

Clubul Copiilor Petroșani

DPIR



<< Detector de Proximitate in InfraRoșu >>

Popescu Marius, Clasa a X-a A
Liceul Teoretic Petrosani

Profesor coordonator: Kovacs Imre

Mai 2004

CUPRINS

1. Introducere in MicroControllere
2. Despre PIC12F675
3. Conceperea softului
4. Programarea MicroControllerului
5. Proiectarea si realizarea circuitului imprimat
6. Functionarea DPIP

Introducere

Circumstanțele în care ne găsim astăzi în domeniul microcontrolerelor și-au avut începuturile în dezvoltarea tehnologiei circuitelor integrate. Această dezvoltare a făcut posibilă înmagazinarea a sute de mii de tranzistoare într-un singur cip. Aceasta a fost o premiză pentru producția de microprocesoare, și primele calculatoare au fost făcute prin adăugarea perifericelor ca memorie, linii intrare-ieșire, timer-i și altele. Următoarea creștere a volumului capsulei a dus la crearea circuitelor integrate. Aceste circuite integrate conțin atât procesorul cât și perifericele. Așa s-a întâmplat cum primul cip conținând un microcalculator, sau ce va deveni cunoscut mai târziu ca microcontroler a luat ființă.

Istorie

Este anul 1969, și o echipă de ingineri japonezi de la compania BUSICOM sosesc în Statele Unite cu cererea ca unele circuite integrate pentru calculatoare să fie făcute folosind proiectele lor. Propunerea a fost făcută către INTEL, iar Marcian Hoff a fost desemnat responsabil cu acest proiect. Pentru că el era cel ce avea experiență în lucrul cu un calculator (PC) PDP8, i-a venit să sugereze o soluție diferită fundamental în locul construcției propuse. Această soluție presupunea că funcționarea circuitului integrat este determinată de un program memorat în el. Aceasta a însemnat că configurația ar fi fost mult mai simplă, dar aceasta ar fi cerut mult mai multă memorie decât ar fi cerut proiectul propus de inginerii japonezi. După un timp, cu toate că inginerii japonezi au încercat să caute o soluție mai simplă, ideea lui Marcian a câștigat, și a luat naștere primul microprocesor. În transformarea unei idei într-un produs finit, Federico Faggin a fost de un ajutor major pentru INTEL. El s-a transferat la INTEL, și doar în 9 luni a reușit să scoată un produs din prima sa concepție. INTEL a obținut drepturile de a vinde acest bloc integral în 1971. În primul rând ei au cumpărat licența de la compania BUSICOM care nu au avut idee ce comoară avuseseră. În timpul aceluia an a apărut pe piață un microprocesor numit 4004. Acela a fost primul microprocesor de 4 biți cu viteză 6000 operații pe secundă. Nu mult după aceea, compania americană CTC a cerut de la INTEL și de la Texas Instruments să facă un microprocesor pe 8 biți pentru folosință în terminale. Cu toate că CTC a renunțat la această idee până la sfârșit, INTEL și Texas Instruments au continuat să lucreze la microprocesor și în aprilie 1972 a apărut pe piață primul microprocesor de 8 biți sub numele de 8008. Putea să adreseze 16Kb de memorie și avea 45 de instrucțiuni și viteză de 300.000 de operații pe secundă. Acel microprocesor a fost predecesorul tuturor microprocesoarelor de astăzi. INTEL au continuat dezvoltările lor până în aprilie 1974 și au lansat pe piață microprocesorul de 8 biți sub numele de 8080 ce putea adresa 64Kb de memorie și avea 75 de instrucțiuni, iar prețul începuse de la 360\$.

Într-o altă companie americană Motorola, și-au dat seama repede ce se întâmpla, așa că au lansat pe piață un microprocesor de 8 biți 6800. Constructor șef era Chuck Peddle și pe lângă microprocesorul propriu-zis, Motorola a fost prima companie care să facă alte periferice ca 6820 și 6850. La acel timp multe companii au recunoscut marea importanță a microprocesoarelor și au început propriile lor dezvoltări. Chuck Peddle părăsește Motorola pentru a se muta la MOS Technology și continuă să lucreze intensiv la dezvoltarea microprocesoarelor.

La expoziția WESCON din Statele Unite din 1975 a avut loc un eveniment critic în istoria microprocesoarelor. MOS Technology a anunțat că produce microprocesoarele 6501 și 6502 la 25\$ bucata pe care cumpărătorii le puteau cumpăra imediat. Aceasta a fost atât de senzational încât au crezut că este un fel de înșelăciune, gândind că competitorii vindeau 8080 și 6800 la 179\$. Ca un răspuns la competitorii lor atât INTEL cât și Motorola au scăzut prețurile lor în prima zi a expoziției până la 69.95\$ pe microprocesor. Motorola intenționează repede proces contra lui MOS Technology și contra lui Chuck Peddle pentru copierea protejatului 6800. MOS Technology încetează de a mai produce 6501 dar continuă să producă 6502. 6502 este un microcontroler pe 8 biți cu 56 de instrucțiuni și o capacitate de adresare directă de 64Kb de memorie. Datorită costului scăzut, 6502 devine foarte popular, așa că este instalat în calculatoare ca :KIM-1, Apple I, Apple II, Atari, Comodore, Acorn, Oric, Galeb, Oreo, Ultra și multe altele. Curând apar câțiva producători de 6502 (Rockwell, Sznertek, GTE, NCR, Ricoh și Comodore preiau MOS Technology) ce era în momentul prosperității sale vândut la o rată de 15 milioane de microprocesoare pe an!

Alții totuși nu au cedat. Federico Faggin părăsește INTEL, și își pornește propria sa companie Zilog Inc.

În 1976 Zilog anunță Z80. În timpul creării acestui microprocesor, Faggin ia o decizie crucială. Știind că un mare număr de programe fuseseră dezvoltate pentru 8080, Faggin își dă seama că mulți vor rămâne fideli aceluia microprocesor din cauza marii cheltuieli care ar rezulta în urma refacerii tuturor programelor. Astfel el decide că un nou microprocesor trebuie să fie compatibil cu 8080, sau că trebuie să fie capabil să execute toate programele care deja fuseseră scrise pentru 8080. În afară acestor caracteristici, multe altele noi au fost adăugate, așa că Z80 a fost un microprocesor foarte puternic la vremea lui. Putea adresa direct 64Kb de memorie, avea 176 instrucțiuni, un număr mare de registre, o opțiune incorporată pentru reîmprospătarea memoriei RAM dinamice, o singură sursă, viteză de lucru mult mai mare etc. Z80 a fost un succes mare și toată lumea a făcut conversia de 8080 la Z80. Se poate spune că Z80 comercial, a fost fără nici o îndoială, cel mai de succes microprocesor de 8 biți a acelui timp. În afară de Zilog, alți noi producători apar de asemenea ca: Mostek, NEC, SHARP și SGS. Z80 a fost inima a multor calculatoare ca: Spectrum, Partner, TRS703, Z-3.

În 1976, INTEL iese pe piață cu o versiune îmbunătățită de microprocesor pe 8 biți numit 8085. Totuși, Z80 era cu mult mai bun decât INTEL curând a pierdut bătălia. Chiar dacă au apărut pe piață încă câteva microprocesoare (6809, 2650, SC/MP etc.), totul fusese de fapt deja hotărât. Nu mai erau de făcut îmbunătățiri importante ca să-i facă pe producători să se convertească spre ceva nou, așa că 6502 și Z80 împreună cu 6800 au rămas ca cei mai reprezentativi ai microprocesoarelor de 8 biți ai acelui timp.

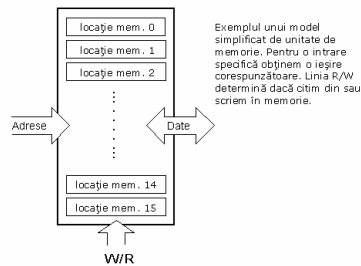
Microcontrolere contra Microprocesoare

Microcontrolerul diferă de un microprocesor în multe feluri. În primul rând și cel mai important este funcționalitatea sa. Pentru a fi folosit, unui microprocesor trebuie să i se adauge alte componente ca memorie, sau componente pentru primirea și trimiterea de date. Pe scurt, aceasta înseamnă că microprocesorul este inima calculatorului. Pe de altă parte, microcontrolerul este proiectat să fie toate acestea într-unul singur. Nu sunt necesare alte componente externe pentru aplicarea sa pentru că toate perifericele necesare sunt deja incluse în el. Astfel, economisim timpul și spațiul necesare pentru construirea de aparate.

1.1 Unitatea de memorie

Memoria este o parte a microcontrolerului a cărei funcție este de a înmagazina date.

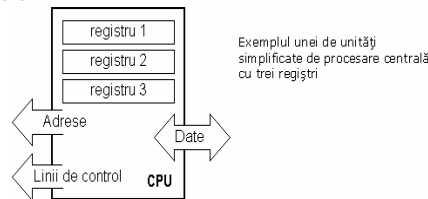
Cel mai ușor mod de a explica este de a-l descrie ca un dulap mare cu multe sertare. Dacă presupunem că am marcat sertarele într-un asemenea fel încât să nu fie confundate, oricare din conținutul lor va fi atunci ușor accesibil. Este suficient să se știe desemnarea sertarului și astfel conținutul lui ne va fi cunoscut în mod sigur.



Componentele de memorie sunt exact așa. Pentru o anumită intrare obținem conținutul unei anumite locații de memorie adresate și aceasta este totul. Două noi concepte ne sunt aduse: adresarea și locația de memorie. Memoria constă din toate locațiile de memorie, și adresarea nu este altceva decât selectarea uneia din ele. Aceasta înseamnă că noi trebuie să selectăm locația de memorie la un capăt, și la celălalt capăt trebuie să așteptăm conținutul acelei locații. În afară de citirea dintr-o locație de memorie, memoria trebuie de asemenea să permită scrierea în ea. Aceasta se face prin asigurarea unei linii adiționale numită linie de control. Vom desemna această linie ca R/W (citește /scrie). Linia de control este folosită în următorul fel: dacă $r/w=1$, se face citirea, și dacă opusul este adevărat atunci se face scrierea în locația de memorie. Memoria este primul element, dar avem nevoie și de altele pentru ca microcontrolerul nostru să funcționeze.

1.2 Unitatea de procesare centrală

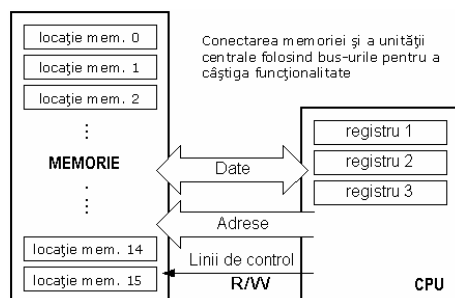
Să adăugăm alte 3 locații de memorie pentru un bloc specific ce va avea o capacitate incorporată de înmulțire, împărțire, scădere și să-i mutăm conținutul dintr-o locație de memorie în alta. Partea pe care tocmai am adăugat-o este numită "unitatea de procesare centrală" (CPU). Locațiile ei de memorie sunt numite regiștri.



Regiștrii sunt deci locații de memorie al căror rol este de a ajuta prin executarea a variate operații matematice sau a altor operații cu date oriunde se vor fi găsit datele. Să privim la situația curentă. Avem două entități independente (memoria și CPU) ce sunt interconectate, și astfel orice schimb de informații este ascuns, ca și funcționalitatea sa. Dacă, de exemplu, dorim să adăugăm conținutul a două locații de memorie și întoarcem rezultatul înapoi în memorie, vom avea nevoie de o conexiune între memorie și CPU. Mai simplu formulat, trebuie să avem o anumită "cale" prin care datele circulă de la un bloc la altul.

1.3 Bus-ul

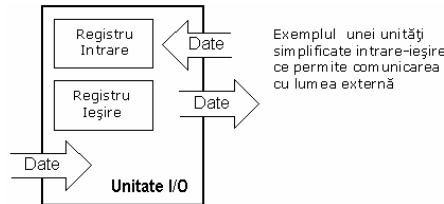
Calea este numită "bus"-magistrală. Fizic, el reprezintă un grup de 8, 16, sau mai multe fire. Sunt două tipuri de bus-uri: bus de adresă și bus de date. Primul constă din atâtea linii cât este cantitatea de memorie ce dorim să o adresăm, iar celălalt este atât de lat cât sunt datele, în cazul nostru 8 biți sau linia de conectare. Primul servește la transmiterea adreselor de la CPU la memorie, iar cel de al doilea la conectarea tuturor blocurilor din interiorul microcontrolerului.



În ceea ce privește funcționalitatea, situația s-a îmbunătățit, dar o nouă problemă a apărut de asemenea: avem o unitate ce este capabilă să lucreze singură, dar ce nu are nici un contact cu lumea de afară, sau cu noi! Pentru a înlătura această deficiență, să adăugăm un bloc ce conține câteva locații de memorie al căror singur capăt este conectat la bus-ul de date, iar celălalt are conexiune cu liniile de ieșire la microcontroler ce pot fi văzute cu ochiul liber ca pini la componenta electronică.

1.4 Unitatea intrare-ieșire

Aceste locații ce tocmai le-am adăugat sunt numite "porturi". Sunt diferite tipuri de porturi: intrare, ieșire sau porturi pe două-căi. Când se lucrează cu porturi, mai întâi de toate este necesar să se aleagă cu ce port urmează să se lucreze, și apoi să se trimită date la, sau să se ia date de la port.



Când se lucrează cu el portul se comportă ca o locație de memorie. Ceva este pur și simplu scris în sau citit din el, și este posibil de a remarca ușor aceasta la pini microcontrolerului.

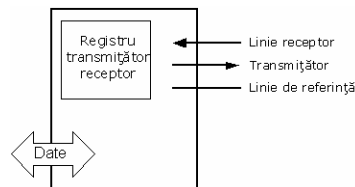
1.5 Comunicația serială

Cu aceasta am adăugat la unitatea deja existentă posibilitatea comunicării cu lumea de afară. Totuși, acest mod de comunicare are neajunsurile lui. Unul din neajunsurile de bază este numărul de linii ce trebuie să fie folosite pentru a transfera datele. Ce s-ar întâmpla dacă acestea ar trebui transferate la distanță de câțiva kilometri? Numărul de linii înmulțit cu numărul de kilometri nu promite costuri eficiente pentru proiect.

Nu ne rămâne decât să reducem numărul de linii într-un așa fel încât să nu scădem funcționalitatea. Să presupunem că lucrăm doar cu 3 linii, și că o linie este folosită pentru trimiterea de date, alta pentru recepție și a treia este folosită ca o linie de referință atât pentru partea de intrare cât și pentru partea de ieșire.

Pentru ca aceasta să funcționeze, trebuie să stabilim regulile de schimb ale datelor. Aceste reguli sunt numite protocol. Protocolul este de aceea definit în avans ca să nu fie nici o neînțelegere între părțile ce comunică una cu alta. De exemplu, dacă un om vorbește în franceză, și altul vorbește în engleză, este puțin probabil că ei se vor înțelege repede și eficient unul cu altul. Să presupunem că avem următorul protocol. Unitatea logică "1" este setată pe linia de transmisie până ce începe transferul.

Odată ce începe transferul, coborâm linia de transmisie la "0" logic pentru o perioadă de timp (pe care o vom desemna ca T), așa că partea receptoare va ști că sunt date de primit, așa că va activa mecanismul ei de recepție. Să ne întoarcem acum la partea de transmisie și să începem să punem zero-uri și unu-uri pe linia de transmisie în ordinea de la un bit a celei mai de jos valori la un bit a celei mai de sus valori. Să lăsăm ca fiecare bit să rămână pe linie pentru o perioadă de timp egală cu T, și la sfârșit, sau după al 8-lea bit, să aducem unitatea logică "1" înapoi pe linie ce va marca sfârșitul transmisiei unei date. Protocolul ce tocmai l-am descris este numit în literatura profesională NRZ (Non-Return to Zero).

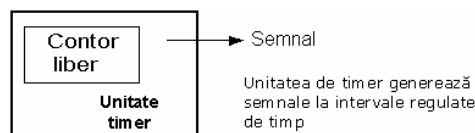


Unitatea serială folosită pentru a trimite date, dar numai prin trei linii

Pentru că avem linii separate de recepție și de transmitere, este posibil să recepționăm și să transmitem date (informații) în același timp. Blocul așa numit full-duplex mode ce permite acest mod de comunicare este numit blocul de comunicare serială. Spre deosebire de transmisia paralelă, datele sunt mutate aici bit cu bit, sau într-o serie de biți, de unde vine și numele de comunicație serială. După recepția de date trebuie să le citim din locația de transmisie și să le înmagazinăm în memorie în mod opus transmiterii unde procesul este invers. Datele circulă din memorie prin bus către locația de trimitere, și de acolo către unitatea de recepție conform protocolului.

1.6 Unitatea timer

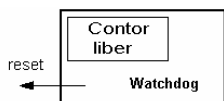
Acum că avem comunicația serială, putem recepționa, trimite și procesa date.



Totuși, pentru noi ca să putem să îl folosim în industrie mai avem nevoie de câteva blocuri. Unul din acestea este blocul timer care este important pentru noi pentru că ne dă informația de timp, durată, protocol etc. Unitatea de bază a timer-ului este un contor liber (free-run) care este de fapt un registru a cărui valoare numerică crește cu unu la intervale egale, așa încât luându-i valoarea după intervalele T1 și T2 și pe baza diferenței lor să putem determina cât timp a trecut. Acesta este o parte foarte importantă a microcontrolerului al cărui control cere cea mai mare parte a timpului nostru.

1.7 Watchdog-ul

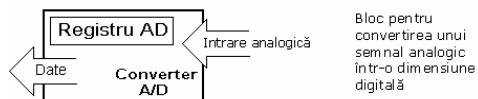
Încă un lucru ce necesită atenția noastră este funcționarea fără defecte a microcontrolerului în timpul funcționării. Să presupunem că urmare a unei anumite interferențe (ce adesea se întâmplă în industrie) microcontrolerul nostru se oprește din executarea programului, sau și mai rău, începe să funcționeze incorect.



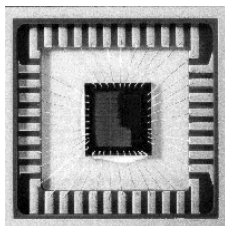
Bineînțeles, când aceasta se întâmplă cu un calculator, îl resetăm pur și simplu și va continua să lucreze. Totuși, nu există buton de resetare pe care să-l apăsăm în cazul microcontrolerului care să rezolve astfel problema noastră. Pentru a depăși acest obstacol, avem nevoie de a introduce încă un bloc numit watchdog-câinele de pază. Acest bloc este de fapt un alt contor liber (free-run) unde programul nostru trebuie să scrie un zero ori de câte ori se execută corect. În caz că programul se "înțepenește", nu se va mai scrie zero, iar contorul se va reseta singur la atingerea valorii sale maxime. Aceasta va duce la rularea programului din nou, și corect de această dată pe toată durata. Acesta este un element important al fiecărui program ce trebuie să fie fiabil fără supravegherea omului.

1.8 Convertorul Analog-Digital

Pentru că semnalele de la periferice sunt substanțial diferite de cele pe care le poate înțelege microcontrolerul (zero și unu), ele trebuie convertite într-un mod care să fie înțeles de microcontroler. Această sarcină este îndeplinită de un bloc pentru conversia analog-digitală sau de un convertor AD. Acest bloc este responsabil pentru convertirea unei informații despre o anumită valoare analogică într-un număr binar și pentru a o urmări pe tot parcursul la un bloc CPU așa ca blocul CPU să o poată procesa.

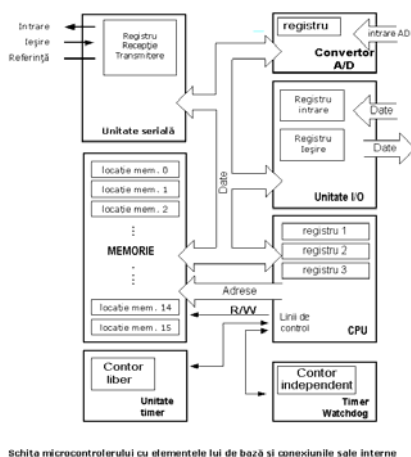


Astfel microcontrolerul este acum terminat, și tot ce mai rămâne de făcut este de a-l pune într-o componentă electronică unde va accesa blocurile interioare prin pinii exteriori. Imaginea de mai jos arată cum arată un microcontroler în interior.



Configurația fizică a interiorului unui microcontroler

Linii subțiri ce merg din interior către părțile laterale ale microcontrolerului reprezintă fire conectând blocurile interioare cu pinii capsulei microcontrolerului. Schema următoare reprezintă secțiunea centrală a microcontrolerului.



Pentru o aplicație reală, un microcontroler singur nu este de ajuns. În afară de microcontroler, avem nevoie de un program pe care să-l executăm, și alte câteva elemente ce constituie o interfață logică către elementele de stabilizare.

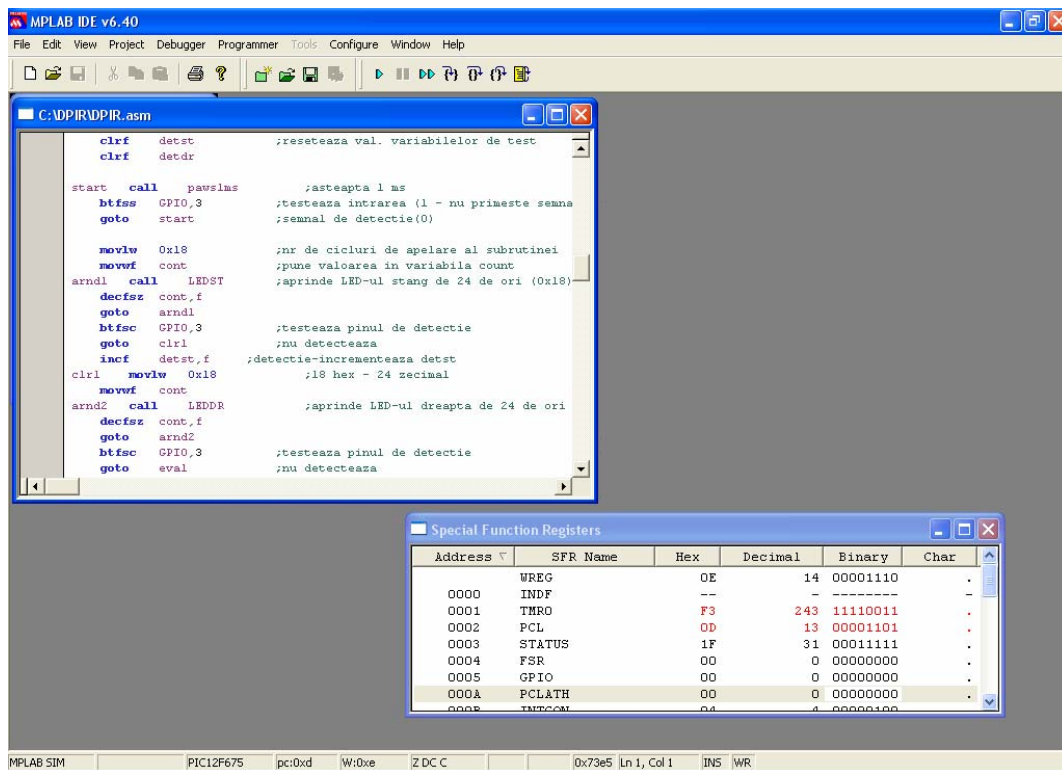
1.9 Programul

Scrierea programului este un domeniu special de lucru al microcontrolerului și este denumit "programare". Programarea poate fi făcută în câteva limbaje ca Assembly, C și Basic care sunt cele mai folosite limbaje.

Assembler aparține limbajelor de nivel scăzut ce sunt programate lent, dar folosesc cel mai mic spațiu în memorie și dă cele mai bune rezultate când se are în vedere viteza de execuție a programului. De asemenea este cel mai folosit în programarea MicroControllerelor. Programele în limbajul C sunt mai ușor de scris, mai ușor de înțeles, dar sunt mai lente în executare decât programele în Assembler.

Basic este cel mai ușor de învățat, și instrucțiunile sale sunt cele mai aproape de modul de gândire a omului, dar ca și limbajul de programare C este de asemenea mai lent decât Assembler-ul. În orice caz, înainte de a vă hotărî în privința unuia din aceste limbaje trebuie să studiați cu atenție cerințele privind viteza de execuție, mărimea memoriei și timpul disponibil pentru asamblarea sa.

După ce este scris programul, trebuie să instalăm microcontrolerul într-un aparat și să-l lăsăm să lucreze. Pentru a face aceasta trebuie să adăugăm câteva componente externe necesare pentru funcționarea sa. Mai întâi trebuie să dăm viață microcontrolerului prin conectarea sa la o sursă (tensiune necesară pentru operarea tuturor instrumentelor electronice) și oscilatorului al cărui rol este similar inimii din corpul uman. Bazat pe ceasul său microcontrolerul execută instrucțiunile programului. Îndată ce este alimentat microcontrolerul va executa un scurt control asupra sa, se va uita la începutul programului și va începe să-l execute. Cum va lucra aparatul depinde de mulți parametri, cel mai important fiind priceperea dezvoltatorului de hardware, și de experiența programatorului în obținerea maximumului din aparat cu programul său.



MPLAB - mediu de dezvoltare PIC

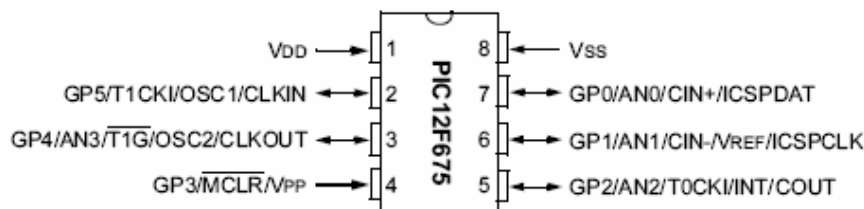
2. Despre PIC12F675



CPU RISC de mare performanta:

- doar 35 de instructiuni
 - viteza de operare:
 - DC - 20 MHz oscillator/clock input
 - DC - 200 ns ciclu de instructiune
 - capabilitate de intrerupere a programului
 - stiva hardware organizata pe 8 nivele
 - Adresare directa,indirecta si relativa
- Special Microcontroller Features:
- optiune de setare a oscilatorului intern sau extern
 - Oscilator intern de precizie calibrat din fabricatie la 4Mhz
 - suporta oscilatoare externe cu cuart sau rezonatoare
 - 5 μ s "trezirea" din modul SLEEP, 3.0V,
 - SLEEP mode
 - Suporta tensiuni de alimentare intre 2.0V si 5.5V
 - Low power Power-on Reset (POR)

- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Detect (BOD)
- Watchdog Timer (WDT) cu oscillator independent
- pin MCLR/Input multiplexat
- rezistente de pull-up programabile individual
- Protectie la copiere
 - 100,000 scrieri FLASH
 - 1,000,000 scrieri EEPROM
 - Retentia FLASH/Data EEPROM : > 40 ani
- Functionare la putere scazuta :
 - Curent de Stand-By:
 - 1 nA @ 2.0V, tipic
 - Current de operare:
 - 8.5 μ A @ 32 kHz, 2.0V, tipic
 - 100 μ A @ 1 MHz, 2.0V, tipic
 - Curent in modul Watchdog Timer
 - 300 nA @ 2.0V, tipic
 - Curent in modul Timer1 oscillator :
 - 4 μ A @ 32 kHz, 2.0V, tipic
- Capabilitati periferice:
 - 6 pini I/O cu control individual al sensului
 - Suporta curenti de 20 mA direct pe pin (Led Drive)
 - Modul comparator analog:
 - comparator analog
 - Posibilitatea programarii tensiunii de referinta
 - Suporta multiplexare
 - Modul convertor Analog-Digital:
 - 10-bit rezolutie
 - intrare programabila pe 4 canale
 - intrare pt tensiunea de referinta
 - Timer0: timer 8 biti programabil
 - Timer1 avansat:
 - 16-bit timer/counter cu prescaler
 - mod External Gate Input
 - Posibilitatea de a folosi OSC1 shi OSC2 pt timer1 daca este selectat modul IntOsc
 - Posibilitate de programare in circuit pe 2 pini



Instructiunile lui PIC12F675

Introducere

Am menționat deja că microcontrolerul nu este ca orice alt circuit integrat. Când ies din producție cele mai multe circuite integrate sunt gata de a fi introduse în aparate ceea ce nu este cazul cu microcontrolerele. Pentru a "face" microcontrolerul să îndeplinească o sarcină, trebuie să-i spunem exact ce să facă, sau cu alte cuvinte trebuie să scriem programul pe care microcontrolerul să-l execute. Vom descrie în acest capitol instrucțiunile care alcătuiesc assembler-ul, sau limbajul de programare cu nivel scăzut pentru microcontrolerele PIC.

Set de Instrucțiuni în Familia Microcontrolerului PIC12Fxxx

Setul complet care cuprinde 35 de instrucțiuni este dat în tabela următoare. Un motiv pentru un număr așa de mic de instrucțiuni stă în primul rând în faptul că discutăm despre un microcontroler RISC ale cărui instrucțiuni sunt bine optimizate având în vedere viteza de lucru, simplitatea arhitecturală și compactitatea codului. Singurul neajuns este că programatorul trebuie să controleze o tehnică "neconfortabilă" în a utiliza un set modest de 35 de instrucțiuni.

Transfer de Date

Transferul de date într-un microcontroler este făcut între registrul de lucru (W) și un registru 'f' ce reprezintă orice locație în RAM-ul intern (indiferent dacă aceștia sunt regiștri speciali sau de scop general).

Primele trei instrucțiuni (a se vedea următorul tabel) fac ca o constantă să fie înscrisă în registrul W (MOVLW este prescurtarea pentru MOVE Literal to W), și ca datele să fie copiate din registrul W în RAM și datele din RAM să fie copiate în registrul W (sau în aceeași locație RAM, la care punct numai starea stegulețului Z se schimbă). Instrucțiunea CLRF scrie constanta 0 în registrul 'f', iar CLRW scrie constanta 0 în registrul W. Instrucțiunea SWAPF schimbă locurile câmpului de nibbles- bucăți de 4 biți în interiorul unui registru.

Aritmetică și logică

Din toate operațiile aritmetice, PIC ca majoritatea microcontrolerelor, acceptă doar scăderea și adunarea. Stegulețele C, DC și Z sunt setate funcție de rezultatul adunării sau scăderii, dar cu o excepție: pentru că scăderea se face ca o adunare a unei valori negative, eticheta C este inversă urmând scăderii. Cu alte cuvinte, este setată dacă operația este posibilă, și este resetată dacă un număr mai mare a fost scăzut din unul mai mic.

Unitatea logică a PIC-ului are capabilitatea de a face operațiile AND (ȘI), OR (SAU), EX-OR (SAU-EXCLUSIV), complementare (COMF) și rotație (RLF și RRF).

Instrucțiunile ce rotesc conținutul registrului mută biții în interiorul registrului prin eticheta C cu un spațiu la stânga (către bitul 7), sau la dreapta (către bitul 0). Bitul ce "iese" din registru este scris în stegulețul C, și valoarea stegulețului C este scrisă într-un bit al "părții opuse" a registrului.

Operații cu biți

Instrucțiunile BCF și BSF fac setarea sau ștergerea unui singur bit oriunde în memorie. Chiar dacă pare o simplă operație, este executată în așa fel ca CPU citește mai întâi întregul byte, schimbă un bit în el și apoi scrie întregul byte în același loc.

Direcționarea debitului unui program

Instrucțiunile GOTO, CALL și RETURN sunt executate în același fel ca și în celelalte microcontrolere, numai stiva este independentă de RAM-ul intern și limitată la opt nivele.

Instrucțiunea 'RETLW k' este identică cu instrucțiunea RETURN, cu excepția că înainte de a se întoarce dintr-un subprogram, constanta definită operandul de instrucțiuni este scrisă în registrul W. Această instrucțiune ne permite să proiectăm ușor tabelele (listele) Look-up. Cel mai mult le folosim la determinarea poziției datelor în tabelul nostru adăugând-o la adresa la care încep tabelele, și apoi citim datele din acea locație (care este uzual găsită în memoria program).

Tabelul poate fi format ca un subprogram ce constă dintr-o serie de instrucțiuni 'RETLW k', unde constantele 'k' sunt membri ai tabelului.

```
Main    molov 2
        call Lookup
Lookup  addwf PCL, f
        retlw k
        retlw k1
        retlw k2
        :
        :
        retlw kn
```

Scriem poziția unui membru al tabelului nostru în registrul W, și folosind instrucțiunea CALL apelăm un subprogram care crează tabelul. Prima linie de subprogram ADDWF PCL, f adaugă poziția unui membru al registrului W la adresa de start a tabelului nostru, găsită în registrul PCL, și astfel obținem adresa datelor reale în memoria program. Când ne întoarcem dintr-un subprogram vom avea în registrul W conținutul unui membru al tabelului adresat. În exemplul anterior, constanta 'k2' va fi în registrul W urmând unei întoarceri dintr-un subprogram.

RETFIE (RETurn From Interrupt - Interrupt Enable) este o întoarcere dintr-o rutină de întrerupere și diferă de o RETURN numai în aceea că setează automat bitul GIE (Global Interrupt Enable). La o întrerupere, acest bit este automat șters. Când începe întreruperea, numai valoarea contorului de program este pusă în vârful stivei. Nu este prevăzută memorarea automată a valorilor și stării registrului.

Jump-urile (salturile) condiționale sunt sintetizate în două instrucțiuni: BTFSZ și BTFS. Funcție de starea bitului în registrul 'f' ce este testat, instrucțiunile sar sau nu peste instrucțiunea de program următoare.

Perioada de Execuție a Instrucțiunii

Toate instrucțiunile sunt executate într-un ciclu cu excepția instrucțiunilor ramură condiționale dacă condiția a fost adevărată, sau dacă conținutul contorului de program a fost schimbat de o anumită instrucțiune. În acest caz, execuția cere două cicluri de instrucțiuni, iar al doilea ciclu este executat ca NOP (No Operation-Fără operații). Patru clock-uri oscilator fac un ciclu instrucțiune. Dacă folosim un oscilator cu frecvența de 4 MHz, timpul normal pentru execuția instrucțiunii este 1 μs, și în caz de branching-ramificare condițională, perioada de execuție este 2 μs.

Listă de cuvinte

- f orice locație de memorie într-un microcontroler
- W registru de lucru
- b poziție bit în registru 'f'
- d bit destinație
- label grup de opt caractere ce marchează începutul unei părți de program
- TOS vârful stivei
- [] opțiune
- <> poziție bit în registru

Mnemonic	Descriere	Operație	Steguleț	Ciclu	Notă
Transfer date					
MOVLW	k	Mută constanta în W	k → W		1
MOVWF	f	Mută W în f	W → f		1
MOVF	f, d	Mută f	f → d	Z	1, 2
CLRWF	-	Șterge W	00h → W, 1 → Z	Z	1
CLRF	f	Șterge f	00h → f, 1 → Z	Z	1, 2
SWAPF	f, d	Interschimbă nibble-urile în f	f(7:4) → f(3:0), f(3:0) → f(7:4)		1, 2
Aritmetică și logică					
ADDFW	k	Adună constanta cu W	W+k → W	C, DC, Z	1
ADDWF	f, d	Adună W cu f	W+f → d	C, DC, Z	1, 2
SUBLW	k	Scade W din constanta	k-W → W	C, DC, Z	1
SUBWF	f, d	Scade W din f	f-W → d	C, DC, Z	1, 2
ANDLW	k	ȘI literal cu W	W.AND.k → W	Z	1
ANDWF	f, d	ȘI W cu f	W.AND.f → d	Z	1, 2
IORLW	k	SAU inclusiv constanta cu W	W.OR.k → W	Z	1
IORWF	f, d	SAU inclusiv W cu f	W.OR.f → d	Z	1, 2
XORLW	k	SAU exclusiv constanta cu W	W.XOR.k → W	Z	1, 2
XORWF	f, d	SAU exclusiv W cu f	W.XOR.f → d	Z	1
INCF	f, d	Incrementează f	f+1 → f	Z	1, 2
DECF	f, d	Decrementează f	f-1 → f	Z	1, 2
RLF	f, d	Rotește la stânga prin Carry		C	1, 2
RRF	f, d	Rotește la dreapta prin Carry		C	1, 2
COMF	f, d	Complement f	f → d	Z	1, 2
Operații cu biți					
BCF	f, b	Șterge bitul f	0 → f(b)		1, 2
BSF	f, b	Setează bitul f	1 → f(b)		1, 2
Direcționarea unui debit de program					
BTFSZ	f, b	Testează bitul f, Sari dacă este șters	salt dacă f(b)=0		1 (2) 3
BTFS	f, b	Testează bitul f, Sari dacă este setat	salt dacă f(b)=1		1 (2) 3
DECFSZ	f, d	Decrementează f, Sari dacă este 0	f-1 → d, salt dacă Z=1		1(2) 1,2,3
INCFSZ	f, d	Incrementează f, Sari dacă este 0	f+1 → d, salt dacă Z=1		1(2) 1,2,3
GOTO	k	Du-te la adresă	W.AND.k → W		2
CALL	k	Apelează subrutina	W.AND.f → d		2
RETURN	-	Întoarcere din Subrutină	TOS → PC		2
RETLW	k	Întoarcere cu constanta în W	k → W, TOS → PC		2
RETFIE	-	Întoarcere din întrerupere	TOS → PC, 1 → GIE		2
Alte instrucțiuni					
NOP	-	Fără Operații			1
CLRWDI	-	Șterge Timer-ul Watchdog	0 → WDI, 1 → ICF, 1 → PD	T _O , PD	1
SLEEP	-	Du-te în mod standby	0 → WDI, 1 → ICF, 0 → PD	T _O , PD	1

*1 Dacă portul I/O este operand sursă, este citită starea pinilor microcontrolerului

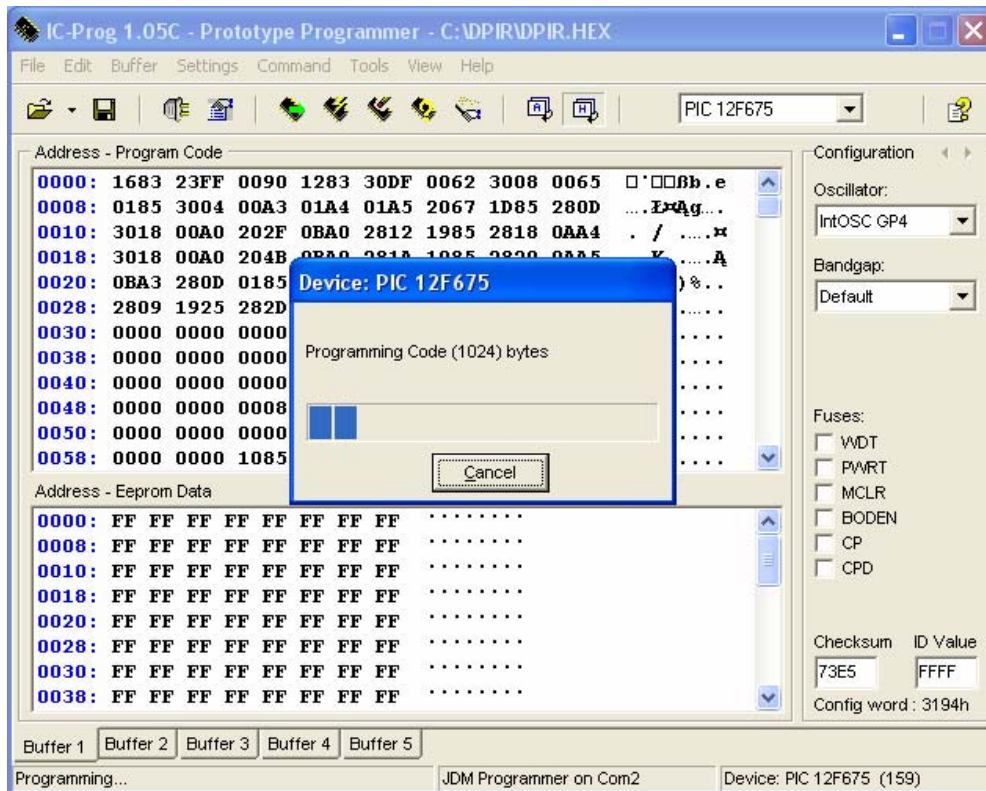
*2 Dacă această instrucțiune este executată în registrul TMRO și dacă d=1, prescaler-ul asignat aceluia timer va fi automat șters

*3 Dacă PC s-a modificat, sau rezultatul testului =1, instrucțiunea s-a executat în două cicluri

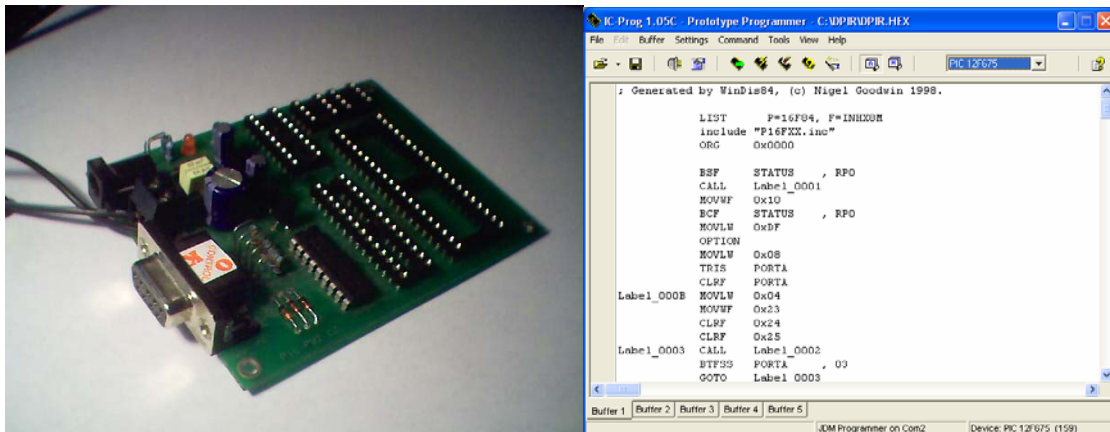
Programarea MicroController-ului

Programarea Micro Controllerului s-a facut in IC PROG (www.icprog.co.uk) aceasta dupa o compilare in prealabil a fisierului assembler (astfel transformandu-l in HEX).

In timpul programarii s-au utilizat optiunile IntOsc (foloseste oscilatorul intern de 4 Mhz) , ClkOut Disabled (GPIO 4 are rol de port bidirectional – nu este folosit ca pin de clock).



ICPROG- Programming

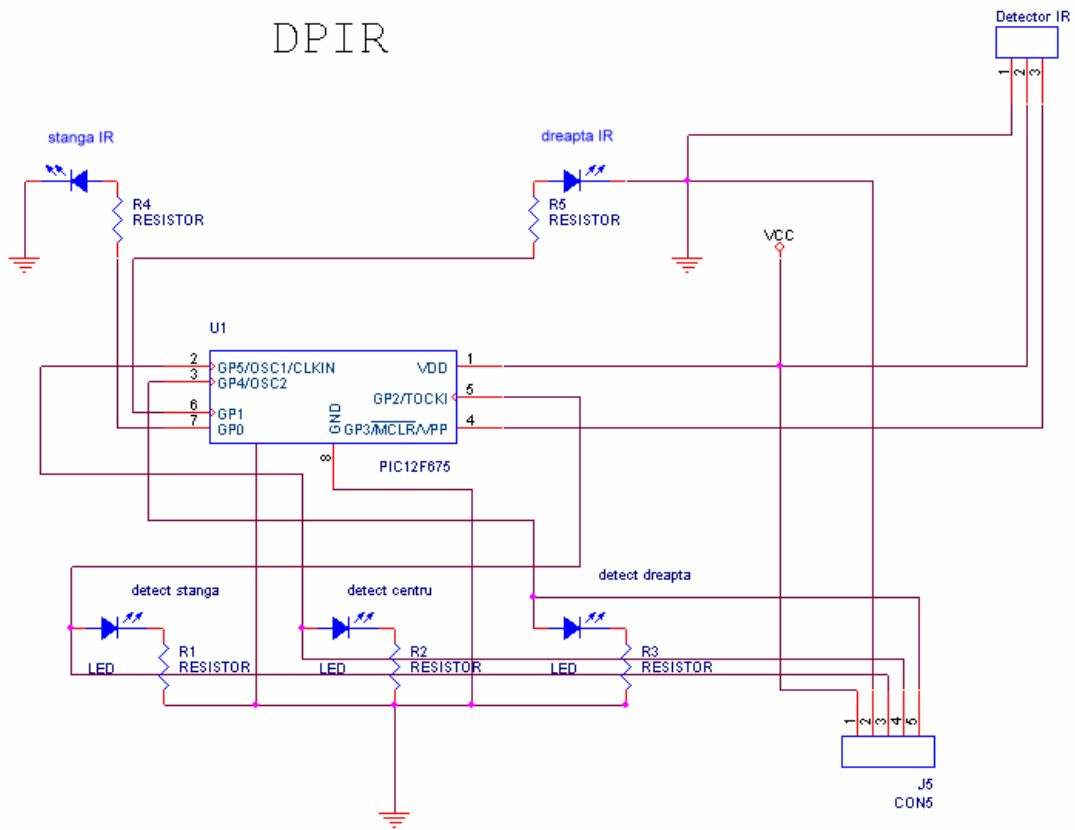


IC PROG

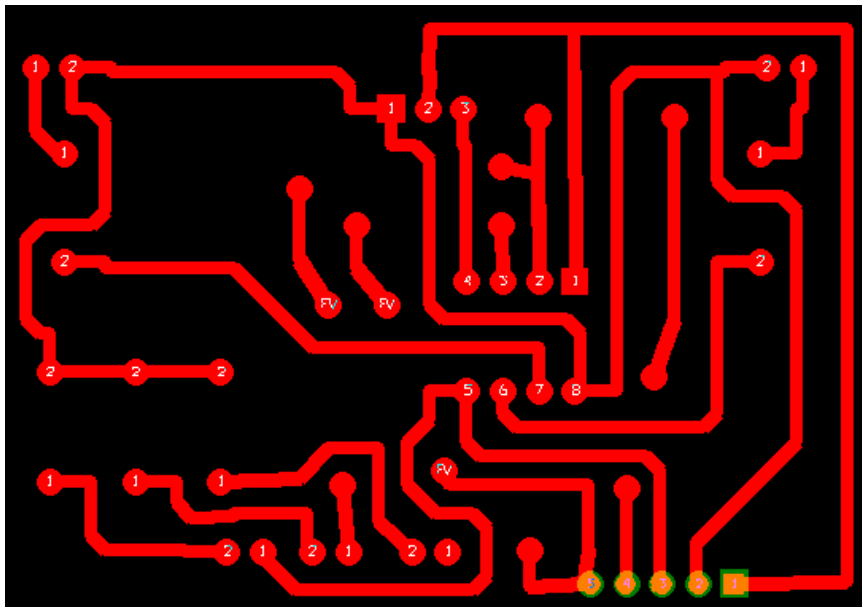
AD-PIC01 PROGRAMMER

Proiectarea si realizarea circuitului imprimat

Schema electrica:



Circuitul imprimat:



Realizarea circuitului imprimat prin metoda fotografica:

Metoda fotografica este una din cele mai folosite metode de realizarea a circuitelor imprimate, fapt care o face atat de populara printre amatori. Are un grad mediu de dificultate in realizare, nefiind necesara o dotare tehnica superioara. In urmatoarele randuri voi prezenta pe scurt metoda de realizare.

Curatarea si degresarea:

Suprafata care urmeaza a fi acoperita de FotoResist trebuie sa fie lipsita de orice urma de grasime. Pentru a realiza acest lucru se curata praf de curatat sau cu glasspapr foarte fin. Curatarea se face pana in momentul cand suprafata devine perfect lucioasa. Nu se vor folosi produse abrazive!

Aplicarea fotoresistului

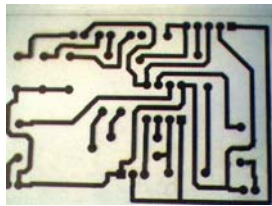
Aplicarea se face intr-o camera slab luminata si fara praf. Placa se aseaza orizontal si se acopera cu FotoResist de la o distanta de 20cm. Pentru o aplicare uniforma se va acoperi pe linii pornind din stanga sus si terminand dreapta jos. In cazul aplicarii unui strat prea gros sau neuniform timpii de expunere se pot chiar dubla.

Uscarea placii

Uscarea se face in intuneric complet. Se lasa 24 de ore la o temperatura de 20° C sau se 20 de min la o temperatura fixa de 70° C. Nu se vor depasi 70° C !

Pregatirea foliei

Modelul circuitului se va tipari pe folie transparenta, utilizand o imprimanta laser pentru rezultate optime. Tiparirea se face in mod pozitiv la o scara de 1 la 1. Se aseaza folia peste placa si se fixeaza astfel incat nici o raza de lumina nu patrunde pe sub trasee.



Folia tiparita

Expunerea la UV

Timpul de expunere depinde de grosimea statului aplicat cat si de intensitatea sursei de lumina. Pentru o lampa UV de 125W timpul de expunere este de la 60 pana la 120 s. Foarte importanta este incalzirea in prealabil a lampii UV de la 3 la 5 min.



Bec UV cu vapori de Hg 125W



Expunere la UV

Developarea

Placa expusa se introduce intr-o solutie de NaOH (7 g la 1l de apa rece) si se agita pentru max 2 min. Dupa acest interval de timp zonele acoperite de trasee trebuie sa fie singurele vizibile. Din acest punct se continua developarea cu inca 50% din timpul scurs pana in momentul respectiv pentru a elimina stratul invizibil de FotoResist care inca persista.

Corodarea

FotoResistul este rezistent la bai acide de Clorura Ferica, Persulfat de Amoniu si Acid Fluorhidric. Cea mai uzuala metoda de corodare este cu Clorura Ferica (45° C la o concentratie de 35-40%).

Dupa corodarea se inlatura FotoResistul folosind acetona si se acopera pentru o mai buna protectie cu un lac protector transparent.

Functionarea DPIR

Functionarea DPIR a fost descrisa anterior in pseudocodul programului dar pentru o mai buna intelegere voi prezenta pe scurt algoritmul.

-se emit 4 randuri de cate 24 de salve IR la o frecventa de 40 kHz (Led IR stanga)

-se face detectia stanga

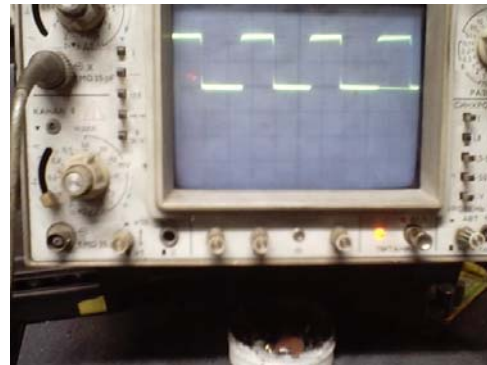
-se emit 4 randuri de cate 24 de salve IR la o frecventa de 40 kHz (Led IR dreapta)

-se face detectia dreapta

-detectie dreapta = 4 , detectie stanga < 4 => aprinde LED detectie dreapta

-detectie dreapta = 4 , detectie stanga = 4 => aprinde LED detectie centru

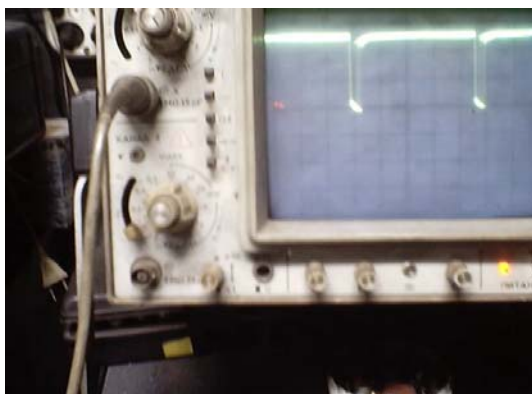
-detectie dreapta < 4 , detectie stanga = 4 => aprinde LED detectie stanga



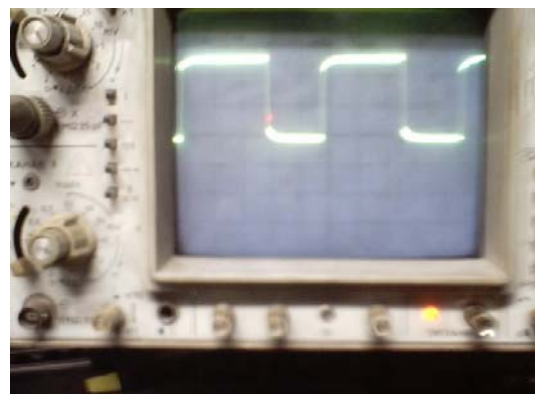
Impulsuri catre LED-urile IR

Exemple de utilizare:

- sistem de evitare a coliziunilor in cazul robotilor autonomi
- bariera IR in sisteme de supraveghere
- orientarea pe timp de noapte



Nu se face detectie (pinul detectorului IR)



Se face detectia

Bibliografie

- www.Microchip.com (Site-ul oficial MicroChip – producator PIC)
- www.mikroelektronika.co.yu
- www.howstuffworks.com
- Totul despre MicroControllere-Emil Pop