UNIVERSITATEA POLITEHNICA TIMISOARA
FACULTATEA DE ELECTRONICA SI TELECOMUNICATII
DEPARTAMENTUL DE COMUNICATII

GENERATOR DE FUNCTII MULTIPLE

PROIECT DE DIPLOMA

Coordonator stiintific,
Prof. dr. ing. MIRANDA NAFORNITA

Coordonator SIEMENS VDO,
Ing. LASZLO GYULAI

Absolventa,
GRADINARU ANDRA-ROXANA

2007
UNIVERSITATEA POLITEHNICA TIMISOARA
FACULTAREA DE ELECTRONICA si TELECOMUNICATII
DEPARTAMENTUL COMUNICATII

Titlul temei: GENERATOR DE FUNCTII MULTIPLE
Enuntul proiectului de diploma:

Sa se realizeze un generator de functii multiple cu scopul de a simula toate semnalele necesare testarii unitatii de control electronice (mai departe numita ECU). Generatorul de functii multiple e destinat a fi introdus într-un echipament complex de testare si validare software si hardware.

Trebuie generate urmatoarele semnale:

- 2 semnale sinusoidale cu frecventa, amplitudine si offset ajustabile
- 1 semnal dreptunghiular cu frecventa si factor de umplere ajustabile (semnal PWM)
- 1 semnal dreptunghiular cu frecventa ajustabila

Tensiunea de alimentare va fi furnizata de o baterie cu valori cuprinse între 8V-36V.
Cele trei semnale trebuie sa întruneasca caracteristicile:

- Semnalul sinusoidal:

Nr.	Descriere	Simbol	Min.	Nom.	Max.	Unitate
1	Amplitudine	V_{AMPL}	0	-	10	V
2	Frecvent a	f_{SINE}	0	-	10	KHz
3	Offset	V_{OFF_SINE}	1	-	5	V

Generatorul de semnal sinusoidal va contine o cale diferentiala folosind un transformator 1:1, si o cale simpla careia ii vom aplica si ajustarea de offset

- Semnal PWM

Nr.	Descriere	Simbol	Min.	Nom.	Max.	Unitate
1	Amplitudine	V_{AMPL_TOSS}	0	-	10	V
2	Frecvent a	f_{SINE}	0	-	2	KHz
3	Durata impulsului	t_{HIGH}	0.4	-	5	ms

- Semnal dreptunghiular

Nr.	Descriere	Simbol	Min.	Nom.	Max.	Unitate
1	Amplitudine	V_{AMPL_TW}	8	-	34	V
2	Frecvent a	f_{SINE}	0	-	2	KHz

Pentru implementarea generatorului se pot folosi circuite integrate sau circuite dedicate pentru sinteza digitala directa, ambele variante fiind acceptate.
Se doreste realizarea unui PCB pentru generatorul de semnal sinusoidal si un PCB pentru cele doua semnale dreptunghiulare.

La proiectarea PCB se va folosi EAGLE, pentru simulari PSpice, iar pentru diverse calcule MathCad.

În final, lucrarea de diploma trebuie sa contina documentatia detaliata referitoare la conceptul de realizare a generatorului de functii multiple, explicatii referitoare la schemele realizate, calcule si simulari, software-ul realizat, foile de catalog folosite si placile electronice realizate.

Locul de executie:
Siemens VDO Timisoara, Calea Martirilor 1989 Nr.1

Data emiterii temei: 01.11.2006

Termen de predare: 26.06.2007

Sef de catedra,
Prof.dr.ing. Ioan Nafornita

Absolventa,
Gradinaru Andra-Roxana

Conducator,
Prof.dr.ing. Miranda Nafornita

Coordonator Siemens VDO
Ing. Laszlo Gyulai

CUPRINS

Capitolul 1
Introducere

1.1 Generalitati despre generatoarele de functii

Un generator de functii este un echipament electronic de testare care genereaza forme de unda. Aceste forme de unda pot fi repetitive sau singulare, caz în care generatorul necesita o sursa de trigger. Formele de unda rezultante sunt aplicâte dispozitivului testat si analizate asa cum parcurg dispozitivul confirmand operabilitatea acestuia, sau din contra, indicând defectele.

Generatoarele de functii au în componenta lor un oscilator electronic, adica un circuit care e capabil sa creeze forme de unda repetitive. Generatoarele moderne folosesc procesarea digitala de semnal, pentru a sintetiza forma de unda , urmata de un convertor numeric-analogic (CNA), pentru a produce o iesire analogica.

Cele mai des folosite forme de unda sunt: semnalul sinusoidal, rampa, dinte de fierastrau, dreptunghiular, PWM (Pulse Width Modulation). În cazul în care se genereaza semnale cu frecvente mai mari de frecventa audio (>20kHz) generatorul de semnal trebuie sa includa si circuite pentru modularea semnalului, modulatie care poate sa fie de amplitudine, de frecventa si de faza.

Generatoarele analogice de functii genereaza un semnal triunghiular ce sta la baza celorlalte forme de unda pe care le poate genera. Semnalul triunghiular se obtine prin încarcarea si descarcarea pe un condensator a unei surse de curent constant. Marea majoritate a generatoarelor analogice contÎn un circuit non-liniar cu o caracteristi ca asemanatoare cu cea a unei diode care poate converti semnalul triunghiular într-un sinus.

O alta câtegorie de generatoare de functii sunt generatoarele de unde arbitrare AWG (Arbitrary Waveform Generators), mai sofisti câte, care permit utilizatorului sa genereze forme de unda arbitrare intr-o anumita gama de frecvente, de nivel de iesire si de acuratete. Spre deosebire de generatoarele de functii obisnuite, care sunt limitate la un anumit numar de forme de unda , aceste generatoare de unde arbitrare permit utilizatorului sa foloseasca un tip de unda în moduri diferite. Aceste generatoare sunt mai scumpe si sunt limitate ca banda.

Generatoarele de functii mai avansate folosesc sinteza digitala directa DDS (Direct Digital Syntesis) pentru a genera forme de unda , prin care se pot crea forme de unda de frecvente arbitrare pornind de la o singura frecventa fixa. Un circuit de baza pentru implementarea metodei DDS contine : un controller electronic, o memorie RAM, un oscilator (cu cuart de obicei) care genereaza frecventa fixa, un numarator/timer, si un convertor numeric-analogic, CNA.

Pentru a face acest dispozitiv sa functioneze trebuie urmati doi: *programarea* si *rularea*.

Etapa de programare consta în încarcarea datelor în memorie de catre controler. Datele sunt cuvinte binare care reprezinta o caracteristi ca a semnalului pe care vrem sa-l generam: amplitudine, faza, la un moment dat.

În etapa de rulare, numaratorul sau acumulatorul de faza se incrementeaza, cu o valoare programata, la fiecare nou impuls al semnalului de referinta. Iesirea acumulatorului de faza e folosita pentru a selecta datele din tabela inscrisa în memorie, iar în cele din urma se face conversia cuvântului binar în semnal analogic prin intermediul CNA. Sistemul poate genera frecvente mai mari la iesire prin marirea incrementului de faza, astfel încât numaratorul sa parcurga mai repede tabela.

Formula pentru calculul frecventei semnalului generat este:

$$F_{OUT}=(M*F_{CLK})/2^N$$

Deoarece DDS este digital iar frecventa si faza sunt determinate numeric, nu apar erori de la variatiile de temperatura sau de la îmbatrânirea componentelor.

1.2 Utilitatea generatorului de functii multiple

Generatorul despre care discutam în acest caz trebuie sa realizeze 3 tipuri de semnale:

- Semnal sinusoidal cu frecventa, amplitudinea si offset-ul ajustabile între limite date
- Semnal dreptunghiular cu frecventa variabila
- Semnal dreptunghiular cu frecventa si factor de umplere ajustabile (semnal PWM).

Ne referim în realitate la 2 generatoare de functii, întrucât pe o placa electronica vom realiza un generator de sinus, cu doua iesiri independente, care la rândul lor vor genera alte 2 iesiri: una diferentiala realizata cu un transformator 1:1 si o iesire simpla, si o a doua placa electronica ce va consti tui un generator de semnal dreptunghiular, cu doua iesiri: una cu factor de umplere de 1/2 si alta cu factor de umplere variabil.

Scopul realizarii acestor doua generatoare este testarea unitatilor de control electronice (ECU) a autovehiculelor dar si pentru simularea tahografelor folosite de masinile comerciale.

Pentru testarea unitatii de control a autovehiculului folosim semnalul sinusoidal, cu frecvente cuprinse între 0 si 10kHz si amplitudine variabila. În realitate testarea consta în transmiterea semnalului sinusoidal catre un microcontroller care detecteaza semiperioada de amplitudine pozitiva si semiperioada de amplitudine negativa a semnalului sinusoidal (adica high time si low time), comportându-se ca un senzor de viteza. De fapt prin transmiterea semnalului sinusoidal simulam deplasarea autovehiculului, si testam raspuns ul unitatii de control la acest sti mul.

Semnalul dreptunghiular este folosit pentru simularea alternatorului care are rolul de a transforma energia mecanica primita de la motor în energie electrica. La pornirea autovehiculului bateria genereaza învartirea motorului care determina alternatorul sa produca curent electric. Pentru a genera curent, alternatorul primeste energie mecanica de la motor si pe baza fenomenului de inductie magnetica, genereaza la bornele sale o tensiune electrica. Alternatorul propriu-zis se compune în esenta dintr-un rotor si un stator. Rotorul se numeste inductor si are rolul unui magnet permanent iar statorul se numeste indus, este bobinajul în care se induce curent electric prin variatia fluxului magnetic generat de rotor, variatie ce se realizeaza prin rotirea acestuia în vecinatatea statorului. Asadar pentru simularea alternatorului folosim semnalul dreptunghiular cu frecventa variabila între valorile 0 si 2 kHz.

Semnalul PWM simuleaza tahograful si se comporta ca si un senzor de viteza. Tahograful este un aparat de masura care combina functionarea unui timer cu cea a unui vitezometru. Conectat la motorul unui autovehicul, tahograful nu va înregistra doar viteza de deplasare a acestuia, ci si durata în timp a deplasarilor sau stationarilor. Tahografele sunt folosite la masinile comerciale, gen camioane, tiruri, autobuze pentru a monitoriza activitatiile soferilor, pentru a verifica daca pauzele sunt luate la timp si daca se respecta legislatia în vigoare. Tahografele au fost initial analogice si înregistrau datele despre deplasarea autovehiculului pe un disc de hartie. Acum s-au introdus tahografele digitale, care înregistreaza datele pe carduri personalizate, câte un card pentru fiecare sofer.Datele sunt digitale si codate. Semnalul nostru PWM va consti tui de fapt iesirea tahografului, si va functiona ca si un senzor de viteza.

În concluzie generatorul de functii multiple va fi utilizat pentru testarea unitatii de control a autovehiculului, iar semnalele generate de acesta vor simula deplasarea automobilului: semnalul sinusoidal va simula un senzor de viteza (deplasare), semnalul dreptunghiular va simula alternatorul (pornirea automobilului), iar semnalul PWM va simula un tahograf, deci tot un senzor de viteza. Pentru a respecta valorile pe care unitatea de control le primeste, trebuie ca semnalele generate sa respecte valorile mai sus mentionate.

Capitolul 2
Generarea semnalului sinusoidal

2.1 Cerinte

Sa se genereze un semnal sinusoidal cu frecventa, amplitudine si offset variabil, care sa se încadreze între valorile:

Nr.	Descriere	Simbol	Min.	Nom.	Max.	Unitate
1	Amplitudine	V_{AMPL}	0	-	10	V
2	Frecventa	f_{SINE}	0	-	10	KHz
3	Offset	V_{OFF_SINE}	1	-	5	V

Se vor genera 2 semnale sinusoidale independente, fiecare dintre ele având 2 cai:

- O cale diferentiala, care se va realiza cu un transformator 1:1
- O cale simpla, careia i se va putea modifica si offset-ul

Semnalul sinusoidal se va genera pe o singura placa, diferita de placa pe care vom genera semnalele dreptunghiulare si PWM.

Se doreste ca utilizatorul sa aiba o interfata care sa-i permita sa modifice frecventa, amplitudinea si offset-ul. Pentru aceeasta se pot folosi potentiometre la care utilizatorul sa aibe acces.

Se pot folosi circuite integrate sau circuite dedicâte care folosesc metoda DDS pentru obtinerea semnalului sinusoidal, ambele variante sunt acceptate. Se va avea în vedere faptul ca amplitudine semnalului trebuie sa varieze între 0 si 10 V, având o alimentare care porneste de la 8V. Se recomanda folosirea de stabilizatoare de tensiune.

2.2 Concept

Generatorul de semnal sinusoidal va trebui sa realizeze forme de unda cu frecvente variabile primind de la intrare niste date furnizate de utilizator, în speta ceea ce citeste generatorul de la potentiometrele care reprezinta interfata cu utilizatorul. Astfel ca, generatorul având o frecventa interna, va aplica acesteia diverse operatii determinate de ceea ce citeste de la potentiometrele de la intrare si va trece semnalul propriu prin diverse blocuri de circuite pentru a furniza la iesire semnalul sinusoidal variabil.

Alegem ca si principiu de realizare metoda DDS, de sinteza digitala directa, care consta în crearea unor forme de unda de frecvente arbitrare pornind de la o singura frecventa fixa. Este o metoda electronica, digitala care se realizeaza folosind un circuit ce are la baza un controller electronic, o memorie RAM, un oscilator (cu cuart de obicei) care genereaza frecventa fixa, un numarator/timer, si un convertor numeric-analogic (CNA).

Asa cum am mai spus, acest dispozitiv trebuie sa urmeze doi pasi pt a functiona: programarea si rularea.

Etapa de programare consta în încarcarea datelor în memorie de catre controller. Datele sunt în forma unor cuvinte binare care reprezinta o caracteristi ca a semnalului pe care vrem sa-l generam: amplitudine, faza, frecventa la un moment dat.

În etapa de rulare, numaratorul sau acumulatorul de faza se incrementeaza cu o valoare programata la fiecare nou impuls al semnalului de referinta. Iesirea acumulatorului de faza e folosita pentru a selecta datele din tabela inscrisa în memorie, iar în cele din urma se face conversia cuvântului binar în semnal analogic prin intermediul CNA. Sistemul poate genera frecvente mai mari la iesire prin marirea incrementului de faza, astfel încât numaratorul sa parcurga mai repede tabela.

Un astfel de principiu e simplu, iar paleta de integrate dedicâte care sa îndeplineasca aceasta functie e variata. Între urmatoarele integrate: XR-2206, AD5932, AD9833 disponibile în cadrul firmei în care s-a realizat proiectul, am ales ultimul circuit dedicat de la firma Analog Devices.

AD9833 este un circuit programabil care poate genera la iesirea sa diverse forme de unda : sinusoidale, dreptunghiulare, triunghiulare. Functionarea acestuia se bazeaza pe transpunerea caracteristi cii amplitudinii semnalului sinusoidal în caracteristi ca de faza, deoarece variatiile de amplitudine nu sunt liniare, pe când variatiile de faaa cu timpul sunt liniare, si periodice (la fiecare perioada a semnalului sinusoidal faza oscileaza între 0 si 2_p). Astfel ca, AD9833 va lucra cu variatii de faza, având scris în memoria sa interna o tabela cu valori de faza. În acest circuit scriem folosind o interfata seriala pe 3 fire, aceste fire reprezentând: SCLK (slave clock), SDATA(datele transmise propriu-zis pe 16 biti), FSYNC (semnalul de sincronizare între receptionarea datelor si procesarea lor).

AD9833 are în structura sa un oscilator controlat numeric, un modulator de faza si frecventa, un SIN ROM (o memorie în care se afla datele legate de faza) si un convertor numeric-analogic CNA. Iesirea oscilatorului controlat numeric reprezinta intrarea pentru citirea din tabela de sinus, unde vom transforma informatia de faza a semnalului sinusoidal (în continuare digitala), care prin CNA se transforma în semnal analogic sinusoidal.

Faptul ca AD9833 e comandat atat printr-un cuvânt binar, SDATA, cat si prin semnalele de sincronizare FSYNC si SCLK, sugereaza folosirea unui circuit inteligent care sa transforme semnalul de la intrare, adica ceea ce citim de la potentiometre, în cuvinte binare speciale care sa comande circuitul integrat dedicat. În acest sens optam pentru un microcontroller din gama foarte mare de microcontrollere prezenta pe piata alegem un PIC, produs de Microchip, care e un RISC

(Reduce Instruction Set Computer) si care poate fi programat cu usurinta. Pentru proiectul de fata alegem microcontrollerul PIC16F876.

.

PIC16F876 este un microcontroller cu 28 de pini, 3 porturi de intrare/iesire, cu 5 canale pentru conversie analog-digitala, memorie Flash de program, memorie EEPROM, cu comunicare seriala, si multe altele. în cadrul acestui proiect ne intereseaza ca PIC-ul sa fie interfata dintre utilizator (potentiometre) si modulul DDS. În acest sens, PIC-ul trebuie sa converteasca semnalul analogic receptionat la unul din porturile sale de intrare în semnal digital, sa îl prelucreze astfel încât sa comande AD9833. Modulul DDS trebuie comandat serial, ceea ce putem realiza cu PIC16F876, care are un modul SPI (Serial Peripheral Interface bus), cu 3 pini: Serial Data In, Serial Data Out, si Serial Clock. Astfel ca, pentru a realiza sincronicarea între cele doua integrate folosim Serial Clock, iar pentru transmiterea datelor necesare programarii modulului DDS folosim Serial Data Out.

Observam ca atât AD9833 cat si PIC16F876 se alimenteaza de la 5V, ceea ce inseamna ca trebuie sa folosim un stabilizator de tensiune pentru a realiza 5V de la sursa noastra de alimentare, care e bateria.

De asemenea, citind cu atentie foile de câta log ale AD9833 constatam ca nu putem modifica prin intermediul acestuia decat frecventa semnalului sinusoidal, întrucât integratul furnizeaza la exterior un semnal sinusoidal cu valoare vârf la vârf de 0,6V. Asta inseamna ca la iesirea AD9833 va trebui sa punem un bloc care amplifica si care face reglarea de nivel.
În acest punct, pentru semnalul sinusoidal am avea nevoie de amplificatoare operationale, si binenteles, pentru iesirea diferentiala vom folosi un transformator 1:1. Iar pentru a obtine 2 iesiri sinusoidale, vom folosi 2 circuite AD9833, însa un singur PIC, cu care sa le comandam pe amandoua.

În concluzie, generatorul de semnal sinusoidal va fi realizat din urmatoarele blocuri:

- Alimentarea, cu stabilizator de tensiune
- Microcontrollerul care comanda AD9833
- Blocul DDS, compus din 2 circuite AD9833
- Amplificatorul si regulatorul de nivel, unde vom face amplificare semnalului sinusoidal, ajustarea amplitudinii si vom regla offset-ul .

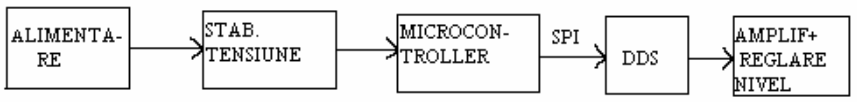


Fig.1 Schema bloc

În ceea ce priveste microcontrollerul utilizarea sa trebuie inteleasa în urmatoarea maniera: primind date la intrare de la potentiometre, el trebuie sa asocieze fiecarei valori de la intrare o alta frecventa pentru semnalul sinusoidal, pe care sa spunem ca o citeste dintr-o tabela definita în memoria sa EEPROM, apoi prelucreaza acel cuvânt binar, pentru al transforma în cuvânt de comanda pentru modulul DDS.

Etapele firesti pe care ar trebui sa le urmam în cadrul microcontrollerului sunt: citire de la intrarile analogice (corespunzatoare pentru cele 2 semnale sinus pe care vrem sa le generam),

conversia analog-numerica, poate si o filtrare a datelor, cu un filtru digital (de tip IIR sau FIR), citirea din tabela a cuvântului digital pe care trebuie sa-l transmitem catre AD9833, si binenteles transmisia prin SPI a cuvântului. De la microcontroller catre modulul DDS trebuie sa porneasca toate cele 3 semnale: FSYNC, SDATA, SCLK. Daca în ceea ce priveste semnalele SDATA si SCLK avem pini dedicati, în ceea ce priveste FSYNC acest lucru nu se mai întampla. Însa, cum FSYNK e un semnal activ pe zero câta vreme transmitem SDATA, si activ pe 1 când AD9833 nu receptioneaza nimic, ci doar proceseaza, asta inseamna ca putem folosi orice pin care poate fi iesire digitala în acest sens.

Cum am mai spus, iesirea DDS este un semnal sinusoidal cu amplitudine vârf la vârf de 0,6V care trebuie amplificat si la care trebuie reglat nivelul.

Amplificarea sa corespunde cu ajustarea amplitudinii, ceea ce inseamna ca în cadrul acestui bloc ar trebui sa avem potentiometrul pentru modificarea amplitudinii situat undeva în circuitul de reactie al amplificatorului operational, dar tot aici ar trebui sa realizam si ajustarea de offset. Înainte de a ajusta offsetul trebuie sa reglam nivelul semnalului si sa verificam daca e axat pe 0, pentru a modifica corect offset-ul.

În concluzie, semnalul sinusoidal cu frecventa, amplitudine si offset variabile îl obtinem folosind metoda de sinteza digitala directa, folosind un circuit integrat dedicat, AD9833, care e comandat de un microcontroller, PIC16F876, care interpreteaza datele de la un potentiometru, iar la iesire, semnalul sinusoidal e prelucrat si ii aplicam variatia de amplitudine si de offset.

2.3 Functionarea electronica

În cele ce urmeaza vom explica functionarea circuitului pe blocuri. asa cum am mai spus, avem în vedere 4 blocuri:

- Alimentarea
- Microcontroller
- DDS
- Amplificare si reglare de nivel

Întrucât pentru a realiza schema blocului de alimentare trebuie sa alegem toate circuitele integrate pentru a stii cu exactitate la ce tensiuni trebuie alimentate, vom lasa blocul de alimentare ultimul. În acest caz vom incepe prin a descrie celelalte blocuri.

2.3.1 Microcontrollerul

Am ales pentru realizarea generatorului de semnal sinusoidal PIC16F876, pentru comandarea blocului DDS. Un astfel de microcontroller are ca si avantaj modulul de SPI (Serial Peripheral Interface bus), de care avem nevoie pentru AD9833, dar si faptul ca are incorporat un convertor analog-numeric care sa transforme în cuvinte binare semnalele receptionate de la intrare.

Vom conecta la 2 pini care pot fi folositi ca si intrari analogice 2 potentiometre pe care le vom folosi ca sa variem frecvent ele celor doua semnale sinusoidale pe care vrem sa le obtinem.

Alimentarea microcontrollerului

Tensiunea de alimentare pentru un astfel de microcontroller trebuie sa fie cuprinsa între 2V si 5,5V. Noi alegem o tensiune standard de 5 V cu care sa alimentam microcontrollerul.
Tot în cadrul alimentarii vorbim si despre pinii de V_{DD} (pinul 20)si V_{SS} (pinii 8 si 19) care se pot folosi si ca tensiuni de referinta în cadrul operatiilor pe care le face integratul. V_{DD} se leaga la alimentarea de 5V, conectand la acest pin si un condensator pentru protectia PIC-ului la variatii de tensiune, iar V_{SS} se va conecta la masa, fiind referinta pentru 0V.

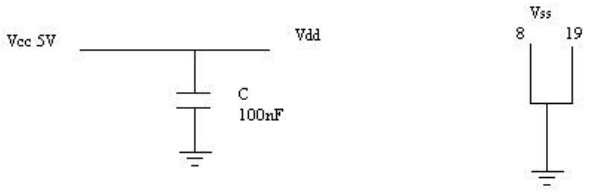


Fig.2 Mod de conectare a V_{DD} si V_{SS}

Pentru conectarea V_{DD} alegem un condensator de valoare 100nF, recomandat de producator.

De asemenea mai avem un pin MCLK/Vpp pe care îl folosim ca si intrare de tensiune de programare si care va fi conectat la tensiunea de alimentare V_{DD} de 5V, însa care necesita un circuit de protectie, realizat din doua rezistente si un condensator, asa cum este el recomandat de producator. Pentru rezistente sunt recomandate valorile:
R1< 40k? , noi alegem R1=10k?
R2 = 1k? , noi alegem chiar valoarea de 1k?
Iar pentru C alegem o valoare standard de 100nF, asa cum recomanda producatorul la celelalte condensatoare pentru protectie .

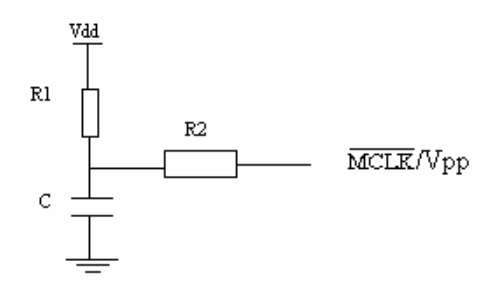


Fig.3 Mod de conectare a pinului MCLK/Vpp la alimentare

Bloc oscilator pentru PIC16F876

Pentru ca cele 3 timere ale microcontrolerului sa functioneze trebuie sa conectam PIC-ul la un timer extern sau la un oscilator cu cristal. Citind foile de catalog ale PIC16F876 referitoare la frecventa cu care merg timerele raportat la oscilatorul (adica frecventa interna a microcontrollerului), care e Fosc/4, respectiv durata unui ciclu masina 4/Fosc, alegem un oscilator cu o frecvent a de oscilatie de 4MHz, care va determina o durata a ciclului masina de 1us.
În acest sens alegem un rezonator de 4MHz , QMIM004, pe care îl vom conecta asa cum ne recomanda producatorii:

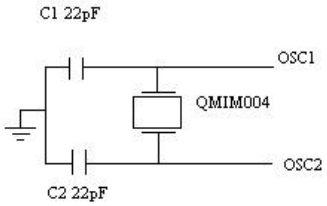


Fig.4 Conectarea oscilatorului la PIC

OSC1 si OSC2 sunt pinii 9 si 10 ai microcontrollerului la care conectam oscilatorul.

Porturile microcontrollerului

PIC16F876 are 3 porturi: PORTA, PORTB, PORTC. Vom folosi toate cele trei porturi pentru a comanda modulul DDS, deoarece avem nevoie de intrari analogice, iesiri digitale si SPI.

PORTA este portul în cadrul caruia putem seta pana la 5 pini pentru intrari analogice din 6 pini disponibili. Noi avem nevoie doar de doua intrari analogice, care vin de la potentiometrele asupra carora actioneaza utilizatorul. În acest sens vom seta pinii RA0/AN0 si RA1/AN1, adica pinii 1 si 2 ai PIC-ului. PORTA are ca si corespundent un registru TRISA, prin intermediul caruia putem seta pinii corespunzatori acestui port ca si intrari sau ca si iesiri punând bitul corespunzator pinului pe care vrem sa-l setam pe 1 pentru intrare si pe 0 pentru iesire.

În cazul nostru, pentru a seta pinii RA0 si RA1 ca si intrari analogice vom seta bitii 0 si 1 din cadrul registrului TRISA pe 1 si restul bitilor pe 0.

Pentru iesirea de SPI folosim PORTC,care are un modul dedicat pentru SPI, adica are disponibili urmatorii pini: SDI (Serial Data Input), SDO (Serial Data Output), SCK (Serial Clock), corespunzatori pinilor: RC4, RC5, respectiv RC3.

Lucram în modul MASTER, adica SCK este furnizat de PIC, tot el transmite si SDO, iar AD9833 functioneaza în modul SLAVE, deci se sincronizeaza cu PIC-ul, folosind ca si clock extern SCK. Asta inseamna ca pinul de SDI nu îl folosim si ca SDO si SCK sunt iesiri digitale. Adica, setam bitii 3 si 5 ai TRISC pe 0 logic.

De asemenea pentru a comanda modulul DDS mai avem nevoie de doua semnale de FSYNC, câte unul pentru fiecare AD9833, care în realitate sunt cuvinte binare, 0 logic câta vreme facem transmisia de date de la PIC la AD9833 si 1 logic când modulul DDS facem procesarea de date. În realitate vom transmite date pe la pinul de SDO în permanenta, însa în functie de cum comandam FSYNC1 si FSYNC2 selectam circuitul AD9833 care vrem sa faca procesarea de date. În felul acesta interpretarea datelor de la cele 2 potentiometre se va face intermitent, si transmiterea datelor la modulele DDS se va face tot pe rând, caci vom avea grija ca prin program, când FSYNC1 este pe 0, FSYNC2 sa fie pe 1 si invers, asa încât modulul DDS 1 sa prelucreze doar datele provenite de la potentiometrul 1 si nu si de la potentiometrul 2.

Înseamna ca avem nevoie de doua iesiri digitale pentru FSYNC1, 2 si alegem pentru aceasta PORTB, care are 8 pini ce se pot folosi în acest scop. Noi alegem pinii RB0 si RB1 ca si iesiri digitale si ii setam punând pe 0 bitii 0 si 1 ai registrului corespondent TRISB.

Convertorul analog-numeric

PIC16F876 beneficiaza de un convertor analog-numeric (CAN) cu 5 intrari analogice, a carui valoare numerica corespondenta semnalului analogic de la intrare este un numar binar pe 10 biti, ce se scrie în 2 registre ADRESL si ADRESH, 8 biti într-un registru si 2 în altul, selectabil din software.

CAN face conversia având ca referinta doua tensiuni care pot fi V_{DD} (high) V_{SS}(low), sau tensiuni citite de la pinii RA3 si RA4, în orice fel de combinatii. Pentru circuitul de fata alegem ca referinta tensiuniile de 5V high si 0V low, adica V_{DD} si V_{SS} .

Pentru selectia canalelor 0 si 1 analogice trebuie sa programam registrele ce control ai CAN: ADCON0 si ADCON1, asa cum se va vedea în cadrul programului aplicat PIC-ului.
Timpul de achizitie pentru convertorul analog-numeric în cadrul acestui integrat este de aproximativ 20us, asa cum ne informeaza producatorul (nota bibliografica Anexa 4*)

TACQ	=	Amplifier Settling Time + Hold Capacitor Charging Time + Temperature Coefficient
	=	TAMP + TC + TCOFF
	=	2 µs + TC + [(Temperature – 25°C)(0.05 µs/°C)]
TC	=	CHOLD (RIC + RSS + RS) In(1/2047)
	=	- 120 pF (1 kΩ + 7 kΩ + 10 kΩ) In(0.0004885)
	=	16.47 µs
TACQ	=	2 µs + 16.47 µs + [(50°C – 25°C)(0.05 µs/°C)
	=	19.72 µs

Cum CAN-ul este pe 10 biti, inseamna ca la iesirea sa vom avea de fapt maxim 2^{10} valori, adica 1024 valori diferite pentru ceea ce citim de la intrarile analogice, de la potentiometre. Putem spune, pentru usurarea calculelor, ca acest CAN lucreaza cu o frecventa de esantionare de 1kHz, deci ca avem de fapt doar 1000 de valori corespunzatoare variatiilor de tensiune de la intrare si ca la fiecare 1ms avem o conversie analog-numerica, deci la fiecare 1ms avem o alta valoare în registrele ADRESL si ADRESH.

Formula dupa care se face conversia este:

$$U_{IN}= [(U_{REF+} +U_{REF-})/1000]*N- U_{REF-}$$

unde $U_{REF+} =V_{DD}=5V$

$U_{REF-} =Vss =0V$

N= numar ul returnat de CAN însa în baza zece

În cele ce urmeaza vom folosi acest numar returnat de CAN ca un pointer la o adresa în memoria program unde vom stoca datele pe care trebuie sa le trimitem prin SPI la AD9833 astfel încât sa generam semnalul sinusoidal cu frecventa impusa de pozitia potentiometrului. Datele care trebuie trimise la modulul DDS le vom determina atunci când vom discuta despre AD9833.

Dupa ce citim aceste date din memoria program EEPROM a PIC-ului, si înainte de a le transmite spre AD9833 trebuie sa filtram digital datele, pentru a fi siguri ca nu transmitem si componente nedorite, care sa ne distorsioneze semnalul sinusoidal.

Filtrarea IIR

Asa cum am spus anterior, datele preluate din memoria de program trebuie filtrate înainte de a le transmite la modulul DDS. Fiind vorba despre procesare de semnal digital, putem utiliza, la alegere, filtru IIR (Infinit Impulse Response) sau filtru FIR (Finit Impulse Response). Filtrul IIR are o functie de raspuns la impuls diferita de zero pe o durata în timp infinit de lunga. Spre deosebire de acesta, filtru FIR are functie de raspuns la impuls diferita de zero pe o perioada finita. Filtrele IIR pot fi atât analogice-cel mai simplu filtru este un circuit RC- sau digitale.De mentionat faptul ca în cadrul filtreler IIR digitale, iesirea anterioara a filtrului influenteaza în mod direct valoarea viitoare a iesirii acestuia. Spre deosebire de filtrele FIR, la IIR e necesar sa definim momentul t=0 pentru care valoarea iesirii nu e clar definita.

Pentru cazul de fata, implementam un filtru IIR de ordinul 1, care ar trebui sa se implementeze cu o formula de forma:

$$Yn= (1-a_1)\, Y_{n-1}+ (b_0X_n+ b_1X_{n-1})$$

Pentru calcularea coeficientiilor filtrului folosim programul MathCad, în care ca si date introducem frecventa de esantionare si frecventa de taiere, iar tipul filtrului e Cebîsev.

Deoarece datele pe care le filtram sunt de la iesirea CAN-ului atunci consideram frecventa de esantionare de 1kHz, atât cât e frecventa de esantionare a CAN-ului. În ceea ce priveste frecventa de taiere consideram o frecventa de 10Hz, de aici rezultând o perioada de 100ms.

$fc := 10$ frecventa de taiere

$Fs := 1000$ Frecventa de esantionare(sampling)

$Ts := \frac{1}{Fs}$ perioada de esantionare

$Ts = 1 \times 10^{-3}$

$i := \sqrt{-1}$

$H1(\omega) := \left| s \leftarrow i\cdot\omega \;\middle|\; \frac{2\cdot\pi\cdot fc}{s + 2\cdot\pi\cdot fc} \right|$ functia de transfer a unui filtru trece jos cu frecventa de taiere fc

$$\Phi 1blt(\omega) := \left| \begin{array}{l} z \leftarrow \cos(\omega \cdot Ts) + i \cdot \sin(\omega \cdot Ts) \\ s \leftarrow \frac{2}{Ts} \cdot \frac{z-1}{z+1} \\ \arg\left(\frac{2 \cdot \pi \cdot fc}{s + 2 \cdot \pi \cdot fc}\right) \end{array} \right.$$

Functia de transfer a unui filtru IIR cu transformata biliniara (o aproximare a unui filtru analogic de ordin 1 cu frecv de taiere fc) cu o frecventa de esantionare Fs

$$a1 := \frac{\pi \cdot fc \cdot Ts - 1}{1 + \pi \cdot fc \cdot Ts} \qquad a1 = -0.93908$$

Coeficientii ecuatiei diferentiale a unui filltru IIR de ordinul 1

$$b0 := \frac{\pi \cdot fc \cdot Ts}{1 + \pi \cdot fc \cdot Ts} \qquad b0 = 0.03046$$

$$b1 := \frac{\pi \cdot fc \cdot Ts}{1 + \pi \cdot fc \cdot Ts} \qquad b1 = 0.03046$$

$$n_a := \frac{-\log(1 + a1)}{\log(2)} \qquad n_a = 4.037 \qquad n_{a\_ch} := 4$$

aproximare digitala

$$n_{b\_c} := \frac{-\log(b0)}{\log(2)} \qquad n_{b\_c} = 5.037 \qquad n_{b\_c\_ch} := 5$$

$$a_{1\_aprox} := -1 + \frac{1}{2^{n_{a\_ch}}} \qquad a_{1\_aprox} = -0.938$$

Coeficientii ecuatiei diferentiale a unui filltru IIR de ordinul 1

$$b_{0\_1\_aprox} := \frac{1}{2^{n_{b\_c\_ch}}} \qquad b_{0\_1\_aprox} = 0.031$$

$$a1\_ap := -2^{-1} - 2^{-2} - 2^{-3} - 2^{-4} \qquad a1\_ap = -0.938$$

valori aproximative, verificam calculul

$$b0\_ap := 2^{-5} \qquad b0\_ap = 0.0313$$

$$b1\_ap := 2^{-5} \qquad b1\_ap = 0.0313$$

$$f_{cT} := \frac{1}{\pi \cdot Ts} \cdot \frac{1 + a_{1\_aprox}}{1 - a_{1\_aprox}} \qquad f_{cT} = 10.268$$

valoarea reala a frecventei de taiere

$$HT(\omega) := \left| \begin{array}{l} z \leftarrow e^{i \cdot \omega \cdot Ts} \\ \left| \frac{b0\_ap + b1\_ap \cdot z^{-1}}{1 + a1\_ap \cdot z^{-1}} \right| \end{array} \right.$$

proiectarea filtrului digital

$$HT(2 \cdot \pi \cdot 0) = 1$$

$$\omega := 0, 2 .. 10^5$$

timpul de raspuns al filtrului

$$t_{filt\_resp} := \frac{1}{2 \cdot \pi \cdot fc \cdot Hz} \cdot 5$$

$$t_{filt\_resp} = 79.577\ ms$$

Caracteristi ca de transfer a filtrului, pe care din program l-am ales Cebîsev este:

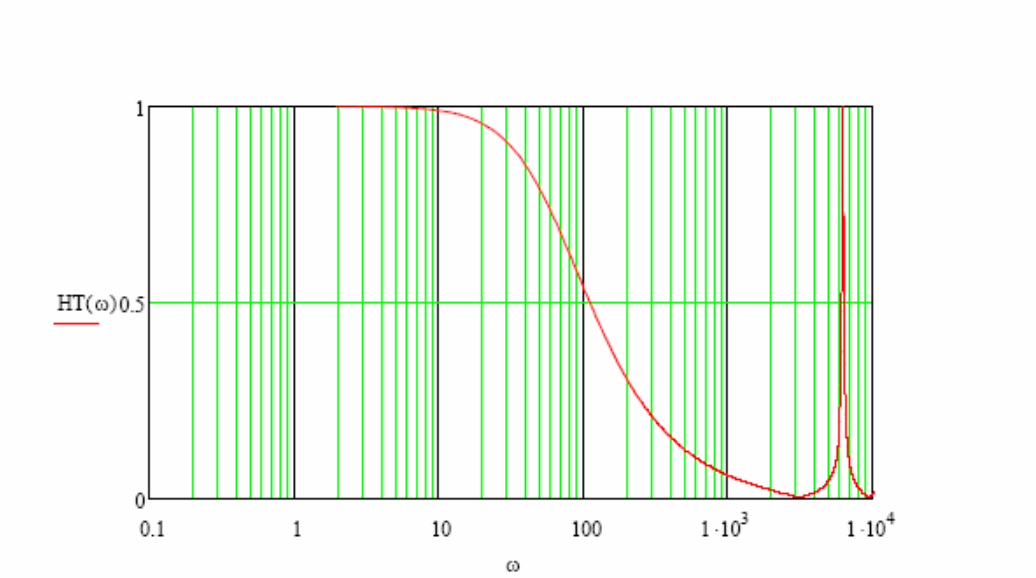


Fig.5 Caracteristi ca de transfer a filtrului IIR Cebîsev

Dupa ce am calculat coeficientii putem sa scriem ecuatia filtrului IIR de ordinul 1, corespunzator unui filtru analogic trece jos:

$$Y_n=(1- 2^{-4})Y_{n-1}+ 2^{-5}(X_n+X_{n-1})$$

Implementarea filtrului se va face din program, tinând însa cont sa definim timpul zero: t=0 si sa initializam valorile initiale ale filtrului.

Asa cum am calculat mai sus, timpul de raspuns al filtrului IIR este de 80ms. Asta inseamna ca, daca de la CAN se citesc valori la fiecare 1ms, doar a 80-a valoare va fi filtrata, ceea ce inseamna ca la SPI o sa transmitem datele filtrate la fiecare 80ms.

Modulul SPI

Modulul de SPI îl folosim ca sa comandam AD9833, asta înseamna ca o sa folosim modulul SPI din cadrul PIC16F876 în modul MASTER, adica clock-ul provenit de la PIC este cel care va comanda modulul DDS. Asa cum am discutat anterior, modulul SPI este în cadrul PORTC, pe care am decis anterior cum sa-l programam. Pentru a stii cu exactitate ce sa transmitem prin modulul SPI trebuie sa citim înainte cu atentie foile de catalog ale AD9833. Important de retinut în acest punct

este faptul ca transmitem datele prin SDO, iar selectia unuia dintre circuitele AD9833 pe care le folosim o facem prin iesirile FSYNC1 si FSYNC2.

Schema electronica

Schema bloc a partii de microcontroller, daca respectam toate cerintele impuse pe tema proiectului si de recomandarile producatorului este:

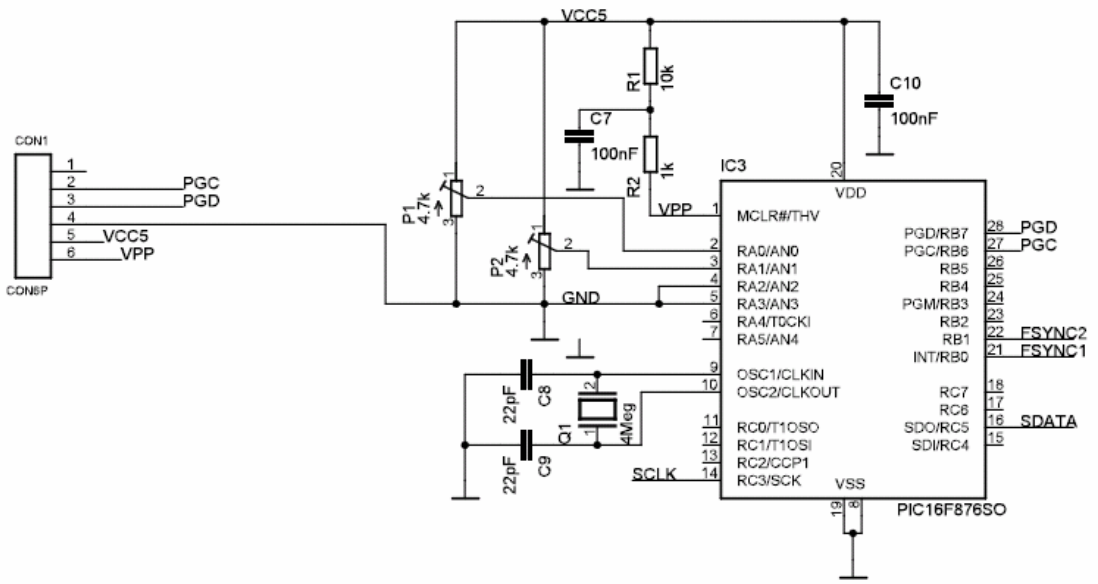


Fig. 6 Schema blocului de microcontroller

2.3.2 Modulul DDS

Modulul DDS este realizat din 2 circuite AD9833, câte unul pentru fiecare semnal sinusoidal pe care vrem sa-l generam si alte circuite pentru protectia celor 2 integrate, care ne sunt recomandate de producator, Analog Devices.

AD9833 este un circuit specializat care poate genera semnale sinusoidale, dreptunghiulare si triunghiulare, pe noi interesându-ne numai semnalul sinusoidal. Functionarea acestuia se bazeaza pe transpunerea caracteristicii amplitudinii semnalului sinusoidal în caracteristica de faza, deoarece variatiile de amplitudine nu sunt liniare, pe când variatiile de faza cu timpul sunt liniare, si periodice.

Alimentarea

Circuitul AD9833 se alimenteaza cu tensiune între 2,3V si 5,5V. Pentru a folosi aceeasi alimentare cu microcontrollerul, vom alimenta si acest circuit de la 5V. Pentru a întelege mai bine cum trebuie conectati pinii unui astfel de circuit pornim de la configuratia pinilor:

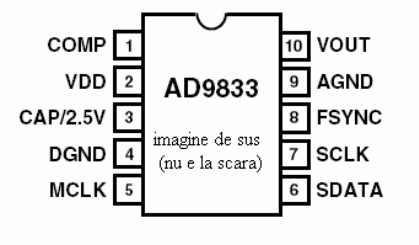


Fig.7 Configuratia pinilor la AD9833

Asadar V_{DD} îl conectam la 5V, si între V_{DD} si AGND legam o retea de capacitati de decuplare cu valorile 0.1uF si 10uF.

CAP/2.5V îl legam la DGND printr-o capacitate de 100nF, daca V_{DD} e mai mare decat 2,7V, dupa cum recomanda producatorul, iar COMP (punct de decuplare a tensiunii BIAS a convertorului numeric-analogic) îl legam printr-o capacitate de 100nF la V_{DD}.

Blocul oscilator

Daca ne uitam la configuratia pinilor circuitului AD9833 observam un pin MCLK, adica Master Clock pe care producatorul ne recomanda sa-l conectam la un oscilator cu cuart de 25MHz, care determina acuratetea frecventei semnalului generat, dar si zgomotul de faza al acestuia. Alegem oscilatorul de 25 MHz SG-51/P , pe care îl vom conecta la 5V printr-un condensator de 100nF ca în figura:

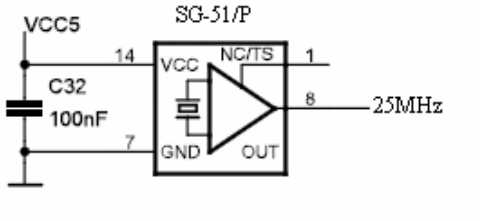


Fig.8 Bloc oscilator pentru AD9833

Principiu de functionare

Circuitul intern al AD9833 este realizat din câteva blocuri importante: oscilatorul controlat numeric, modulator de faza si frecventa, SIN-ROM (o memorie în care e stocata informatia de sinus), convertor numeric-analogic. Cea mai importanta componenta a oscilatorului comandat numeric e acumulatorul de faza, un registru pe 28 de biti. Semnalul sinusoidal are un domeniu de faza de la 0 la 2p , asta inseamna ca, folosind acumulatorul de faza pe 28 de biti, avem $2p= 2^{28}$.

Formula prin care calculam frecventa semnalului sinusoidal generat este :

$$f = P * f_{MCLK}/2^{28}$$

unde P= valoare din acumulatorul de faza care determina frecventa, si care are valori cuprinse între 0 si 2^{28}-1, f_{MCLK} are valoarea de 25 MHz iar f este frecventa semnalului sinusoidal.

Ideea de functionare a acestui circuit dedicat este de a transmite prin SPI de la microcontroller la acumulatorul de faza numarul potrivit care sa genereze semnalele sinusoidale la

frecventele dorite. PIC-ul trebuie sa aibe în memoria sa de program inscrisa o tabela cu valori corespunzatoare fiecarei variatii de frecventa înregistrata de la potentiometru, si cum CAN-ul transforma datele de la intrarea analogica în date digitale, numarul de la iesirea CAN-ului va contitui un pointer la adresa din memorie unde avem memorata tabela.

Pentru aceasta incercam sa determinam un algoritm cu care sa calculam numarul care trebuie transmis de SPI catre registrul acumulator de faza, adica numarul P.
Calculam numarul care trebuie transmis pentru generarea semnalului de frecventa 10kHz (maximul de frecventa care ni se cere la semnalul sinusoidal):

$$P = (10*10^3*2^{28}) / 25*10^6$$
$$P= 107.374= 1A36E_{hexa}$$

Iar pentru o frecventa de 10hz:

$$P = (10*2^{28}) / 25*10^6$$
$$P= 107= 6B_{hexa}$$

Deoarece am stabilit la paragraful anterior, referitor la microcontroller ca în cadrul PIC16F876 convertorul analog-numeric transforma semnalul analogic în semnal digital cu frecventa de esantionare de 1kHz, si asta inseamna o valoare noua în binar la fiecare 10Hz, atunci facem calculul pentru numarul pe care trebuie sa-l transmitem prin SPI la frecvente multiplu de 10Hz.
Mai facem un calcul si pentru o frecventa de 20Hz:

$$P = (2*10*2^{28}) / 25*10^6$$
$$P= 214= 107*2$$

De aici putem deduce urmatoarea regula :

$$P= 107*N$$

unde N este valoarea în zecimal de la iesirea CAN-ului, care ia valori între 0 si 1000.

Trebuie sa fim atenti la respectarea numarului de biti pentru fiecare dintre registrele PIC16F876. Cum noi memoram aceste date în cadrul memoriei Flash de program, si aici putem retine date pe maxim 14 biti, iar pentru valoarea maxima a unui numar de comanda pentru AD9833 este de 17 biti, atunci suntem nevoiti sa scriem datele pe care vrem sa le transmitem prin modulul SPI la 2 locatii diferite, la una Least Significant Word (cei mai putin semnificativi 14 biti) si la alta locatie Most Significant Word (Cei mai semnificativi 3 biti). Alegem ca si locatii pentru memorarea datelor adresele incepând cu 1800h pentru Least Significant Word si 1C00H, Most Significant Word.

Revenind la functionarea modulului DDS, pâna acum am stabilit ca acesta primeste de la PIC prin SPI cuvinte de comanda care determina valoarea frecventei semnalului sinusoidal. Determinarea frecventei se stabileste printr-o serie de operatii pe care le faceAD9833, care includ: citirea valorii primite de la SPI, transmiterea numarului asociat la SIN-ROM,care e o tabela interna circuitului si la care nu avem acces ca si utilizatori si unde se face asocierea între numarul corespunzator fazei si valoarea sinusoidala digitala, conversia numeric analogica si transmiterea semnalului sinusoidal la iesirea sa, V_{OUT}.

Un fapt important de retinut este transmisia corecta a semnalului de selectie a circuitului care genereaza semnalul sinusoidal, si anume FSYNC1 si 2. FSYNC se transmite de la PIC16F876 catre AD9833, si câta vreme acesta are valoarea 0 logica, AD9833 receptioneaza date, iar cât timp FSYNC are valoarea 1 logic, AD9833 proceseaza datele. Trebuie sa tinem cont de acest fapt si sa facem transmisia de date catre modulul DDS intermitent, când catre un AD9833, când catre celalalt, punând când un FSYNC pe 0 când pe celalalt.

Verificând cu atentie foile de catalog de la AD9833 observam ca trebuie ca la fiecare transmisie catre AD9833 sa transmitem o secventa de 16 biti de control, alta secventa de 16 biti, din care primii 2 MSB (most significant bit) sunt 01, urmati de cei 14 biti ai Most Significant Word si înca o secventa de 16 biti, primii 2 MSB fiind 01 si ultimii 14 sunt Least Significant Word din memoria Flash a PIC16F876. Asta insemna ca la fiecare 80ms, când facem filtrarea datelor si citirea

lor din tabelele din memoria Flash trebuie sa transmitem catre AD9833 o structura de date mai complexe decat ceea ce citim din tabela si anume:

0010 0000 0000 0000 cuvânt de control al scrierii
01 Most Significant Word
01 Least Significant Word

Schema electronica a modulului DDS

Blocul de sinteza digitala directa este cel mai important bloc functional din cadrul acestui generator de semnal, deoarece în cadrul acestui bloc se realizeaza propriu-zis semnalul sinusoidal. Acest bloc este realizat din 2 circuite AD9833, pentru cele 2 semnale sinusoidale independente pe care trebuie sa le generam , un bloc oscilator care genereaza o frecventa de 25MHz necesara pentru determinarea acuratetii frecventei semnalului, si binenteles circuitele de protectie pentru AD9833 recomandate de producator. Schema acestui bloc este:

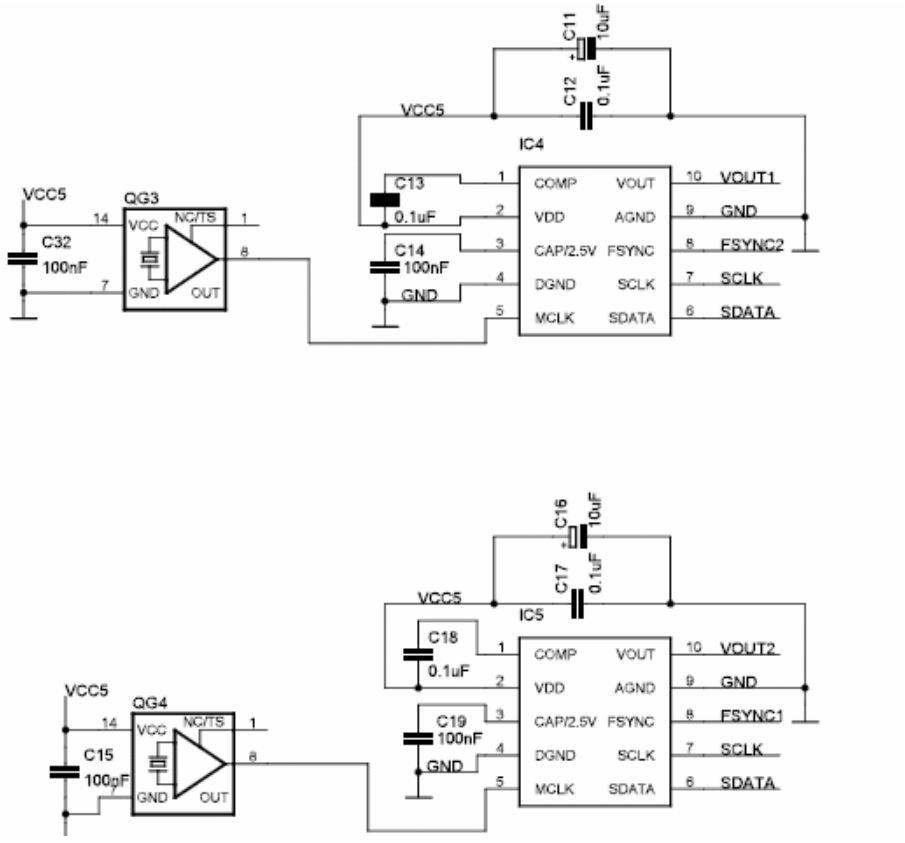


Fig.9 Modulul DDS

2.3.3 Blocul de amplificare si reglare a nivelului

La iesirea modulului DDS avem un semnal sinusoidal cu amplitudine vârf la vârf de 0,6V. De aceea avem nevoie de un bloc unde sa realizam si amplificarea semnalului, dar si variatia amplificarii si a offset-ului, si totodata un etaj final unde sa stabilizam semnalul.

Pentru amplificare folosim amplificatoare operationale, LM2904, care sunt de fapt 2 amplificatoare intr-o capsula, si care integrat se alimenteaza de la o tensiune de 15V.

Amplificarea se determina din raportul rezistentelor din circuitul de reactie al amplificatorului, si cum noi trebuie sa asiguram o amplitudine maxima de 10V vârf la vârf dintr-un semnal de 0,6V vârf la vârf, inseamna ca trebuie sa asiguram o amplificare de :

$$10/0,6 \tilde{} \ 16$$

ceea ce insemna ca în circuitul de reactie trebuie ca folosim rezistente care sa prezinte un asemenea raport între ele ca sa poata asigura o amplificare de 16 ori a semnalului de la iesirea blocului DDS. daca punem în circuitul de reactie o rezistenta R_r si la intrarea negativa punem o rezistenta R, atunci amplificarea va avea valoarea:

$$A= 1+R_r/R$$

ceea ce inseamna ca pentru a obtine o amplificare de 16, trebuie ca raportul celor 2 rezistente sa fie 15. Alegem în acest mod rezistenta de reactie de 15kO si cealalta rezistenta de 1kO.
Schema pentru partea de amplificare si modificare a amplitudinii si a offset-ului este:

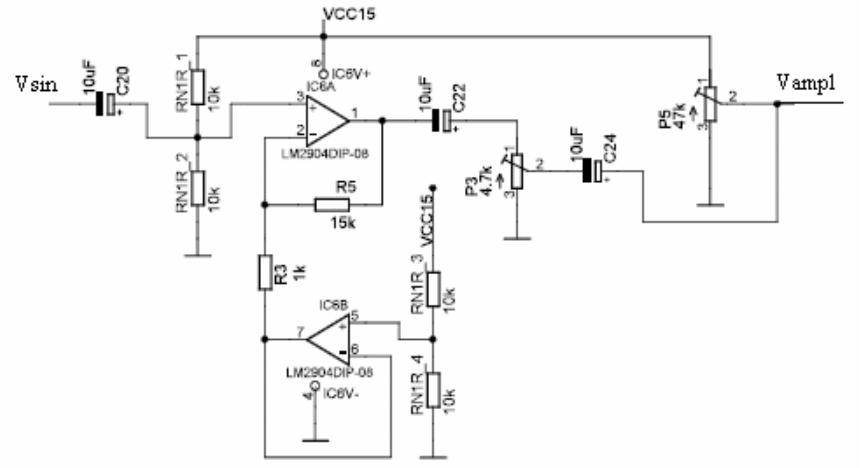


Fig. 10 Schema de amplificare a semnalului si modificare a amplitudinii si offset-ului

Functionarea blocului de amplificare

În primul rând, semnalului care vine de la AD9833 trebuie sa-i eliminam componenta continua, si pentru asta folosim un condensator tantalum de valoare 10uF.

Încercam o structura prin care sa scadem valoarea tensiunii de alimentare care e de 15V si de aceea vom folosi 2 amplificatoare operationale, al doilea doar ca repetor, iar la intrarea sa pozitiva

vom pune o retea de rezistente care au aceeasi valoare si pe care le folosim cu scopul de a injumatatii valoarea medie a semnalului. Alegem pentru aceste rezistente valoarea de 10kO. Folosim o retea similara de rezistente pentru intrarea pozitiva a amplificatorului operational ce realizeaza amplificarea, iar pe intrarea sa negativa si pe circuitul de reactie punem cele doua rezistente a caror valoare am calculat-o deja: 1kO, respectiv 15kO.

La iesirea acestui bloc de amplificare suntem nevoiti din nou sa eliminam componenta continua, prin pozitionarea în circuit a unui condensator identic cu cel de la intrarea în bloc. Semnalul sinusoidal pe care l-am obtinut astfel ar trebui sa fie axat pe 0 si sa aibe valoarea vârf la vârf de maxim 10V. La acest semnal legam un potentiometru cu care vom varia amplitudinea semnalului. Valoarea potentiometrului se alege de 4,7kO astfel încât sa putem obtine variatia amplitudinii de la 0 la 10V vârf la vârf.

Mai facem o eliminare a componentei continue, tot cu un condensator de aceeasi valoare ca si precedentele. Iar pentru a obtine o variatie a offset-ului conectam un potentiometru la sursa de alimentare, punându-l într-un nod comun cu semnalul la care am variat amplitudinea, iesirea din nod fiind semnalul sinusoidal la care modificam atat amplitudinea cat si offset-ul. Alegem pentru acesta o valoare de 47kO, astfel încât sa variem offset-ul cu valori cuprinse între 1 si 5V.

Functionarea etajului final

Etajul final are un rol important deoarece face eliminarea zgomotelor si aduce la un anumit nivel semnalul generat. Pentru etajul final alegem o structura care e folosita si la alte generatoare de functii si care are rolul de a regla nivelul semanlului generat.

Pentru etajul final folosit la generatorul nostru de functii alegem o structura cu amplificatoare operationale în structura repetoare si binenteles o structura de tranzistoare, cu o retea de rezistente de valoare mica, de 100O, fiind vorba despre un etaj final.

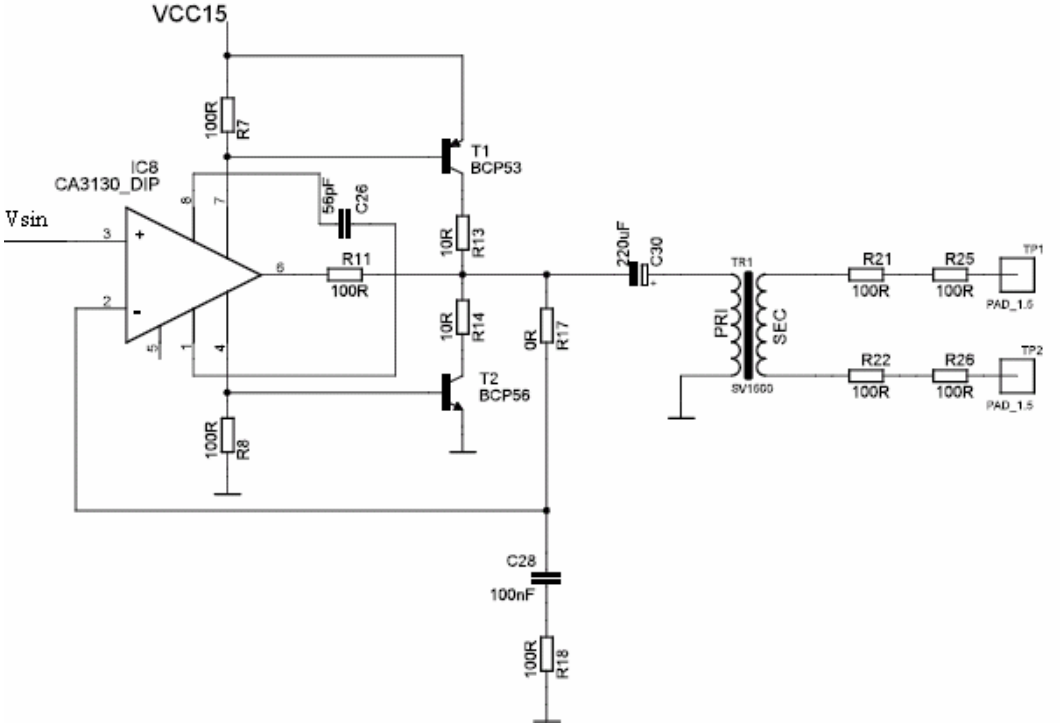


Fig.11 Schema blocului final

Amplificatoarele operationale alese sunt CA3130 care se alimenteaza de la o tensiune de 15V, la fel ca amplificatoarele folosite la blocul de amplificare.

Alegem o structura de tranzistori npn si pnp, si condensatori care sa elimine componenta continua (C_{30} =220uF) si condensatori care sa protejeze circuitul (C_{28}=100nF). Binenteles vom pune la iesirea circuitului, dupa stabilizarea semnalului, si transformatorul care sa realizeze iesirea diferentiala. Punem la iesirea transformatorului 1:1 si doua rezistente de valoare mica pentru protectie .

2.3.4 Blocul de alimentare

Parcurgând subcapitolele anterioare stim ca trebuie sa alimentam mai multe integrate cu tensiuni de 5 si 15V. Pentru aceasta e nevoie de stabilizatoare de tensiune, care sa obtina tensiune de 5, respectiv 15V de la baterie.

Alegem stabilizatoarele LM7805 pentru 5V si LM7815 pentru 15V. Producatorul recomanda o structura cu stabilizatorul încadrat de doua condensatoare de valoare 100nF pentru protectie . De asemenea mai folosim si o dioda de la baterie pana la stabilizator, care sa ofere protectie stabilizatorului la variatii mari de curent. La iesirea blocului de alimentare punem un condensator electrolitic tantalum (metal tranzistiv foarte usor) pentru a stoca sarcini si a modera tensiunea la iesirea si pentru a se opune la fluctuatii mari de curent si tensiune, si pentru a asigura o tensiune stabilizata. Alegem pentru acesta o valoare standard de 10uF

Structura va fi aceeasi si pentru stabilizatorul de 5V si pentru cel de 15V. Schema pentru blocul de alimentare este:

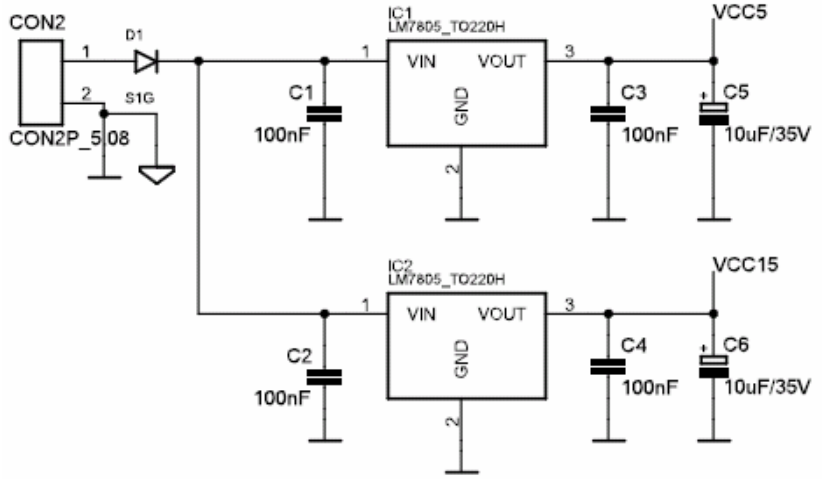


Fig.12 Blocul de alimentare

Cu CON2 notam conectorii de pe placa electronica unde vom conecta bateria. În ceea ce priveste dioda, alegem o dioda S1G, care poate face fata la variatii de curent de 1A.
VCC5 si VCC15 sunt iesirile acestui bloc care sunt de 5, respectiv 15V si care merg la alimentarea microcontrollerului si a modulului DDS, respectiv la alimentarea amplificatoarelor operationale.

2.3.5 Schema electronica a generatorului de semnal sinusoidal

Schema electronica a generatorului de semnal sinusoidal cuprinde toate blocurile pe care le-am amintit anterior, si anume: blocul de alimentare, microcontrollerul, modulul de sinteza digitala directa, si blocul de amplificare si reglare a nivelului.

Schema elctronica sa poate gasi la rubrica Anexe, ca si Anexa1.

2.3.6 PCB-ul circuitului proiectat

PCB-ul (Printed Circuit Board) este designul propriu-zis al circuitului electronic, si se refera la asezarea componentelor pe placa, astfel încât diversele componente din circuit sa nu se interinfluenteze. Pentru asezarea componentelor pe placa folosim doua straturi si doua tipuri de tehnologii: SMT (surface mount technology) si THT (through-hole mount technology). SDT consta în lipirea componentelor cu o pasta speciala pe suprafata placii, iar în felul acesta pot fi folosite ambele fete ale unei placi, pe când THT consta în pozitionarea pinilor componentelor prin orificii destinate special acestei actiuni. Designul PCB-ului consta în trasarea linilor de conectivitate între componente, linii care se pot trasa pe mai multe straturi, nu doar pe cele 2 pe care le putem fizic vedea pe placa (fata-verso), ci si în interiorul placii, prin suprapunerea propriu-zisa a mai multor placi. Acest din urma mod de a trasa linile de conexiune se foloseste în cazul unui numar mare de componente, care trebuie pozitionate pe o placa de mici dimensiuni.

Pentru acest generator de semnal, întrucât nu are multe componente, si nu necesita o placa de dimensiune redusa, vom trasa liniile de conexiune doar pe 2 straturi, fata-verso, ale placii.

Designul PCB-ului se realizeaza folosind programul EAGLE. Iar designul generatorului de semnal sinusoidal se poate vedea la Anexe, ca si Anexa 2.

2.4 Realizarea software-ului pentru generarea semnalului sinusoidal

Modulul DDS este cel mai important în realizarea semnalului sinusoidal, însa pentru a fi comandat corect trebuie sa programam corespunzator microcontrollerul. Pentru aceeasta vom realiza un program în limbaj de asamblare, si vom folosi pentru simularea sa MPLAB IDE v 7.50, realizat de firma Microchip, cea care realizeaza si PIC16F876.

În mod normal un program pentru un microcontroller trebuie sa cuprinda o subrutina de întreruperi, o subrutina de initializare a registrelor folosite, si binenteles programul principal. Respectând ceea ce am discutat la subcapitolul 2.3.1 "Microcontrollerul" trebuie sa activam intrarile analogice, iesirile digitale de SPI si FSYNC, si binenteles trebuie sa avem grija considerabila la folosirea corecta a timerului.

În primul rând, considerând frecventa de esantionare a convertorului analog-numeric de 1kHz, asta inseamna ca la fiecare 1ms trebuie sa facem operatia de conversie în care folosim registrele: ADCON0, ADCON1, ADRESL, ADRESH pe care trebuie sa le initializam în subrutina de initializare.

Pentru a numara corect 1ms adica 1000us (durata unui ciclu masina), setam prescalerul timerului 0 la 1:4, întrucât timerul 0 este pe 8 biti, deci poate atinge o valoare maxima de 2^8 -1=255; timer0 are 256 valori, si pentru a numara pana la 1000, initializam timerul la valoarea 6, ramânându-i 250 de valori pe care le poate parcurge, deci 250 de us, si setând prescalerul la 1:4, astfel încât timerul sa poata atinge o valoare de (256-6)*4= 1000us. Pentru a detecta momentul când timerul a ajuns la 1ms, folosim un flag,OSF, pe care îl initializam la inceputul programului.

Am decis ca este nevoie sa filtram datele înainte de a le trimite spre SPI, însa timpul de raspuns al filtrului este de 80ms. Pentru aceasta va trebui sa initializam un contor cu valoarea de 80, pe care sa îl decrementam dupa fiecare conversie analog-numerica (care se face la fiecare 1ms), si în clipa în care contorul ajunge la valoarea 0 atunci sa facem o filtrare.

În urma filtrarii obtinem un numar hexazecimal care reprezinta un poiter la o adresa unde se afla datele care trebuie de fapt transmise prin SPI la AD9833. Astfel ca dupa filtrare trebuie sa facem o citire dintr-o tabela aflata la o anumita adresa indi câta de numar ul rezultat prin filtrare. în subcapitolul 2.3.2 "Modulul DDS" am stabilit deja ce fel de date trebuie introduse în tabela, unde anume în memorie si în ce fel, si anume:

- Datele care se trec în memorie se vor calcula respectand ecuatia:
  P= N*107,
  unde N este numar ul returnat de convertorul analog numeric si care reprezinta si pointerul la adresa unde se afla tabela si ia valori cuprinse între 0 si 1000
  P este valoare care trebuie transmisa la acumulatorul de faza din cadrul AD9833
- Datele se vor scrie în memorie incepand cu adresele 1800h si 1C00h, dupa cum urmeaza: cei mai putÎn semnificativi 14 biti (Least Significant Word) la adresa 1800h si cei mai semnificativi 3 biti (Most Significant Word) ai aceluiasi cuvânt pe 17 biti incepand cu adresa 1C00h

Dupa citirea valorii din tabela transmitem cuvintele de comanda catre AD9833, care trebuie sa fie de forma:

0010 0000 0000 0000 cuvânt de control al scrierii
01 Most Significant Word
01 Least Significant Word

2.4.1 Schema logica a programului

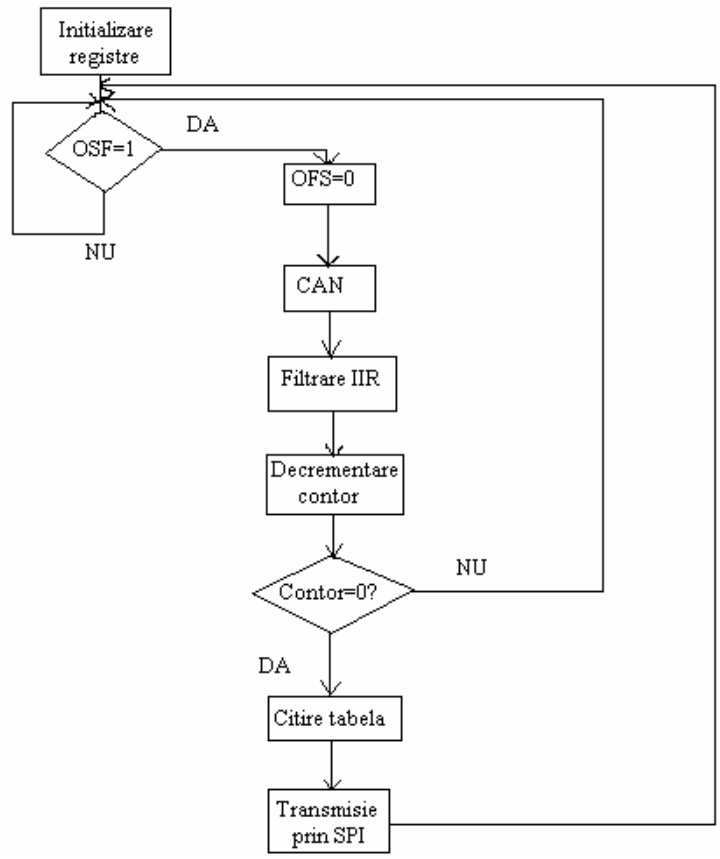


Fig.13 Schema logica a programului pentru microcontroller

Pentru a intelege mai bine functionarea programului vom spune ca la subrutina de initial izare vom initial iza toate registrele folosite pe parcursul programului, dar si variabilele declarate de noi la inceputul programului.

De asemenea vom scrie si în memoria Flash de program tabela de date pentru AD9833 la inceputul programului. Pentru a vedea cu exactitate datele inscrise în memorie (la toate cele 2000 de locatii) gasim la Anexe, Anexa 3 care cuprinde un tabel realizat în programul Excel în care se gasesc datele ce trebuie scrise în memorie atat în zecimal, cat si în hexazecimal, si în format intreg si în format Least Significant Word si Most Significant Word.

La iesirea convertorului analog-numeric(CAN) Obtinem un cuvânt pe 10 biti scris în 2 registre ADRESL si ADRESH. Aceste date vor consti tui intrarea pentru filtrul IIR, adica vor reprezenta Xn din formula:

$$Y_n=(1- 2^{-4})Y_{n-1}+ 2^{-5}(X_n+X_{n-1})$$

Pentru a implementa corect formula vom avea nevoie de niste variabile în care sa memoram valoriile anterioare ale filtrarii, adica X_{n-1} si Y_{n-1}. Cel mai bine ar fi sa vedem programul.

Înainte de a scrie programul însa vom discuta despre registrele folosite în cadrul programului, de configuratia bitilor si de modalitatea în care se programeaza.

Incepem prin definirea bancurilor de lucru. Memoria microcontrollerului este impartita în 4 bancuri, pe care le putem defini ca macrouri la inceputul programului pentru a nu fi nevoiti sa tot setam biti. Alegerea bancului de lucru se face prin setarea a 2 biti din cadrul registrului STATUS : bitii 5 si 6, RP0 si RP1:

Pentru banc0: RP0=0, RP1=0
Pentru banc1 : RP0=1, RP1=0
Pentru banc2 : RP0=0, RP1=1
Pentru banc3 : RP0=1, RP1=1

Continuam cu registrele pe care le folosim la intreruperi:

- INTCON este registrul prin care putem activa intreruperile globale si intreruperile de la timer0, cel pe care noi îl folosim în acest program.
  În acest sens folosim bitii 7, 5, si 2, adica GIE(validarea intreruperilor globale), TMR0IE(validarea intreruperilor de la timer0) si TMR0IF (flag de intrerupere de la timer0, când se produce un overflow).
- OPTION_REG este un registru prin care putem configura timerul0. din cadrul acestui registru vom folosi bitii: 7, adica NOT_RBPU (daca îl punem pe 1 dezactivam intreruperile de la PORTB), bitul 5, adica TOCS (pus pe 0 activeaza clock-ul intern al microcontrollerului care vine la oscilatorul conectat la acesta), bitul 4 adica TOSE (pus pe 0 fa face ca incrementarea clock-ului sa se faca la trenzitia dintre low si high), bitul 3, PSA (pus pe 0 asigneaza prescalerul timerului 0), si deoarece alegem o valoare a prescalerului de 1:4, atunci ultimii 3 biti ii punem pe 001. Asta insemna ca trebuie sa încarcam OPTION_REG cu valoarea:
  1000 0001= 81h

Setarea porturilor A, B si C:

- La PORTA trebuie activati pinii 1 si 2 ca si intrari analogice,dar vom activa toti pinii ca intrari. Pentru aceasta încarcam în registrul TRISA valoarea:
  0011 1111= 3Fh
- La PORTB activam iesirile analogice RB0 si RB1, celelalti pini ii lasam ca si intrari; asta inseamna ca TRISC îl încarcam cu valoarea:
  1111 1100= FCh
- La PORTC trebuie sa activam ieririle SDO si SCLK(bitii 5 si 3 la TRISC), ceea ce insemna ca punem pe 0 acesti biti în cadrul registrului TRISC, ceilalti lasandu-i pe 1(ca si intrari):
  1101 0111= D7h

Pentru configurarea convertorului analog-numeric (CAN), trebuie sa setam urmatoarele registre:

- ADCON0 prin care setam frecvent a de conversie, canalele de la care se face conversia, activam convertorul si îl si putem testa daca avem o conversie în progres sau daca conversia s-a terminat deja. Bitii 7 si 6 ii punem pe 01 pentru a selecta o frecvent a de conversie de Fosc/8; bitii 5-3 determina canalul de unde facem conversia, pe noi ne intereseaza canalul 0 si 1, deci când lucram cu canalul0 setam acesti biti pe 000, iar când lucram cu canalul 1, 001; pentru a porni convertorul, setam bitul ADON(bitul0) pe 1, iar pentru a testa conversia, testam bitul NOT_DONE (bitul 2)
  Pentru o conversie de la canalul 0 vom scrie în registrul ADCON0 valoarea:
  0100 0000=40h, iar pentru a porni conversia 0100 0001=41h

Pentru o conversie de la canalul1 :

0100 1000=48h, iar pentru a porni convertorul 0100 1001= 49h

- ADCON1 cu care putem selecta sa asezam numar ul convertit spre dreapta, adica în ADRESL 8 biti si în ADRESH 2 biti (punând bitul 7 pe 1), apoi configuram poate intrarile de la portul A ca si analogice, astfel ca în acest registru vom încarca valoarea:

  1000 0000= 80h
- Vom mai folosi registrele ADRESL si ADRESH de unde vom citi valorile convertite.
- Pentru a verifica posibilele intreruperi de la convertor testam bitul ADIF al registrului PIR1.

În ceea ce priveste modulul SPI, folosim registrele :

- SSPSTAT: fixam bitul 7, SMP, pe 1astfel încât datele sa se trimita la sfarsitul receptiei lor în modul, bitul 6, CKE îl setam pe 0, astfel încât transmisia sa se intample la tranzitia dinrte starea idle la starea activa, ceea ce insemna ca în registru încarcam valoarea:

  1000 0000=80h
- SSPCON este registrul cu care configuram modulul SPI, si la care trebuie sa setam bitul 5 pe 1, SSPEN, cu care activam modulul, punem bitul 4,CKP, pe 0 astfel încât starea low sa corespunda starii idle, iar ultimii 4 bitii cu care se alege frecvent a de transmisie ii setam pe 0 asa încât frecvent a sa fie Fosc/4. în modul acesta trebuie sa încarcam în SSPCON valoarea

  0010 0000= 20h
- Vom mai folosi registrul SSPBUF, care e un buffer în care si din care sosesc/pleaca datele in/dinspre modulul SPI. prin acest buffer transmitem datele catre AD9833

Pentru citirea din tabela trebuie sa folosim memoria flash de program, vom folosi deci registrele specifice memoriei EEPROM:

- EECON1 ne permitem sa alegem tipul de memorie cu care lucram: de date sau de program prinsetarea bitului EEPGD(îl punem pe 1 pentru memoria de program) si de asemenea putem selecta citirea din memorie daca setam pe 1 bitul RD din acelasi registru.
- În registrele EEADR si EEADRH vom scrie adresele de la care vrem sa citim datele, iar corespunzator acelor adrese, vom gasi datele în registrele EEDATA si EEDATAH

Pentru o mai buna intelegere a acestor registre si a timerului folosit, gasim la Anexe, în Anexa 4 o parte din foile de câta log pentru PIC16F876 si foile de câta log pentru AD9833 la Anexa 5 .

2.4.2 Programul pentru generarea semnalului sinusoidal realizat pentru PIC16F876

```
list      P=pic16F876


#include <p16f876.inc>

;************definim parametrii pentru CANAL0 la urmatoarele locatii RAM****************

ADLS0       equ     0x0020          ; Definim variabile pt achizitia ADC(Xn)
ADHS0       equ     0x0021          ; Valoarea pentru CAN  MSByte
ADLS0O      equ      0x0022         ; Valoarea anterioara de la achizitia CAN (X(n-1))
ADHS0O      equ     0x0024          ; Valoarea anterioara MSByte
FL0A        equ     0x0025          ; Valoarea actuala a filtrarii LSByte (Yn)
FH0A        equ     0x0026          ; Valoare actuala MSByte
FL0O        equ      0x0027         ; Valoare anterioara LSByte, (Yn-1)
FH0O        equ     0x0028          ; Valoare anterioara MSByte
FREQL0L     equ     0x0029          ; Least Significant Word, cei 8 bitii Low(14 bits)
FREQL0H     equ     0x002A          ; Least Significant Word, cei 8 biti High (6 biti de fapt)
FREQH0L     equ     0x002B          ; Most significant Word, cei 8 biti Low (14 bits)
FREQH0H     equ     0x002C          ; Most significant Word, cei 8 biti High


; ************definim parametrii pentru CANAL1 la urmatoarele locatii RAM*****************


ADLS1       equ     0x0030          ; Definim variabile pt achizitia CAN(Xn)
ADHS1       equ     0x0031          ; Valoarea pentru CAN  MSByte
ADLS1O      equ     0x0032          ; Valoarea anterioara de la achizitia CAN (X(n-1))
ADHS1O      equ     0x0034          ; Valoarea anterioara MSByte
FL1A        equ     0x0035          ; Valoarea actuala a filtrarii LSByte (Yn)
FH1A        equ     0x0036          ; Valoare actuala MSByt
FL1O        equ      0x0037         ; Valoare anterioara LSByte, (Yn-1)
FH1O        equ     0x0038          ; Valoare anterioara MSByte
FREQL1L     equ     0x0039          ; Least Significant Word, cei 8 bitii Low(14 biti)
FREQL1H     equ     0x003A          ; Least Significant Word, cei 8 biti High (6 biti de fapt)
FREQH1L     equ     0x003B          ; Most significant Word, cei 8 biti Low (14 bits
FREQH1H     equ     0x003C          ; Most significant Word, cei 8 biti High


;************************* definim diverse contoare*******************************


CONTOR      equ     0x0040          ; definim un contor pt transmisia spre SPI
XnL         equ     0x0041          ; variabile unde memoram valoriile actuale si anterioare
XnH         equ     0x0042          ; pentru filtrarea IIR
SUMXL       equ     0x0043
```

```
SUMXH       equ     0x0044
YantL       equ     0x0045
YantH       equ     0x0046
SUMYL       equ     0x0047
SUMYH       equ     0x0048
YnL         equ     0x0049
YnH         equ     0x004A
OSF         equ     0x004B          ; Definim un Flag pentru timer0 care sa ne indice 1ms


;************************ Macrouri de comutare bancuri*****************************

bank0  macro                                    ;memoria PIC-ului este impartita în 4 bancuri,
            bcf         STATUS,     RP0         ; în cadrul fiecarui banc putandu-se face diverse
            bcf         STATUS,     RP1         ; operatii asupra deverselor registre
       endm                                     ;Pentru a folosi un anumit registru trebuie ca în
                                                ;cadrul memoriei sa ne aflam în bancul unde se
bank1  macro                                    ; afla si acel registru, altfel operatiile nu se
            bsf         STATUS,     RP0         ;efectueaza asupra registrului
            bcf         STATUS,     RP1
       endm

bank2  macro
            bcf         STATUS,     RP0
            bsf         STATUS,     RP1
       endm

bank3  macro
            bsf         STATUS,     RP0
            bsf         STATUS,     RP1
       endm



;****************definim tabela de valori din  cadrul memoriei flash**********************

            org     0x1800          ; definire adresa de start tabela de faza
frecv0l     dw      0x006B          ; la adresa 1800h avem scris 6B
            dw      0x00D6          ; la adresa 1801h avem scris D6h, etc.
; ..............................................................................................................
; tabela cuprinde 1000 de valori corespunzatoare Least Significant Word (14 biti)

            org     0x1C00          ; definire adresa de start tabela de faza
frecv0h     dw      0x0000          ; la adresa 1C00h avem scris 00h
            dw      0x0000
;..............................................................................................................
;tabela are 1000 de valori corespunzatoare Most Significant Word
```

```
;************************programul propriu-zis************************************

                org 0x0000
                nop
                nop
                nop
                goto    start

; **********************rutina de tratare a intreruperii*********************************


intrp           bcf             INTCON,   GIE       ; invalidare intreruperi globale
                btfss           INTCON,   T0IF      ; test intrerupere TIMER0
                goto            endint

                movlw           d'6'                ; încarc valoarea 6 în zecimal în TIMER0
                movwf           TMR0                ; reinitializare TIMER0
                bcf             INTCON, T0IF        ; stergere flag intrerupere TIMER0
                bsf             OSF, 0              ; setare flag 1ms
endint          retfie


;**********************sfarsit rutina de tratare a intreruperii*********************************


start           call    pginit
                call     main


; ************************* rutina de initial IZARE controller **************************

pginit          bank1
                movlw           0x0081              ; prescaler=1:4
                movwf           OPTION_REG
                movlw           0x80                ; setez pe "right justi fied" registrii ADRESH si
                movwf           ADCON1              ; ADRESL la achizitia CAN-ului
                movlw           0x3F                ; secventa pt initial izarea portului PORTA
                movwf           TRISA               ; activez ca si analog input pinii RA0 si RA1
                movlw           0xFC                ; secventa pt initial izarea PORTB
                movwf           TRISB               ; activez ca digital output pinii RB0 si RB1, ce
                                                    ; urmeaza a fi folositi ca FSYNC1,2
                 movlw          0xD7                ; secventa de initial izare PORTC
```

```
 movwf      TRISC           ; activez ca digital output pinii SDO si SCK
 movlw      0x80            ; secventa de initial izare a modulului SPI
movwf       SSPSTAT
bank0
movlw       0x0006          ;încarc în TMR0 valoarea 6, astfel încât
movwf       TMR0            ; sa numere pana la 256-6=250, iar cu un
                    :       ; prescaler de 1:4 sa numar am pana la 1000us
                            ; adica perioada la care se face conversia A/N
movlw       0x20            ; activez portul serial (SPI)
movwf       SSPCON          ; declar idle state= low si transmisia se
                            ; realizeaza la trecerea de la idle la activ
movlw       0x40            ; initial izez modulul A/D, pt care aleg frecventa
                            ; Fosc/8, citire de la intrarea analogica  AN0
movwf       ADCON0          ; modulul e oprit( pinul ADON=0)
bank1
movlw       0x80            ; configurez intrari/iesiri analogice si digitale, si
                            ; selectez frecvent a de lucru
movwf       ADCON1          ; a modulului de conversie analog-digitala
bank0                       ; selectez bank0 pt a initial iza variabilele
                            ; definite mai sus pt canalele analogice 0 si 1
 movlw      0x0000          ; inscriu în toate variabilele valoarea 0h
 movwf      ADLS0
 movwf      ADHS0
 movwf      ADLS0O
 movwf      ADHS0O
 movwf      FL0A
 movwf      FH0A
 movwf      FL0O
 movwf      FH0O
 movwf      FREQL0L
 movwf      FREQL0H
 movwf      FREQH0L
 movwf      FREQH0H
 movwf      ADLS1
 movwf      ADHS1
 movwf      ADLS1O
 movwf      ADHS1O
 movwf      FL1A
 movwf      FH1A
 movwf      FL1O
 movwf      FH1O
 movwf      FREQL1L
 movwf      FREQL1H
 movwf      FREQH1L
 movwf      FREQH1H
 movwf      XnL
 movwf      XnH
 movwf      SUMXL
 movwf      SUMXH
```

```
                movwf       YantL
                movwf       YantH
                movwf       SUMYL
                movwf       SUMYH
                movwf       YnL
                movwf       YnH
                movlw       0x0050          ; initializez contorul cu valoarea de 80 in
                                            ; zecimal, si o sa îl  folosesc la filtrare si
                                            ; trimitere spre SPI

                movwf       CONTOR          ; 80 se refera la raportul dintre timpul de
                                            ; raspuns  al filtrului si perioada de
                                            ; esantionare   a ADC-ului
                clrf        OSF             ; sterg bitul 0 din  OSF, nu am ajuns la 1ms
                clrf        INTCON          ; invalidare intreruperi
                bsf         INTCON,  T0IE   ; validare intrerupere TIMER0
                movlw       d'6'
                movwf       TMR0            ; initial izare TIMER0
                bsf         INTCON,   GIE   ; validare intreruperi


;********************* procedura de conversie pentru modulul ADC **********************


conv            bank0
                movlw       0x0040
                movwf       ADCON0          ; setam canalul de la care se face citirea, timpul
                                            ; de conversie, si ADON=0
                bcf         PIR1, ADIF      ; flag de intreruperi
                bsf         ADCON0,ADON     ; ADON=1
                bsf         ADCON0, NOT_DONE
loopconv0       btfss       PIR1, ADIF      ; când ADIF=1 nu executa urmatoarea
                goto         loopconv0      ; instructiune, când ADIF=0 executa bucla

                bank1
                movfw       ADRESL
                bank0
                movwf       ADLS0           ; transfer ceea ce am citit în variabilele ce le-am
                movfw       ADRESH          ; definit
                movwf       ADHS0
                bcf         ADCON0, ADON
; canal 1                                   ; repetam procedura de mai sus pentru canalul1
                movlw       0x0048
                movwf       ADCON0
                bcf         PIR1, ADIF
                bsf         ADCON0, ADON
                bsf         ADCON0, NOT_DONE
loopconv1       btfss       PIR1, ADIF
```

```
            goto        loopconv1
            bank1
            movfw       ADRESL
            bank0
            movwf       ADLS1
            movfw       ADRESH
            movwf       ADHS1
            bcf         ADCON0, ADON
            return


;**********************procedura de filtrare ***********************************


IIR         bank0                       ; implementam formula
            bcf         STATUS,C        ; Yn=(1- 2-4)Yn-1+ 2-5(Xn+Xn-1)
            movf        XnL, W          ;În  SUMX memoram valoarea anterioana
            addwf       SUMXL, F        ; realizam suma: Xn+Xn-1 , cu transport
            movf        SUMXH, W        ; de la octetul low la octetul High
            btfsc       STATUS, C       ; daca avem transport adaugam 1 la high
            addlw       1               ; daca nu avem, sarim peste instructiune
            movwf       SUMXH
            movf        XnH, W
            addwf       SUMXH, F
            bcf         STATUS, C       ; stergem bitul de carry(transport)
            rrf         SUMXH, F        ; trebuie sa siftam suma celor 2 valori
            nop                         ; achizitionate cu 5 pozitii la dreapta
            rrf         SUMXL, F        ;acest lucru se realizeaza siftand atat octetul
            nop                         ;high cat si low de 5 ori, dupa fiecare siftare
            bcf         STATUS, C       ;bitul de carry se sterge
            rrf         SUMXH, F
            nop
            rrf         SUMXL, F
            nop
            bcf         STATUS, C
            rrf         SUMXH, F
            nop
            rrf         SUMXL, F
            nop
            bcf         STATUS, C
            rrf         SUMXH, F
            nop
            rrf         SUMXL, F
            nop
            bcf         STATUS, C
            rrf         SUMXH, F
            nop
            rrf         SUMXL, F
            nop                         ; în acest moment am obtinut
```

```
bcf     STATUS, C       ; 2^-5(Xn+Xn-1)
movf    YantL, W
movwf   SUMYL           ; în SUMY efectuez calculele de genul
movf    YantH,W         ; (1- 2^-4)Yn-1 pentru inceput încarc în SUMY
movwf   SUMYH           ; Yant
bcf     STATUS, C
rrf     YantH, F        ; pentru inceput determinam 2^-4Yn-1
nop                     ; prin 4 siftari la dreapta a YantL si YantH
rrf     YantL, F
nop
bcf     STATUS, C
rrf     YantH, F
nop
rrf     YantL, F
nop
bcf     STATUS, C
rrf     YantH, F
nop
rrf     YantL, F
nop
bcf     STATUS, C
rrf     YantH, F
nop
rrf     YantL, F
nop
bcf     STATUS, C
comf    YantL, W        ; determinam -2^-4Yn-1  prin metoda
addlw   1               ; Cod complementul lui 2
movwf   YantL           ; adica complementez numar ul si ii adaug 1
comf    YantH, W
btfsc   STATUS, C       ; la YantH ii adaug 1 doar în caz de transport
addlw   1
movwf   YantH
bcf     STATUS, C
movf    YantL, W
addwf   SUMYL, F        ; facem  operatia:
movf    YantH, W        ; (1- 2^-4)Yn-1
btfsc   STATUS, C
addlw   1
nop
addwf   SUMYH, F        ; în cele din  urma aduman SUMY cu SUMX
movf    SUMXL, W        ; adica (1- 2^-4)Yn-1+ 2^-5(Xn+Xn-1)
addwf   SUMYL, F
movf    SUMXH, W
btfsc   STATUS, C
addlw   1
addwf    SUMYH, F
movf    SUMYL, W
movwf   YnL             ; valoarea calculata o scriem în Yn
```

```
                movf        SUMYH, W        ; adica e valoarea actuala filtrata
                movwf       YnH
                return

;************************ Procedura de citire din  tabela *********************************


read_table      bank0                       ; citim tabela pentru canalul0
                movf        FL0A, W         ;intrarea pentru tabela este valoarea actuala
                addlw       0x00            ;a Yn filtrat si acest numar  reprezinta un
                bank2                       ;pointer la adresa unde se afla tabela
                movwf       EEADR           ; datele pe care le caut eu se gasesc la
                bank0                       ;la adresa 1800+Yn
                movf        FH0A, W         ; la EEADR se gaseste partea low a adresei de
                btfsc       STATUS, C       ; mai sus
                addlw       1
                addlw       0x18
                bank2
                movwf       EEADRH          ;
                bank3
                bsf         EECON1, EEPGD   ; acceseaza memoria de date
                bsf         EECON1, RD      ; activez citirea
                nop
                nop
                bank2
                movf        EEDATA, W        ; citesc datele de la adresa 1800h
                bank0
                movwf       FREQL0L         ;e vorba de Least Significant Word
                bank2                       ; octetul cel mai putÎn  semnificativ
                movf        EEDATH, W       ; si octetul cel mai semnificativ
                bank0
                movwf       FREQL0H
                movf        FL0A, W         ; aceeasi procedura pentru determinarea
                addlw       0x00            ; Most Significant Word care se citeste
                bank2                       ; de la adresa 1C00h
                movwf       EEADR
                bank0
                movf        FH0A, W
                btfsc       STATUS, C
                addlw       1
                addlw       0x1C
                bank2
                movwf       EEADRH
                bank3
                bsf         EECON1, EEPGD   ; acceseaza memoria de program
                bsf         EECON1, RD      ; activez citirea
                nop
                nop
                bank2
```

```
                movf        EEDATA, W
                bank0
                movwf       FREQH0L
                bank2
                movf        EEDATH, W
                bank0
                movwf       FREQH0H
;canal 1
                bank0                       ; repetam procedura de mai sus pentru
                movf        FL1A,W          ; canalul 1
                bank2
                movwf       EEADR
                bank0
                movf        FH1A, W
                addlw       0x0018
                bank2
                movwf       EEADRH
                bank3
                bsf         EECON1, EEPGD   ; acceseaza memoria de program
                bsf         EECON1, RD      ; activez citirea
                nop
                nop
                bank2
                movf        EEDATA, W
                bank0
                movwf        FREQL1L
                bank2
                movf        EEDATH, W
                bank0
                movwf       FREQL1H
                movf        FL1A, W
                bank2
                movwf       EEADR
                bank0
                movf        FH1A, W
                addlw       0x1C
                bank2
                movwf       EEADRH
                bank3
                bsf         EECON1, EEPGD   ; acceseaza memoria de program
                bsf         EECON1, RD      ; activez citirea
                nop
                nop
                bank2
                movf        EEDATA, W
                bank0
                movwf       FREQH1L
                bank2
                movf        EEDATH, W
```

```
            bank0
            movwf       FREQH1H
            bcf         EECON1, EEPGD   ; accesez memoria de date
            return


;******************procedura de transmitere prin SPI*****************************


SPI         bank0
            bcf         PORTB, 0        ; setam iesirea FSYNC1 pe 0
            nop
; ****************canal0*********************
            movlw       0x20            ; MSB control bits
            movwf       SSPBUF
            movlw       0x00            ; LSB
            movwf       SSPBUF
            movf        FREQH0H, W      ; Most Significant word  High
            andlw       0x7F            ; facem un "SI" logic cu 7F pentru a obtine
            movwf       SSPBUF          ; la partea high 01 urmat de MSW high
            movf        FREQH0L, W      ; Most Significant word  low
            movwf       SSPBUF
            movf        FREQL0H, W      ; Least significant word high
            andlw       0x7F            ;Si logic cu 7F pentru a transmite un cuvânt
            movwf       SSPBUF          ;de forma 01 LSW high
            movf        FREQL0L, W      ; Least significant word low
            movwf       SSPBUF
            nop
            bsf         PORTB, 0        ; pun FSYNC! pe 1 ( incepe procesarea)
            nop
            bcf         PORTB, 1        ; pun FSYNC2 pe 0, transmit de la
            nop                         ;canalul 2 spre AD9833
; *****************canal1**********************
            movf        SSPBUF, W       ;repetam procedura pentru canalul 1
            movlw       0x20            ; LSB biti de control
            movwf       SSPBUF
            movlw       0x00            ; MSB biti de control
            movwf       SSPBUF
            movf        FREQH1H, W      ; Most significant word
            andlw       0x7F            ; ca mai sus facem un si logic
            movwf       SSPBUF          ;cu 7F pentru a transmite forma corecta
            movf        FREQH1L, W
            movwf       SSPBUF
            movf        FREQL1H, W      ;Least Significant Word
            andlw       0x7F            ; si logic cu 7F
            movwf       SSPBUF
            movf        FREQL1L, W
            movwf       SSPBUF
            nop
```

```
                bsf             PORTB, 1            ; trecem FSYNC2 pe 1 pentru a incepe
                nop                                 ; procesarea datelor
                return


;**********************Programul principal*********************************


main
                bank0                       ; verificam daca OSF(bitul0)=1, adica
                btfss           OSF,0               ; daca am ajuns la 1ms, daca nu, atunci fac
                goto            main                ;bucla pana ajung la 1ms
                call            conv                ; apelez subrutina de conversie
; ******************pentru canal 0*******************
                movf            ADLS0, W
                movwf           XnL                 ; intrarea pentru filtrul va fi iesirea CAN
                movf            ADHS0, W
                movwf           XnH
                movf            ADLS0O, W           ; ADLS0O contine valoarea anteriara a CAN
                movwf           SUMXL
                movf            ADHS0O, W           ; în SUMX efectuam calcule
                movwf           SUMXH
                movf            FL0O, W             ; FL0O contine valoarea Y(n-1) pentru filtrul,
                movwf           YantL               ; ceea ce corespunde la valoarea filtrata anterior
                movf            FH0O, W
                movwf           YantH
                call            IIR                 ; avnd datele de intrare, apelam subrutina de
                movf            YnL, W              ; filtrare
                movwf           FL0A                ; FL0A contina valoarea actuala filtrata
                movf            YnH, W
                movwf           FH0A
                movf            YnL, W              ;  trecem ce e actual la iesirea subrutinei IIR in
                movwf           FL0O                ; anterior
                movf            YnH, W
                movwf           FH0O
                movf            XnL, W              ; si valoarea actuala convertita
                movwf           ADLS0O              ; o trecem în valoare anterioara
                movf            XnH, W
                movwf           ADHS0O
; ******************Pentru canal 1*******************
                movf            ADLS1, W            ; facem acelasi lucru pentru canalul1
                movwf           XnL
                movf            ADHS1, W
                movwf           XnH
                movf            ADLS1O, W           ;ADLS1O reprezinta valoarea anterioara
                movwf           SUMXL
                movf            ADHS1O, W
                movwf           SUMXH
                movf            FL1O, W             ; Vechea valoare filtrata
```

```
movwf     YantL
movf      FH1O, W
movwf     YantH
call      IIR             ; apelam si pentru canalul 1 subrutina
movf      YnL, W          ; de filtrare
movwf     FL1A            ; trecem ce e actual la iesirea subrutinei IIR în
movf      YnH, W          ; valori anterioare
movwf     FH1A
movf      YnL, W
movwf     FL1O
movf      YnH, W
movwf     FH1O
movf      XnL, W
movwf     ADLS1O
movf      XnH, W
movwf     ADHS1
decfsz    CONTOR,F        ; dupa ce am facut conversia si filtrarea
goto      main            ; decrementam CONTOR pana ajunge de la
call      read_table      ;80 la 0 si apoi apelam citirea din  tabela
call      SPI             ; dupa citire facem transmisia prin SPI
movlw     0x50            ; dupa transmisie repunem CONTOR pe 80
movwf     CONTOR          ; si ne intorcem în maÎn  pentru a relua toata
goto      main            ;procedura


end
```

Capitolul 3
Generarea semnalului dreptunghiular si PWM

3.1 Cerinte

Sa se genereze doua semnale dreptunghiulare, unul doar cu frecvent a variabila si un altul cu frecvent a si factor de umplere variabil, care sa se încadreze între limitele:

- Semnal dreptunghiular

Nr.	Descriere	Simbol	Min.	Nom.	Max.	Unitate
1	Amplitudine	V_{AMPL_TW}	8	-	34	V
2	Frecvent a	f_{SINE}	0	-	2	KHz

- Semnal PWM (cu frecvent a si factor de umplere variabile)

Nr.	Descriere	Simbol	Min.	Nom.	Max.	Unitate
1	Amplitudine	V_{AMPL_TOSS}	0	-	10	V
2	Frecvent a	f_{SINE}	0	-	2	KHz
3	Durata impulsului	t_{HIGH}	0.4	-	5	ms

Cele doua semnale se vor genera pe acelasi circuit fizic, iar pentru implementare, ca si pentru generarea semnalului sinusoidal, se pot folosi circuite integrate, sau circuite dedicâte generarii de semnale dreptunghiulare.

Se doreste ca utilizatorul sa aiba o interfata care sa-i permita sa modifice frecvent a, si factorul de umplere. Pentru aceeasta se pot folosi potentiometre la care utilizatorul sa aibe acces.

3.2 Concept

La generarea semnalului sinusoidal ne-am familiarizat deja cu microcontollerul de la Microchip, PIC16F876, asa ca pentru generarea celor doua semnale dreptunghiulare vom folosi acelasi microcontroller.

Cum semnalul dreptunghiular e foarte apropiat de un semnal digital, întrucât insemna transmiterea unui sir de "1 logic" (pe durata high) si a unui sir de "0 logic" (pe durata low), putem realiza cele doua semnale doar prin folosirea microcontrollerului, fara a mai fi nevoie sa folosim un alt circuit dedicat. La generarea semnalelor de catre PIC trebuie însa avut grija la timere si la capacitatea lor de a numar a.

Vom avea nevoie de 3 potentiometre: unul pentru ajustarea frecvent ei semnalului dreptunghiular, altul pentru ajustarea frecvent ei semnalului PWM, si cel de-al treilea pentru ajustarea factorului de umplere al semnalului PWM.

Blocurile functionale ale generatorului de semnal dreptunghiular si PWM de care avem nevoie sunt:

- Blocul de alimentare, care presupune si stabilizatorul de tensiune (alimentam tot cu 5V microcontrollerul)
- Microcontrollerul

Schema bloc a unui astfel de generator este:

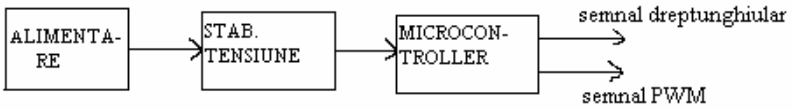


Fig.14 Schema bloc a generatorului

Pentru alimentare vom folosi aceeasi structura ca la generatorul de semnal sinusiodal.

În ceea ce priveste microcontrollerul, alegem în continuare PIC16F876, întrucât are 3 porturi de care ne putem folosi, convertor analog-numeric, si 3 timere cu care putem decide perioada semnalelor pe care vrem sa le generam .

Spre deosebire de generarea semnalului sinusoidal, la generarea semnalului dreptunghiular vom citi din tabele valorile cu care trebuie sa încarcam timerele pentru a genera la iesire semnale cu perioada impusa de potentiometre.

În cadrul acestui generator de semnal, partea cea mai importanta o consti tuie software-ul, întrucât schema în sine este foarte simpla, generarea semnalului dreptunghiular si PWM realizandu-se doar din software.

3.3 Functionare electronica

Vom explica functionarea generatorului de semnale dreptunghiulare pe blocuri, referindu-ne la cele 2 blocuri mai sus amintite:

- Blocul de alimentare, care presupune si stabilizatorul de tensiune (alimentam tot cu 5V microcontrollerul)
- Microcontrollerul

3.3.1 Blocul de alimentare

PIC16F876 necesita o tensiune de alimentarea între 2 si 5,5V. Noi alegem sa-l alimentam cu tensiunea standard de 5V, iar pentru a obtine aceasta tensiune trebuie sa folosim un stabilizator de tensiune, pe care noi îl alegem LM7805.

producatorul recomanda o structura cu stabilizatorul încadrat de doua condensatoare de valoare 100nF pentru protectie . De asemenea mai folosim si o dioda de la baterie pana la stabilizator, care sa ofere protectie stabilizatorului la variatii mari de curent. La iesirea blocului de alimentare punem un condensator electrolitic tantalum (metal tranzisti v foarte usor) pentru a stoca sarcini si a modera tensiunea la iesirea si pentru a se opune la fluctuatii mari de curent si tensiune, si pentru a asigura o tensiune stabilizata. Alegem pentru acesta o valoare standard de 10uF.

Schema blocului de alimentare este:

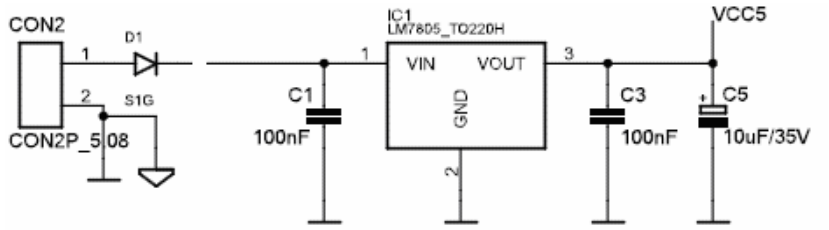


Fig.15 Schema electronica a blocului de alimentare si stabilizare

3.3.2.Microcontrollerul

Am ales pentru realizarea generatorului de semnale dreptunghiulare PIC16F876, fiind deja familiarizati cu el de la realizarea generatorului de semnal sinusoidal.. Un astfel de microcontroller are ca si avantaj faptul ca are incorporat un convertor analog-numeric care sa transforme în cuvinte binare semnalele receptionate de la intrare si faptul ca avem 3 timere pe care le putem utiliza la generarea celor doua semnale..

Vom conecta la 3 pini care pot fi folositi ca si intrari analogice 3 potentiometre pe care le vom folosi ca sa variem frecvent ele celor doua semnale dreptunghiulare, iar cel de-al treilea potentiometru îl vom folosi pentru modificarea factorului de umplere al semnalului care se vrea PWM.

Alimentarea microcontrollerului

Tensiunea de alimentare pentru un astfel de microcontroller trebuie sa fie cuprinsa între 2V si 5,5V. Noi am ales deja o tensiune standard de 5 V cu care sa alimentam microcontrollerul.

Tot în cadrul alimentarii vorbim si despre pinii de V_{DD} (pinul 20)si V_{SS} (pinii 8 si 19) care se pot folosi si ca tensiuni de referint în cadrul operatiilor pe care le face integratul, mai precis conversia. V_{DD} se leaga la alimentarea de 5V, conectand la acest pin si un condensator pentru protectia PIC-ului la variatii de tensiune, iar V_{SS} se va conecta la masa, fiind referint pentru 0V

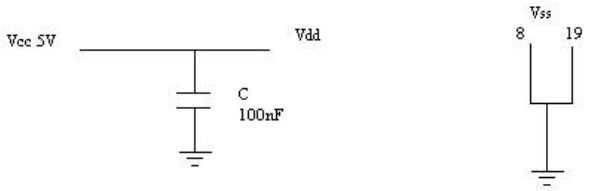


Fig.2 Mod de conectare a V_{DD} si V_{SS}

Pentru conectarea VDD alegem un condensator de valoare 100nF, recomandat de producator.

De asemenea mai avem un pin MCLK/Vpp pe care îl folosim ca si intrare de tensiune de programare si care va fi conectat la tensiunea de alimentare V_{DD} de 5V, însa care necesita un circuit de protectie , realizat din doua rezistente si un condensator, asa cum este el recomandat de producator. Pentru rezistente sunt recomandate valorile:

R1< 40k? , noi alegem R1=10k?

R2 = 1k? , noi alegem chiar valoarea de 1k?

Iar pentru C alegem o valoare standard de 100nF, asa cum recomanda producatorul la celelalte condensatoare pentru protectie .

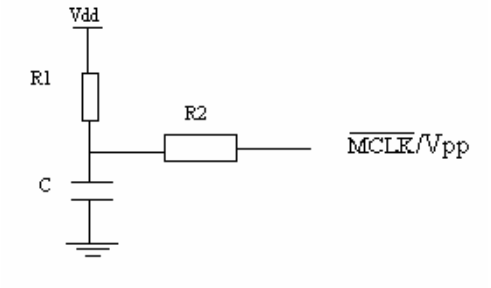


Fig.3 Mod de conectare a pinului MCLK/Vpp la alimentare

Bloc oscilator pentru PIC16F876

Pentru ca cele 3 timere ale microcontrolerului sa functioneze trebuie sa conectam PIC-ul la un timer extern sau la un oscilator cu cristal. Citind foile de câta log ale PIC16F876 referitoare la frecvent a cu care merg timerele raportat la oscilatorul (adica frecvent a interna a microcontrollerului), care e Fosc/4, respectiv durata unui ciclu masina 4/Fosc, alegem un oscilator cu o frecvent a de oscilatie de 4MHz, care va determina o durata a ciclului masina de 1us.

În acest sens alegem un rezonator de 4MHz , QMIM004, pe care îl vom conecta asa cum ne recomanda producatorii:

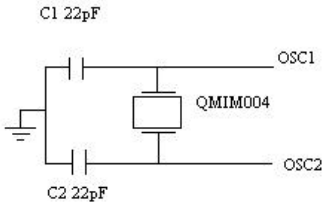


Fig.4 Conectarea oscilatorului la PIC

OSC1 si OSC2 sunt pinii 9 si 10 ai microcontrollerului la care conectam oscilatorul.

Porturile folosite ale microcontrollerului

Dupa cum deja sti m, microcontrollerul are 3 porturi, si le vom folosi pe toate.
Intrarile analogice le vom seta la PORTA, fiind vorba despre 3 intrari(doua pentru ajustarea frecvent ei si una pentru ajustarea factorului de umplere) setam RA0, RA1, RA2 ca si intrari. Asta inseamna setarea pe 1 a bitilor 0,1 si 2 corespunzatori registrului TRISA.
În cazul portului B setam una din iesiri, fie ea dreptunghiulara sau PWM, la pinul RB1, ceea ce insemna scrierea bitului 1 al regitrului TRISB pe 0. Pentru semnalul care ne mai ramane, PWM sau dreptunghiular vom alege una din iesirile PORTC, deoarece acest port are si un modul de PWM, însa pentru al putea folosi trebuie sa facem o verificare a capacitatii timerului asociat acestui modul, adica timer2. daca vrem sa activam iesirea de PWM a acestul modul din cadrul portului C trebuie sa punem pe 0 bitul 1 al registrului TRISC.

Convertorul analog-numeric

PIC16F876 beneficiaza de un convertor analog-numeric (CAN) cu 5 intrari analogice, a carui valoare numerica corespondenta semnalului analogic de la intrare este un numar binar pe 10 biti, ce se scrie în 2 registre ADRESL si ADRESH, 8 biti într-un registru si 2 în altul, selectabil din software.

CAN face conversia având ca referint 2 tensiune care pot V_{DD} (high) V_{SS}(low), sau tensiuni citite de la pinii RA3 si RA4, în orice fel de combinatii. Pentru circuitul de fata alegem ca referint tensiuniile de 5V high si 0V low, adica V_{DD} si V_{SS} .

Pentru selectia canalelor 0, 1 si 2 analogice trebuie sa programam registrele ce control ai CAN: ADCON0 si ADCON1, asa cum se va vedea în cadrul programului aplicat PIC-ului.

Formula dupa care se face conversia este:

$$U_{IN}= [(U_{REF+} +U_{REF-})/1000]*N- U_{REF-}$$

unde $U_{REF+} =V_{DD}=5V$

$U_{REF-} =Vss =0V$

N= numar ul returnat de CAN însa în baza zece

Acest numar va fi un pointer la o adresa din tabelele de unde vom citi valorile care trebuie încarcâte în timere astfel ca ele sa determine perioada si factorul de umplere pe care vrem sa le generam .

Timerele folosite

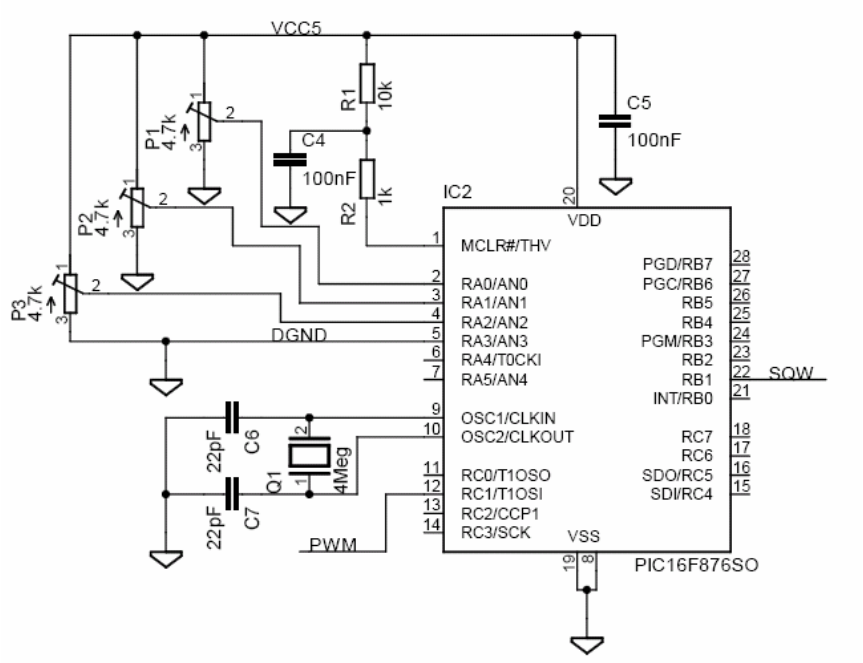
PIC16F876 are 3 timere, care lucreaza la o frecvent a de Fosc/4, adica 1MHz, deci oricare dintre aceste timere determina durata unui ciclu masina de 1us.
Diferenta dintre timere este aceea ca timer0 si timer2 sunt pe 8 biti, iar timer1 este pe 16 biti, iar timer2 se foloseste în aplicatii care implica modulul PWM.

Dat fiind datele mai sus mentionate, alegem pentru a genera cele doua semnale timerele1 si 2.

Despre modul de realizare a tabelelor si de alegere a iesirilor vom discuta în cadrul subcapitolului dedicat realizarii software-ului.

Schema blocului de microcontroller

Luand în considerare recomandarile producatorului si tinând cont de intrarile si iesirile care ne intereseaza în cadrul acestui generator, schema electronica a acestui bloc este:



3.3.3 Schema electronica a generatorului de semnale dreptunghiular si PWM

Schema electronica a generatorului de semnal dreptunghiular si de semnal PWM va cuprinde cele doua blocuri prezentate interior si poate fi vizualizata la rubrica de Anexe, ca si Anexa 6.

3.3.4 PCB-ul generatorului de semnal dreptunghiular si PWM

Ca si PCB-ul generatorului de semnal sinusoidal, si acest PCB va fi realizat utilizandu-se cele doua metode: SMT si THT, fiind vorba de putine componente necesare reslizarii sale. Designul acestui PCB a fost realizat folosÎn pogramul EAGLE, si schema finala poate fi studiata la Anexe, fiind vorba despre Anexa 7.

3.4 Realizarea software-ului pentru generarea de semnale dreptunghiulare si PWM

Pentru inceput trebuie sa ne decidem asupra principiului de functionare a acestui generator de functii si sa hotaram ce timer folosim pentru generarea semnalului dreptunghiular si ce timer pentru semnalul PWM. Pornim de la ideea ca citim de la potentiometre cum vrem sa variem frecvent ele si factorul de umplere. De fapt citim o valoare de tensiune care poate sa varieze între 0 si 5V, iar convertorul analog numeric al PIC-ului va asocia la aceste valori de tensiune pana la 1000 de valori digitale, care exprimate în binar sunt cuvinte pe 10 biti.

Aceste cuvinte vor fi pointeri la adresele unde vom stoca în memoria de program tabela de datele care determina frecvent a si factorul de umplere al semnalelor.

Determinarea datelor care trebuie trecute în tabele o facem în cele ce urmeaza:

Semnalul dreptunghiular

Trebuie sa generam un semnal dreptunghiular a carui frecventa sa o putem modifica între valorile 10Hz-2kHz. întrucât convertorul analog-numeric poate diferentia 1000 de valori, insemna ca din 2 în 2 Hz avem o alta valoare numerica.

Ambele timere au un tact cu perioada de 4/Fosc= 4/4MHz= 1us, deci un ciclu masina are o durata de 1us.

La o frecvent a de 10Hz, perioada semnalului si durata sa high sunt:

T= 1/10Hz= 100ms=100 000us (cat e un ciclu masina)

t_{high} = T/2= 50 000us (impartim la 2 deoarece factorul de umplere e 1/2)

Asta insemna ca numar ul inscris în tabela pentru o frecvent a de 10Hz este 50 000, adica un numar pe 16 biti, ceea ce în mod normal doar timer 2 ne poate oferi. însa înainte de a lua o decizie mai facem niste calcule pentru semnalul dreptunghiular, dar si pentru semnalul PWM.

Pentru acelasi semnal la frecvent a maxima de 2kHz durata semnalului si perioada High sunt:

T=1/2kHz= 500us

t_{high} = T/2 = 250us

Numar ul pe care trebuie sa-l inscriem în acest caz este 250, adica un numar pe 8 biti.

Semnal PWM

În cazul semnalului PWM trebuie sa variem frecvent a între valorile: 10Hz-2kHz , iar durata impulsului între valorile: 0,4ms – 5ms.

Pentru semnalul PWM avem o formula de calcul a frecvent ei raportata la viteza de deplasare a autovehiculului si la durata impulsului. asa cum am specificat la inceputul acestei lucrari, semnalul PWM se realizeaza pentru a simula semnalul de la iesirea unui tahograf, care este în realitate un senzor de viteza.

Formula cu care determinam viteza de deplasare este:

$v = 3600/ (T*16/ t_{high}) = 255* t_{high} / T$

unde v= viteza de deplasare care poate lua valori între 0-120km/h

T= perioada semnalului PWM

t_{high} = durata impulsului

DÎn formula precedenta deducem ca perioada semnalului PWM se calculeaza în raport cu viteza si durata impulsului:

$T= 255* t_{high} /v$

Avem nevoie de 2 tabele, una pentru viteza si una pentru durata impulsului

Tabela de viteza ar trebui sa cuprinda 120 de valori diferite, câte una la fiecare km/h, iar tabela cu valori pentru durata impulsului ar trebui sa contina 1000 de valori, atat cat ne genereaza si CAN-ul, adica câte o valoare la fiecare:

(5000-4)/1000˜ 5us

Initial am decis sa incercam si folosirea modulului PWM, iar în foile de câta log ale PIC16F876 aflam formula pentru calculul perioadei semnalului, raportat la frecvent a oscilatorului intern:

T_{PWM} = (PR2 +1)* 4*Tosc* PRESCALER

Unde: Tosc*4 = 1us

Prescaler ia valoarea maxima de 1:16

PR2 este un registru pe 8 biti, deci cu valoare maxima 256

Incercam sa calculam ce valoare trebuie încar câta în registrul PR2 care e în mod direct asociat cu timer2 astfel încât sa obtinem frecvent ele minime si maxime de 10Hz si 2kHz:

Pentru o frecvent a de 10Hz, perioada e de 100ms, deci:

PR2+1= 100ms/(16*1us) = 6250

Dar 6250 este un numar care se scrie în binar pe 13 biti, deci nu poate fi folosit timer2 si modulul PWM, pentru generarea semnalului PWM, dar nici pentru generarea semnalului dreptunghiular.

Cel mai bine ar fi sa folosim pentru generarea semnalului PWM timer1 care poate numar a pana la 2^{16} , deci cu care vom putem numar a usor 2^{13} tacturi.

Asta insemna ca înca mai avem de rezolvat problema semnalului dreptunghiular folosindu-ne doar de timer2, care poate atinge pana la 256 tacte.

Cel mai util mod de a rezolva problema semnalului dreptunghiular este de a introduce, pe langa registrul PR2 asociat timerului2, înca un contor, în care sa încarcam o parte din date.

Daca vrem ca perioada semnalului dreptunghiular sa fie de forma:

$T_{SQ} = T_{clk} * M$

unde M este numar ul de tacte pe care trebuie sa le transmitem astfel încât sa fie echivalente cu semiperioada semnalului nostru, T_{SQ} este perioada semnalului dreptunghiular, si T_{clk} este 1us, adica perioada unui tact (un ciclu masina).

M ia valori între 50 000 si 250. în acest caz scriem M ca produs de doua numere pe care le memoram în memoria de program, si pe care le asociem cu PR2 si cu CONTOR. sa luam de exemplu cazul lui M=50 000, putem spune ca PR2=250 si CONTOR=200.

Sau pentru M=250, punetm spune ca PR2=250 si CONTOR= 1, sau PR2= 50 si CONTOR=50 sau orice alta combinatie de numere scrise pe maxim 8 biti care inmultite sa ne dea valoarea M.

Important de retinut pentru a intelege principiul de functionare al timerul 2, care este asociat cu generarea semnalului dreptunghiular, este faptul ca timerul se va incrementa cu câte o pozitie atata timp cat decrementand valorile din contoare nu ajung ambele pe 0, chiar daca timerul 2 trece prin mai multe depasiri ale domeniului sau.. Decrementarea contoarelor trebuie inteleasa ca si decrementarea unui numar care are atat unitati cat si zeci, mai intai decrementam unitati le, apoi zecile si revenim la unitati pana ce ambele sunt zero. în cazul nostru putem considera PR2 reprezentantul unitati lor, iar CONTOR reprezentantul zecilor, decrementarea celor doua se va face în felul urmator:

Sa consideram PR=250, iar CONTOR=200 (pentru M=50 000)

Decrementarea se face astfel: PR2=249 CONTOR=200
PR2=248 CONTOR=200
.....
PR2=0 CONTOR=200
PR2=250 CONTOR=199
PR2=249 CONTOR=199
.....

PR2=1 CONTOR=0
PR2=0 CONTOR=0

Principiul de realizare a software-ului urmeaza pasii:

- Citire de la cele 3 intrari analogice (2 sunt pentru ajustarea frecvent ei si una pentru ajustarea factorului de umplere)
- Conversie analog-numerica , în urma careia rezulta 3 numere pe 10 biti care vor fi pointeri la adresele tabelelor de unde citesc datele cu care pot stabili frecvent ele si factorul de umplere al celor 2 semnale.
- Citirea din tabele. Am 4 tabele, o tabela de viteza (a carei intrare vine de la potentiometrul de modificare a frecvent ei semnalului PWM) si una de durata a impulsului(pentru modificarea factorului de umplere) A treia tabela este cea cu datele care trebuie introduse în PR2 si cea de-a patra cu datele de introdus în CONTOR(pentru care intrarea o consti tuie potentiometrul de modificare a frecvent ei semnalului dreptunghiular).
- Generarea semnalului PWM , calculand perioada semnalului, prin folosirea ca si intrare a rezultatelor din tabelele de viteza si de durata a impulsului. Folosim timer1. Realizam operatia :

  $$T_{PWM}= 255\ t_{high}/v$$

  si transmitem pe iesirea din PORTB, RB1, "1 logic" pe durata t_{high} , si "0 logic" pe durata $T_{PWM} - t_{high}$.
- Generarea semnalului dreptunghiular prin calculul semiperioadei semnalului, care se face cu ajutorul contoarelor PR2 si CONTOR, prin intermediul timerului 2. Transmitem pe la iesirea RC1 a portului C "1 logic" pe o durata de (PR2*CONTOR)us, si aceeasi durata de timp transmitem apoi "0 logic".

Pentru o mai buna intelegere a structurii programului vom realiza o schema logica si vom explica programarea anumitor registre.

3.4.1 Schema logica a programului

Deoarece avem de generat doua semnale,pentru a intelege mai bine structura programului, vom realiza doua scheme logice, una pentru semnalul PWM si una pentru semnalul dreptunghiular

Schema logica pentru generarea semnalului PWM

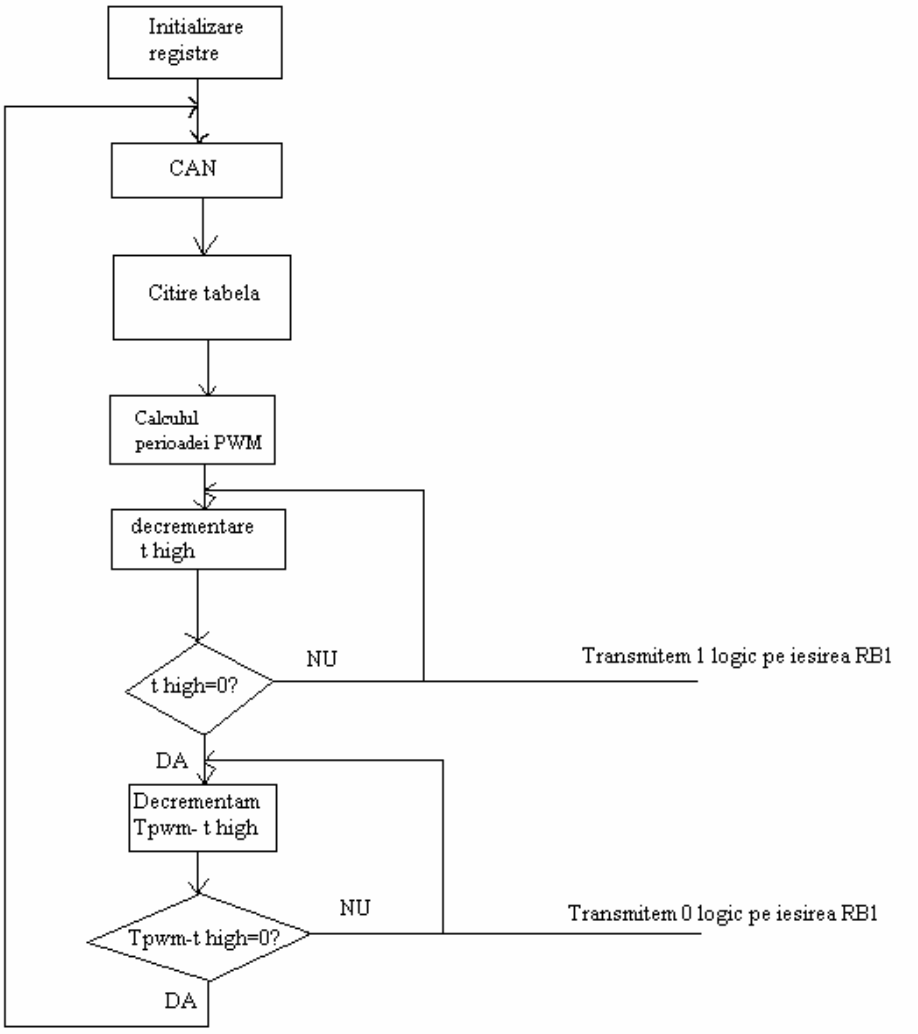


Fig.16 Schema logica pentru generarea semnalului PWM

Schema logica pentru generarea semnalului dreptunghiular

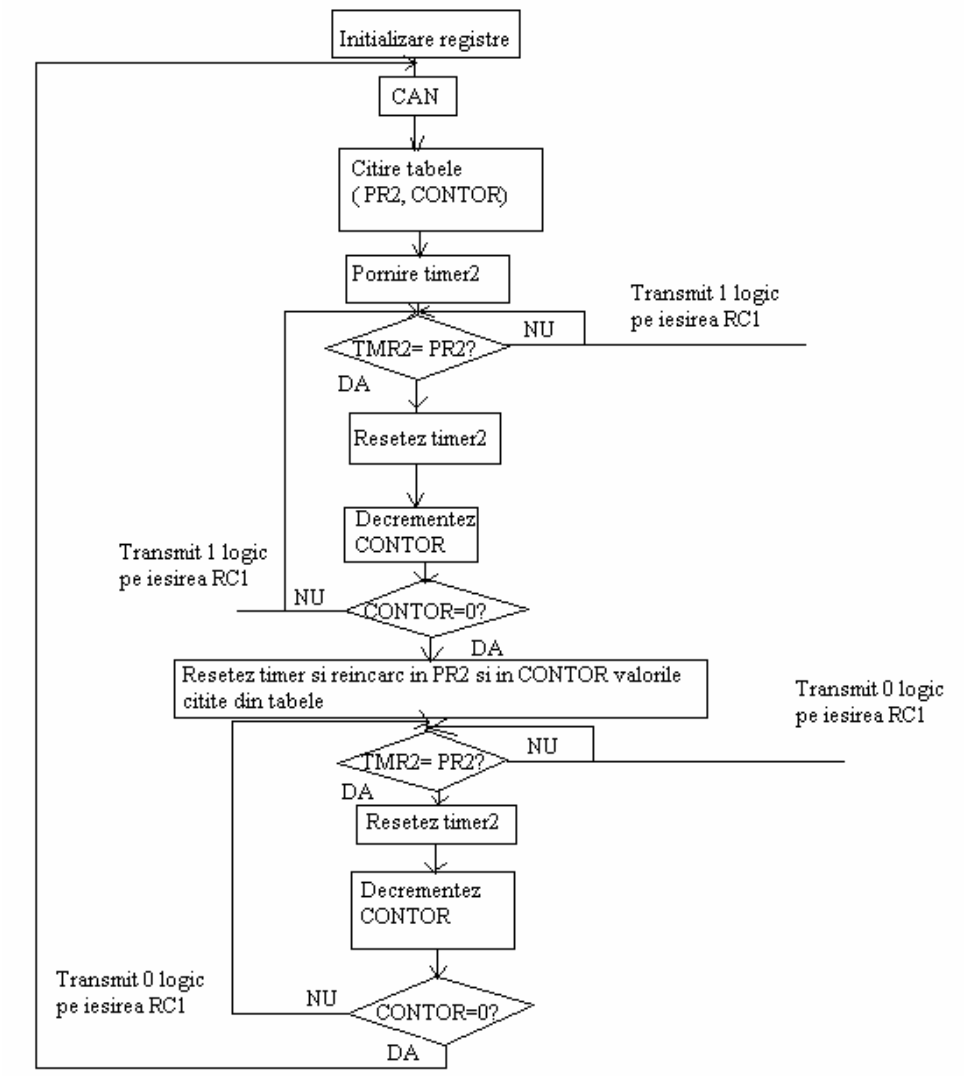


Fig. 17 Schema logica pentru generarea semnalului dreptunghiular

În cele ce urmeaza vom stabili cu ce valori trebuie sa initial izam registrele astfel sa putem genera cele doua semnale.
Incepem cu registrele pe care le folosim la intreruperi:

- INTCON, cu care validam intreruperile globale prin setarea pe 1 a bitului 7, GIE.
- PIE1, registru ce activeaza intreruperile periferice, si la care vom seta bitii 6, ADIE, care activeaza intreruperile de la convertorul analog-numeric, bitul 1, TMR2IE si bitul 0, TMR2IE care activeaza intreruperile de la timer2 si timer1. Asadar, vom încarca în registrul PIE1 valoarea:

  0100 0011= 43h

- De asemenea, pe parcursul executie programului, pentru a vedea daca s-a produs o intrerupere de la CAN, sau de la timere, verificam daca bitii ADIF, TMR1IF si TMR2IF din cadrul registrului PIR1 sunt pe 1.
- În registrul OPTION_REG dezactivam intreruperile de la portul B, punând pe 1 bitul 7, NOT_RPBU, adica încarcând în acest registru valoarea 80h

În ceea ce priveste porturile A, B, si C valorile pe care le încarcam în registrele corespunzatoare sunt:

- Pentru PORTA, avem nevoie de 3 intrari analogice, însa putem seta toate cele 6 intrari ca si intrari anlogile, deci punem pe 1 ultimi 6 biti ai registrului corespunzator portului A, TRISA:
  0011 1111= 3Fh
- Pentru PORTB, avem nevoie de o iesire digitala, RB!, asta inseamna ca vom pune pe 0 bitul 1 al registrului TRISB si si toti ceilalti pini ai portului B ii lasam ca si intrari, deci ii punem pe 1. în TRISB vom inscrie valoarea:
  1111 1101= FDh
- Pentru PORTC, folosim pinul RC! ca si iesire digitala, toti ceilalti pini fiind considerati intrari, deci în TRISC vom seta pe 0 bitul 1 si toti ceilalti pe 0:
  1111 1101= FDh

Urmeaza sa facem setarile pentru modulul de conversie analog-numerica, si în acest sens folosim registrele:

- ADCON1, unde vom pune pe 1 bitul 7, ADFM, care determina scrierea spre dreapta a numar ului convertit, adica 8 biti din 10 sunt scrisi în registrul ADRESL si ultimi 2 biti în ADRESH, si deoarece am setat toate intrarile portului A ca si analogice, ceilalti biti vor ramane pe 0, deci în ADCON1 se încarca valoarea:
  1000 0000=80h
- ADCON0, punem bitii ADCS1, 0 pe 01, selectand pentru conversie o frecvent a de Fosc/8.

  Tot aici selectam si canalul de lucru, prin bitii 5,4,si 3. De exemplu, pentru canalul 0 inscriem în ADCON0 valoarea: 0100 0000= 40h
  Pentru canal1: 0100 1000= 48h
  Pentru canal2: 0101 0000= 50h

  Tot în cadrul acestui registru, pentru a activa conversia analog-numerica trebuie sa setam pe 1 bitul 0, ADON.
- În cadrul conversiei vom mai folosi registrele ADRESL si ADRESH, de unde vom citi valorile convertite.

În ceea ce priveste timerele pe care le folosim, setarile se fac în felul urmator:

- Pentru timer1, pe care îl folosim la generarea semnalului PWM, setam registrul T1CON , punând pe 1 bitul3, T1OSCEN, cu care activam oscilatorul, setam TMR1CS pe 0, activand clock-ul intern(Fosc/4) , iar bitul 0, TMR1ON îl punem pe 1 pentru a activa acest timer. Deci în acest registru inscriem valoarea:
  0000 1001= 09h
- Pentru timer2, pe care îl folosim la generarea semnalului dreptunghiular, setam registrul T2CON , punând pe 1 bitul 3, TMR2ON, adica pornim timerul, iar în ceea ce priveste ceilalti biti ii punem pe 0, deoarece nu folosim o alta valoare de prescaler si postscaler diferita de 1:1. Asadar în T2CON inscriem valoarea :
  0000 0100= 04h

Citirea din tabele va implica folosirea urmatoarelor registre:

- EECON1, în care pentru a reusi sa citim valorile din memorie va trebui sa activam bitul 7, EEPGD, care face selectia memoriei de program în care avem memorate datele, si bitul 0, RD, care activeaza citirea din aceasta memorie.
- Adresele de la care citim datele sunt indicâte de registrele EEADR si EEADRH
- Datele pe care le citim sunt citite din registrele EEDATA si EEDATH, care corespund adreselor indicâte de registrele EEADR si EEADRH.

În ceea ce priveste variabilele introduse de noi în program, le vom initial iza pe toate pe valoarea 00h.

De asemenea precizam ca pentru selectia bancului de lucru folosim aceeleasi macrouri ca la programul folosit la generarea semnalului sinusoidal. Alegerea bancului de lucru se face prin setarea a 2 biti din cadrul registrului STATUS : bitii 5 si 6, RP0 si RP1:

Pentru banc0: RP0=0, RP1=0
Pentru banc1 : RP0=1, RP1=0
Pentru banc2 : RP0=0, RP1=1
Pentru banc3 : RP0=1, RP1=1

Pentru a avea o privire de ansamblu asupra modului de utilizare a microcontrollerului am atasat în cadru Anexe, ca si Anexa 4 câteva din foile de câta log ale PIC16F876.

3.4.2 Programul realizat pentru PIC16F876 pentru generarea semnalelor dreptunghiular si PWM

```
list P=pic16F876


#include <p16f876.inc>

;************************ Macrouri de comutare bancuri*******************************

bank0  macro                                    ;memoria PIC-ului este impartita în 4 bancuri,
               bcf             STATUS,     RP0  ; în cadrul fiecarui banc putandu-se face diverse
               bcf             STATUS,     RP1  ; operatii asupra deverselor registre
       endm                                     ;Pentru a folosi un anumit registru trebuie ca în
                                                ;cadrul memoriei sa ne aflam în bancul unde se
bank1  macro                                    ; afla si acel registru, altfel operatiile nu se
               bsf             STATUS,     RP0  ;efectueaza asupra registrului
               bcf             STATUS,     RP1
       endm

bank2  macro
               bcf             STATUS,     RP0
               bsf             STATUS,     RP1
       endm

bank3  macro
               bsf             STATUS,     RP0
               bsf             STATUS,     RP1
       endm


;***************************Definim variabile**********************

ADLS0          equ     0x20            ; Definim variabila de achizitie a CAN-ului pt canal 0
ADHS0          equ     0x21            ; Valoarea de la CAN, MSByte
ADLS1          equ     0x22            ;Definim variabila de achizitie a CAN-ului de la canal1
ADHS1          equ     0x23            ; Valoarea de la CAN, MSByte
ADLS2          equ     0x24            ;Definim variabila de achizitie a CAN-ului de la canal2
ADHS2          equ     0x25            ;Valoarea de la CAN, MSByte
TABLE0L        equ     0x26            ; Pointerii care imi indica adresa din  tabele de la care
TABLE0H        equ     0x27            ; trebuie sa citesc datele pentru perioada si factor de
TABLE1L        equ     0x28            ; umplere
TABLE1H        equ     0x29
TABLE2L        equ     0x2A
TABLE2H        equ     0x2B
NUMAR L        equ     0x2C            ; variabile folosite la impartirea T_HIGH la viteza
NUMAR H        equ     0x2D
NUMIT          equ     0x2E
```

```
REST        equ     0x2F

;*****************************Definim variabile**************************


CONTOR      equ     0x30        ;Variabila care alaturi de PR2 imi determina frecvent a
VITEZAL     equ     0x31        ; Viteza de deplasare a masinii, se citeste din  tabela
VITEZAH     equ     0x32
T_HIGHL     equ     0x33        ; LSByte a duratei impulsului pt semnalul PWM
T_HIGHH     equ     0x34        ; MSByte, se citesc din  tabela
PERIODL     equ     0x35        ; Perioada semnalului PWM= 255*t_high/v
PERIODH     equ     0x36        ; MSByte a perioadei
PERBUFL     equ     0x37        ; buffer pe care îl  folosesc la calcule unde e impli câta
PERBUFH     equ     0x38        ; si perioada semnalului PWM
T_LOWL      equ     0x39        ; durata cat semnalul PWM e "0" si care e egala cu
T_LOWH      equ     0x3A        ; t_low=perioada-t_high
CONTORBUF   equ     0x3B        ; un buffer în care facem calcule
DATAL       equ     0x40
DATAH       equ     0x41
DIV         equ     0x42
DIVCNT      equ     0x43
CNT         equ     0x44



;************************definim tabela0 pt SQUARE_HIGH(PR2)*********************


            org     0x1000
SQUAREPR2
        dw      0x00FA              ; retinem ca PR2*CONTOR= M, unde M e durata
        dw      0x00F9              ; impulsului semnalului dreptunghiular
        dw      0x00F8
```

..... tabela pentru contorul PR2 se poate vedea la Anexe, ca si Anexa 8.....

```
; ********definim adresa de start pt tabela1 pentru semnalul  SQUARE contor***********

            org   0x1400
SQUAREC     dw    0x00C8
            dw    0x00A7
            dw    0x0090
```

.....tabela pentru CONTOR se poate citi la Anexe, ca si Anexa 8.....

```
;*************** DEFINIM TABELA 2 PT VITEZA*****************************

              org    0x1800              ;definim adresa de start pt tabela de viteza a automobilului

VITEZA        dw     0x00                ; viteza de deplasare ia valori între 0 si 120km/h
              dw     0x01
              dw     0x02
              dw     0x03
.....tabelul poate fi citit în cadru Anexelor, la Anexa 9.....


;*******************definim tabela3 de T_HIGH************************

              org 0x1C00
t_high        dw     0x0000              ; durata impulsului semnalului PWM ia valori intre
              dw     0x0005              ; 0 si 5 ms cu pas de 5us
              dw     0x000A
              dw     0x000F

.....tabelul poate fi citit în cadrul Anexelor, la Anexa 10  .....


; *******************incepem programul propriu-zis*******************************


              org 0x0000
              nop
              nop
              nop
              goto         start


;***********************rutina de tratare a intreruperilor**********************
intrp         bcf    INTCON, GIE          ; invalidare intreruperi
              btfss   PIR1, TMR1IF        ; testare intrerupere TMR1
              goto    endint
              bcf     PIR1, TMR1IF        ; stergere flag intrerupere de la TMR1
              btfss   PIR1, TMR2IF        ; testare intrerupere TMR2
              goto    endint
              bcf     PIR1, TMR2IF        ; stergere flag intrerupere de la TMR2
endint        retfie


;*******************sfarsitul subrutinei de intrerupere si start program ******************
start         call   pginit
              nop
              call    main
              nop
```

```
;**********************rutina de initial izare a controllerului*****************************
pginit      bank1
            movlw       0x0080              ;dezactivez intreruperile de la PORTB
            movwf       OPTION_REG          ;inscriind 80H în registru
            movwf       ADCON1              ;Selectam modul de scriere spre dreapta
            movlw       0x003F              ; activez toate intrarile anlogice ale
            movwf       TRISA               ; portului A
            movlw       0x00FD              ; Setez pinul RB1 ca si iesire si toti
            movwf       TRISB               ;ceilalti ca si intari
            movwf       TRISC               ; setez pinul RC1 ca si iesire, si toti
                                            ; ceilalti pini ca si iesiri
            bank0
            movlw       0x0009              ; activez TMR1, având ca si clock
            movwf       T1CON               ;Fosc/4, adica o durata de 1us
            movlw       0x0004              ; activez TMR2 ce functioneaza tot cu
            movwf       T2CON               ; clock intern, durata tactului de 1us

            bank1
            movlw       0x0000
            movwf       PR2                 ;contorul PR2 îl  stergem, îl  punem pe 0
            bank0
            movlw       0x0000              ;initializam pe 0 toate variabilele
            movwf       ADLS0               ;declarate anterior
            movwf       ADHS0
            movwf       ADLS1
            movwf       ADHS1
            movwf       ADLS2
            movwf       ADHS2
            movwf       ADLS3
            movwf       ADHS3
            movwf       TABLE0L
            movwf       TABLE0H
            movwf       TABLE1L
            movwf       TABLE1H
            movwf       TABLE2L
            movwf       TABLE2H
            movwf       DEIMP
            movwf       IMP
            movwf       CAT
            movwf       REST
            movwf       VITEZAL
            movwf       T_HIGHL
            movwf       T_HIGHH
            movwf       T_LOWL
            movwf       T_LOWH
            movwf        PERIODL
            movwf       PERIODH
```

```
                movwf       PERBUFL                 ; e un buffer în care fac calcule
                movwf       PERBUFH
                movwf       CONTOR
                movwf       CONTORBUF
                bank1
                clrf        INTCON                  ; dezactivez orice fel de intrerupere
                movlw       0x0043                  ; activez intreruperile de la ADC, TMR1
                movwf       PIE1                    ; (overflow), TMR2 (PR2 match)
                bsf         INTCON, GIE             ; activez intreruperile globale


;**************************procedura de conversie a CAN-ului**********************


conv            bank0

;************************  pentru canal0  **********************

                movlw       0x0040          ;setam canalul de la care se face
                movwf       ADCON0          ; citirea, timpul de conversie, si ADON=0
                bcf         PIR1, ADIF      ; flag de intreruperi
                bsf         ADCON0,ADON     ; ADON=1
                bsf         ADCON0, NOT_DONE
loopconv0       btfss       PIR1, ADIF      ; când ADIF=1 nu executa urmatoarea
                goto        loopconv0       ; instructiune, când ADIF=0 executa bucla
                bank1
                movfw       ADRESL          ; transferam valorile citite la iesirea CAN
                bank0                       ; în variabilele declarate mai devreme
                movwf       ADLS0           ;
                movfw       ADRESH          ;
                movwf       ADHS0
                bcf         ADCON0, ADON    ; oprim conversia

; **************************canal 1***********************************

                movlw       0x0048          ; setam canalul de la care se face conversia
                movwf       ADCON0
                bcf         PIR1, ADIF      ;flag de intreruperi
                bsf         ADCON0, ADON    ; convertorul analog-numeric este pornit
                bsf         ADCON0, NOT_DONE    ; conversia nu e  realizata
loopconv1       btfss       PIR1, ADIF      ; când ADIF=1 nu executa urmatoarea
                goto        loopconv1       ; instructiune, când ADIF=0 executa bucla
                bank1
                movfw       ADRESL          ;transferam datele citite la iesirea CAN
                bank0
                movwf       ADLS1           ;În  variabilele declarate anterior

                movfw       ADRESH
```

```
                movwf       ADHS1
                bcf         ADCON0, ADON    ;oprim convertorul
;*********************************canal2*************************
                movlw       0x0050          ; selectam canalul2 pentru conversie
                movwf       ADCON0
                bcf         PIR1, ADIF      ; stergem flagul de intreruperi
                bsf         ADCON0, ADON    ; pornim convertorul
                bsf         ADCON0, NOT_DONE
loopconv2       btfss       PIR1, ADIF      ;verificam daca s-au produs intreruperi
                goto        loopconv2       ; de la CAN
                bank1
                movfw       ADRESL          ; transferam datele la variabilele declarate
                bank0                       ;anterior
                movwf       ADLS2
                movfw       ADRESH
                movwf       ADHS2
                bcf         ADCON0, ADON    ;oprim convertorul
                return                      ; intoarcere în programul principal
                nop


;*****************  Procedura de citire din  TABELA0 (PR2)  ***************************


read_table0
                bank0
                movf        TABLE0L,     W          ; citim de la adresa 1000+ adresa
                bank2                               ; indi câta  de iesirea ADC valoarea din
                movwf       EEADR                   ; registru PR2
                bank0
                movf        TABLE0H, W
                addlw       0x10
                bank2
                movwf       EEADRH
                bank3                               ;acceseaza memoria de program unde
                bsf         EECON1, EEPGD           ; am sto câta  tabela 0
                bsf         EECON1, RD              ; activez citirea
                nop
                nop
                bank2
                movf        EEDATA, W               ; PR2 fiind un registru pe 8 biti
                bank1                               ; datele sunt stocâte numai în EEDATA
                movwf       PR2                     ; nu si în EEDATH
                return
                nop
```

```
;***************procedura de citire din  TABELA 1(CONTOR)**********************

read_table1
            bank0                               ; citim de la adresa 1400+ adresa
            movf        TABLE1L,     W          ; indi câta  de iesirea CAN valoarea
            bank2                               ; CONTORULUI
            movwf       EEADR
            bank0
            movf        TABLE1H, W
            addlw       0x14
            bank2
            movwf       EEADRH
            bank3                               ; acceseaza memoria de program unde
            bsf         EECON1, EEPGD           ; am stocat tabela1
            bsf         EECON1, RD              ; activez citirea
            nop
            nop
            bank2
            movf        EEDATA, W
            bank0                               ; datele le preiau din  registrul
EEDATA,
            movwf       CONTOR                  ; CONTOR fiind pe 8 biti
            return
            nop


;***********************procedura de citire din  TABELA2 (VITEZA) **********************

read_table2                                     ; citesc VITEZA  de deplasare a
            bank0                               ; autovehiculului
            movf        TABLE2L,     W          ; citim de la adresa 1800+ adresa
            bank2                               ; indi câta  de iesirea CAN valoarea
            movwf       EEADR                   ;vitezei
            bank0
            movf        TABLE2H, W
            addlw       0x18
            bank2
            movwf       EEADRH
            bank3                               ; acceseaza memoria de program unde
            bsf         EECON1, EEPGD           ; am salvat tabelul 2
            bsf         EECON1, RD              ; activez citirea
            nop
            nop
            bank2
            movf        EEDATA, W
            bank0
```

```
                movwf       VITEZAL                 ; transfer datele citite în VITEZAL
                return                              ; intoarcere la programul principal
                nop
;**********************procedura de citire din  tabela 3( T_HIGH )*********************


read_table3                                         ; citesc din  tabela2 durata HIGH a
                                                            ; semnalului PWM pe care vreau
sa-l
                bank0                               ; generez
                movf        TABLE3L,W
                bank2
                movwf        EEADR                  ; aceasta valoare o citesc de la adresa
                bank0                               ; 1C00+ adresa indi câta  de ADC, din
                movf        TABLE3H, W              ; memoria de program
                addlw       0x001C
                bank2
                movwf       EEADRH
                bank3
                bsf         EECON1, EEPGD           ; acceseaza memoria de program
                bsf         EECON1, RD              ; activez citirea
                nop
                nop
                bank2
                movf        EEDATA, W
                bank0
                movwf       T_HIGHL
                bank2
                movf        EEDATH, W
                bank0                               ; transfer datele citite din  tabel în
                movwf       T_HIGHH                 ; variabila T_HIGH
                return                              ; reintoarcere în programul principal
                nop

;***********procedura impartire pt calcului perioadei PWM=255*viteza/t_high**********


divide          movf        DATAH,      W           ;intoduc datele în numar ator
                movwf       NUMAR H                 ;care e în cazul meu T_HIGH
                movf        DATAL,      W           ;intoduc datele la numitor
                movwf       NUMAR L                 ; care e VITEZA
                movf        DIV,    W
                movwf       NUMIT
                clrf        REST
                movlw       d'16'                   ; impartirea se poate facem pt numere de
                movwf       DIVCNT                  ; cel mult 16 biti
nextd1          bcf         STATUS,     C           ;procedura de impartire consta intr-o
                rlf         NUMAR L,    F           ;bucla de siftari
                rlf         NUMAR H,    F
```

```
            rlf     REST, F                 ; shift MSb
            movf    NUMIT, W                ; w=nev
            bcf     STATUS, C
            subwf   REST, W                 ; w=mar-nev
            btfss   STATUS, C
            goto    $+3
            movwf   REST
            bsf     NUMAR L,    0
            decfsz  DIVCNT, F
            goto    nextd1
            return
            nop
;*********************************inmultire*************************
inmultire   comf    NUMAR L, W          ;255*X= 256*X-X
            addlw   1                   ;obtinem înainte  –X= -THIGH/VITEZA
            movwf   NUMAR L
            comf    NUMAR H, W
            btfsc   STATUS, C
            addlw   1
            movwf   NUMAR H
            movf    CNT, W              ; inmultirea unui numar  cu 256=2^8
            movlw   d'8'                ; inseamna siftarea numar ului cu 8
            movwf   CNT                 ; pozitii la stanga
            bcf     STATUS, C
siftare     rlf     PERIODL, F
            nop
            rlf     PERIODH, F
            nop
            decfsz  CNT, F
            goto    siftare
            movfw   NUMAR L
            addwf   PERIODL, F              ;facem adunare între numar ul siftat
            movfw   NUMAR H                 ; si numar ul negativ
            btfsc   STATUS, C
            addlw   1
            addwf   PERIODH, F
            return
            nop

;******************************** programul principal ******************************
main
            bank0
            call    conv                    ;apelam subrutina de conversie
            nop

;************generarea semnalului PWM, cu frecventa si duty-cycle variabil************
            movf    ADLS1, W                ; transfer ceea ce am citit de la ADC la
            movwf   TABLE2L                 ; intrarea tabelelor
            movf    ADHS1, W                ;citim datele despre viteza si durata
```

```
                movwf       TABLE2H                 ; impulsului la semnalul PWM
                movf        ADLS2,W
                movwf       TABLE3L
                movf        ADHS2, W
                movwf       TABLE3H
                call        read_table2             ; citesc valoarea vitezei(pe 7 biti)
                nop
                call        read_table3             ; citesc valoarea t_high (pe 13 biti)
                nop
                movfw       VITEZAL                 ; transfer datele citite în variabilele
                movwf       DIV                     ;desti nate realizarii impartirii
                movfw       T_HIGHL
                movwf       DATAL
                movfw       T_HIGHH
                movwf       DATAH
                call        divide                  ;apelez subrutina de impartire
                movfw       NUMAR L                 ;NUMAR L= T_HIGH/VITEZA
                movwf       PERIODL
                movfw       NUMAR H
                movwf       PERIODH
                call        inmultire               ;PERIOD=255* t_high/viteza
; calculam valoarea  T_LOW, adica durata în care semnalul este 0(low)   T_LOW= PERIOD-
T_HIGH
                bcf         STATUS, C
                comf        T_HIGHL, W              ; aplicam cod complementul lui 2  pt a
                addlw       1                       ;determina " –T_HIGH"
                movwf       T_LOWL
                comf        T_HIGHH,W
                btfsc        STATUS, C              ;când avem tansport de la LSByte
                addlw       1                       ;efectuam adunarea cu 1, daca nu
                movwf       T_LOWH                  ;trecem la urmatoarea instructiune
                movf        PERIODL, W
                addwf        T_LOWL
                movf        PERIODH, W
                btfsc       STATUS, C
                addlw       1
                addwf       T_LOWH
;calculam valoarea ce trebuie încar câta  în TMR1 astfel încât sa transmitem semnal cu frecvent a
aleasa de noi
; în TMR1 încarcam valoarea 65536 (0xFFFF= 2^16)- PERIODL,H
                bcf         STATUS, C               ;stergem bitul de transport
                comf        PERIODL,W               ;aplicam cod complementul lui 2
                addlw       1                       ; pentru a determina "-PERIOD"
                movwf       PERBUFL                 ; operatiile le fac în PERBUF
                comf        PERIODH,W
                btfsc       STATUS, C
                addlw       1
                movwf       PERBUFH
                bcf         STATUS, C
```

```
                movf        PERBUFL, W          ; aici memorez  valoarea pe care o
                addlw       0xFF
                movwf       PERBUFL
                movf        PERBUFH, W
                btfsc       STATUS, C
                addlw       1
                addlw       0xFF
                bcf         STATUS, C
                movwf       PERBUFH
; invalidez intrerupere de la TMR1 daca exista deja una
                bcf         PIR1, TMR1IF
                movf        PERBUFL, W
                movwf       TMR1L
                movf        PERBUFH, W
                movwf       TMR1H
                btfss       PIR1, TMR1IF        ; intrerupere datorita overflow la TMR1
                                                ; ( adica TMR1 a ajuns la 65536)
                bsf         PORTB, 1            ; transmitem prin pinul 1 al PORTB 1 pe
                                                ; durata t_high si 0 pe durata
                                                ;  T_LOW= PERIOD-T_HIGH
looph           decfsz      T_HIGHL             ; cat timp T_HIGH e diferit de 0
                                                ; transmitem prin PORTB valoarea 1
                goto        looph
                decfsz      T_HIGHH
                goto        looph
loophl          decfsz      T_HIGHL
                goto        loophl
                bcf         PORTB,1             ; când T_HIGH devine zero, pe o durata
loopl           decfsz      T_LOWL              ; T_LOW transmitem prin PORTB
                                                ; valoarea 0
                goto        loopl
                decfsz      T_LOWH
                goto        loopl
loopll          decfsz      T_LOWL
                goto        loopll


; ************************  generarea semnalului SQUARE  ************************
                movlw       0x00
                movwf       TMR2
                movlw       0x0004
                movwf       T2CON
;testare ADLS0
                movlw       0xE3
                movwf       ADLS0
;GATA
                movf        ADLS0, W
                movwf       TABLE0L             ; ceea ce citesc de la ADC imi va fi pointer
                                                ; pentru intrarile tabelelor 0 si 1
```

```
                movwf       TABLE1L             ; adica tabelul de PR2 si de CONTOR
                movf        ADHS0, W
                movwf       TABLE0H
                movwf       TABLE1H
                call        read_table0                 ; citesc PR2 si CONTOR
                nop
                call        read_table1
                nop
                movf        CONTOR, W
                movwf       CONTORBUF

                bcf         PIR1, TMR2IF
                bank1
                bsf         PIE1, TMR2IE
                bank0
                bsf         PORTC, 1

loop1
                btfss       PIR1,TMR2IF
                goto        loop1
                bank0
loop11          decfsz      CONTORBUF
                goto         loop11
;*********************************************************************************
;dupa ce am intalnim intrerupere de la timer2(PR2=TMR2), repun TMR2 pe 0 si incep numaratoarea
;pt partea low a semnalului SQUARE*************************************
                movlw       0x00
                movwf       TMR2

                bcf         PIR1, TMR2IF
                bank0
                bcf         PORTC,1
loop0           btfss       PIR1, TMR2IF
                goto        loop0
                bank0
loop00          decfsz      CONTOR
                goto        loop00
                goto        main
                nop


                end
```

Capitolul 4
Simulare si testare

4.1 Simularea generatorul de semnal sinusoidal

Generatorul de semnal sinusoidal e realizat din mai multe blocuri functinale. Pentru simularea PIC16F876 folosim un mediu de programare realizat de Microchip, MPLAB ide 7.50v, in care scriem si programul, si de asemenea putem vedea si evolutia registrelor la rularea programului.

De asemenea, pentru a sesiza erorile în timpul rularii programului, putem folosi un dispozitiv numit „in circuit debugger” prin intermediul caruia sesizam in timp real reactiile microcontrollerului. Si acest dispozitiv este realizat tot de Microchip.

În ceea ce priveste partea electronica, si nu cea software, pentru simulare folosim programul Pspice, Orcad Capture. Din pacate cu Pspice putem simula numai partea de amplificare si etajul final al generatorului de semnal sinusoidal.

În acest sens desenam schema cu Orcad Capture, si ii introducem pe la intrare un semnal sinusoidal cu frecventa maxima de 10 kHz. Acest semnal reprezinta semnalul generat de AD9833 si are amplitudinea vârf la vârf de 0,6 V.

Pentru a simula variatia de amplitudine, vom folosi schema din urmatoarea figura, in care prin variatia potentiometrului R28 ce are valoarea maxima de 4,7 k? cu un pas de 500? obtinem la iesire mai multe semnale sinusoidale cu frecventa egala cu frecventa semnalului de la intrarea blocului amplificator, însa amplificat de câteva ori.

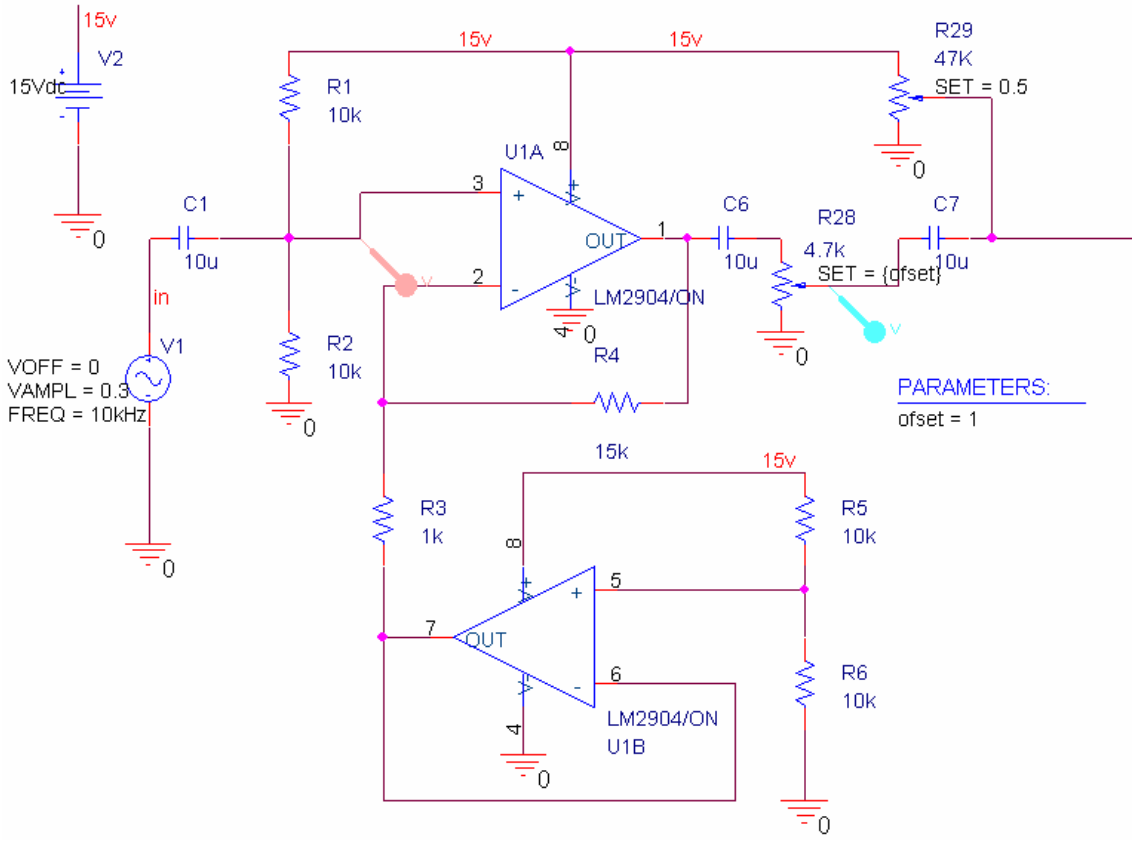


Fig. 18 Schema pentru simularea variatiei amplificarii