionat ceva mai personal, relaxant, şi chiar puţin comic—sper. În plus, în ciuda faptului că eu prezint aici o cantitate enormă de informaţii noi pentru dumneavoastră, cartea de faţă trebuie să rămână, cumva, esenţa simplităţii.

Vedeţi dumneavoastră, fiecare din domeniile de bază (hardware, firmware, software) necesită cel puţin o carte de trei ori mai groasă decât cea de faţă, şi asta numai pentru a *zgâria suprafaţa puţin*. Să deveniţi un designer matur de hardware, plus unul de firmware, plus unul de software, aceasta nu este de loc o joacă de copii. Oamenii pierd mulţi ani din viaţa lor scurtă, preţioasă, pentru a asimila ceva din HFS design. Totuşi, cu ajutorul cărţii de faţă dumneavoastră puteţi s-o faceţi mult mai rapid—fără discuţie—însă vă rog să nu vă aşteptaţi chiar la „totul": mai trebuie să veniţi şi dumneavoastră cu „câte ceva".

NOTĂ Din când în când apare câte un cititor mai aparte care nu poate înţelege această carte teoretic. În acele câteva instanţe particulare, este evident faptul că cititorul respectiv are nevoie de practică, dar şi de mai mult studiu. Aspectul practicii în domeniul HFS design necesită, din păcate, investiţii serioase, la fel ca şi studiul: aceasta situaţie creează tensiune.

Într-un caz deosebit, un cititor se plângea că nu am explicat cum trebuie „protocolul RS232". Vedeţi dumneavoastră, cartea de faţă „nu tratează comunicaţiile seriale de tip RS232" (sau orice alt concept teoretic despre comunicaţii). În plus, trebuie remarcat că modulele de hardware şi firmware care „execută" communicaţia RS232 pe placa LHFSD-HCK funcţionează extraordinar de bine, în ciuda faptului că atât implementarea în hardware cât şi cea în firmware sunt cele mai simple pe care eu le-am văzut în viaţa mea.

Remarcaţi, vă rog, că această carte nu face parte din categoria „Ghidul Idiotului în HFS Design" [există o serie întreagă de cărţi în limba engleză, în USA, de tip „Ghidul Idiotului"]. Cartea de faţă este un un ghid practic de a începe munca dificilă de HFS design folosind un exemplu care, din nou, FUNCŢIONEAZĂ!

Acest exemplu a fost testat ani de zile, el este suficient de bine documentat, şi este destul de detaliat pentru această primă carte din seria HFS design. Mai departe, este posibil că o să găsesc ceva resurse materiale (şi psihice) să mai scriu câteva cărţi din seria HFS design ca să vă ajut mai mult. Deocamdată, asta e, dragi cititori.

COMENTARII DESPRE LEGEA COPYRIGHT

Mai devreme sau mai târziu în viaţa dumneavoastră o să aveţi nevoie ca legea *Copyright* să vă protejeze creaţia intelectuală proprie; de aceea, cel mai bine este dacă legea *Copyright* lucrează din plin, oriunde în lume. Din nefericire, foarte mulţi oameni violează legea *Copyright*, fiindcă aplicarea ei (mai ales pe Internet) este deosebit de dificilă. Tentaţia este cauzată de o formă sau alta de venituri/profituri ilicite, şi ea poate fi efectiv controlată numai de răspunsul individual corect la întrebarea, ***„S-o fac, sau nu?"***

Legile de copyright sunt fundamentul dezvoltării în civilizaţia noastră. Remarcaţi că antonimul lui „copyright" este „hoţie". Bineînţeles, vedem în filme că hoţii numiţi *„hackers"* sunt prezentaţi ca „eroi" numai că asta este doar *propaganda oamenilor mediocri.* Cei care cedează tentaţiilor ieftine nu pot fi eroi, dragi cititori; numai cei care *rezistă tentaţiilor* sunt exemple demne de invidiat, şi de urmat. Să furi munca unei persoane în cinci minute este un act mizerabil; dar, să munceşti zece ani pentru a crea ceva folositor, bun pentru toată lumea, aceasta este cu adevărat extraordinar. A ceda senzaţiilor şi dorinţelor nu înseamnă absolut nimic: oricine este „capabil" s-o facă. Însă, dacă aveţi puterea să spuneţi „nu" oricărei forme de tentaţie, aceasta o să va întărească mintea, voinţa, şi organismul dumneavoastră în faţa greutăţilor—şi fiţi sigur că numai puţini sunt în stare s-o facă.

Este datoria noastră să protejăm şi să respectăm proprietatea intelectuală—din nou, chiar şi dumneavoastră o să aveţi nevoie de legea *Copyright* atunci când o să deveniţi un designer sau un programator. Mai mult chiar, dincolo de „nevoie" mai este şi problema „principiului". Dacă înţelegeţi acest concept acum, eu unul nu mă îndoiesc că lucrurile au să meargă foarte bine pentru dumneavoastră în viitor; dacă nu . . . Ei, asta e.