DEZVOLTAREA APLICAŢIILOR SUB CODE COMPOSER STUDIO®

1. Prezentare generală

Code Composer Studio **(CCS®)** este un instrument foarte eficace pentru dezvoltarea rapidă şi performantă a aplicaţiilor scrise pentru procesoarele de semnal din familiile TMS320C54x, C55x, C6x. **CCS** permite:construirea, configurarea, testarea, rularea şi analiza în timp real a aplicaţiilor. **CCS** facilitează rularea aplicaţiilor pe diferite sisteme de dezvoltare sau pe simulator. Selecţia sistemului pentru care se dezvoltă aplicaţiile se face prin intermediul componentei **Setup CCStudio.**

Fazele dezvoltării unei aplicaţii sub **CCS** sunt ilustrate în fig.1.

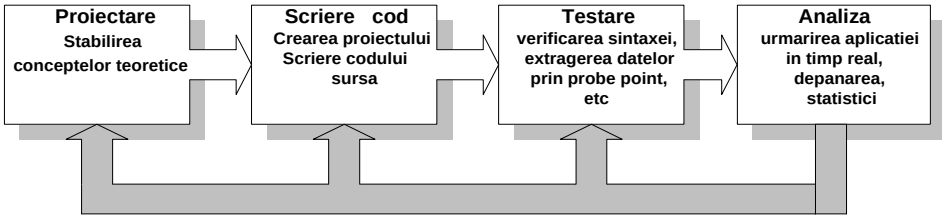


Fig. 1 Fazele dezvoltării unei aplicaţii sub CCS

Code Composer Studio include următoarele componente:

- **TMS 320C54X Code Generation Tools** – Conţine principalele elemente software necesare în dezvoltarea aplicaţiilor sub **CCS**
- **CCS Integrated Development Enviroment (IDE)** - mediul care integrează şi gestionează toate componentele CCS care permit proiectarea, editarea şi depanarea aplicaţiilor.
- **DSP/BIOS -** sunt bibloteci de funcţii care permit comunicarea cu aplicaţia în timpul rulării pe sistemele hardware. Componenta DSP/BIOS este structurată în două secţiuni:

  -DSP/ BIOS Plug-Ins oferă posibilitatea de analiză a aplicaţiilor pentru estimarea performanţelor cu impact minim asupra performanţelor acestora în timp real.

  -DSP/BIOS API (Application Interfaces) **–** furnizează componente software care pot fi apelate prin intermediul sursei aplicaţiei

  Componentele API oferă posibilitatea de interfaţare a aplicaţiilor ce rulează pe sistemul DSP cu aplicaţii externe care rulează pe PC.

Toate aceste componente lucrează împreună după cum este ilustrat în fig. 2.

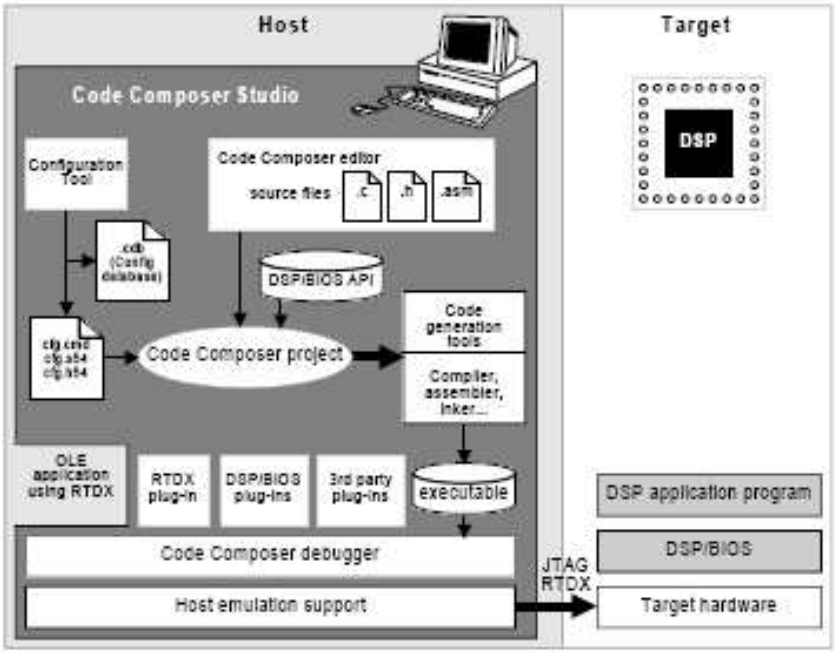


Fig.2 Componentele utilizate la dezvoltarea unei aplicaţii sub CCS

2. Descrierea utilitarelor software

Majoritatea utilitarelor software utilizate în procesul de dezvoltatre a unei aplicaţii sub CCS sunt gestionate de mediul IDE. Alte componente sunt apelate ulterior pe parcursul dezvoltării aplicaţiei.

Fluxul de dezvoltare a unei aplicaţii este ilustrat în Fig 3.

- **Compilatorul C** - acceptă sursa C în standardul ANSI şi generează fişiere sursă .ASM
- **Asamblorul –**translatează fişierele sursă ASM în fişiere obiect COFF
- **Linkeditorul –**combină fişierele obiect într-un singur modul obiect executabil
- **Arhivorul –**colectează grupuri de fişiere într-un singur fişier bibliotecă (arhivă). Arhivorul poate modifica bibloteca în sensul: stergerii, reînscrierii, extragerii sau adăugării unui nou membru
- **Translatorul de cod –**Translatează codul ASM din forma mnemonică în forma algebrică
- **Managerul de biblioteci** –gestionează bibliotecile run-time
- **Biblotecile run-time** -conţin:
  - funcţii pentru lucru în timp real
  - funcţii pentru calcul aritmetic în virgulă flotantă
  - funcţii I/O suportate de standardul C - ANSI
- **Utilitarul de conversie hex** – converteste un fisier obiect de tip COFF intr-un fisier obiect de tip Ti-Tagged, ASCII-hex, Intel, Motorola-S sau Tektronix;
- Modulul **cross-reference lister** foloseste fisiere obiect pentru a produce un fisier de tip .LST ce va arata simbolurile, definitiile si referintele in fisierele sursa linkeditate.
- Modulul **absolute lister** – primeste la intrare fisiere obiect linkeditate si creaza fisiere de tip .ABS la iesire. Fisierul .ABS se asambleaza pentru a produce un fisier .LST ce va contine in locul adreselor relative, adrese absolute.

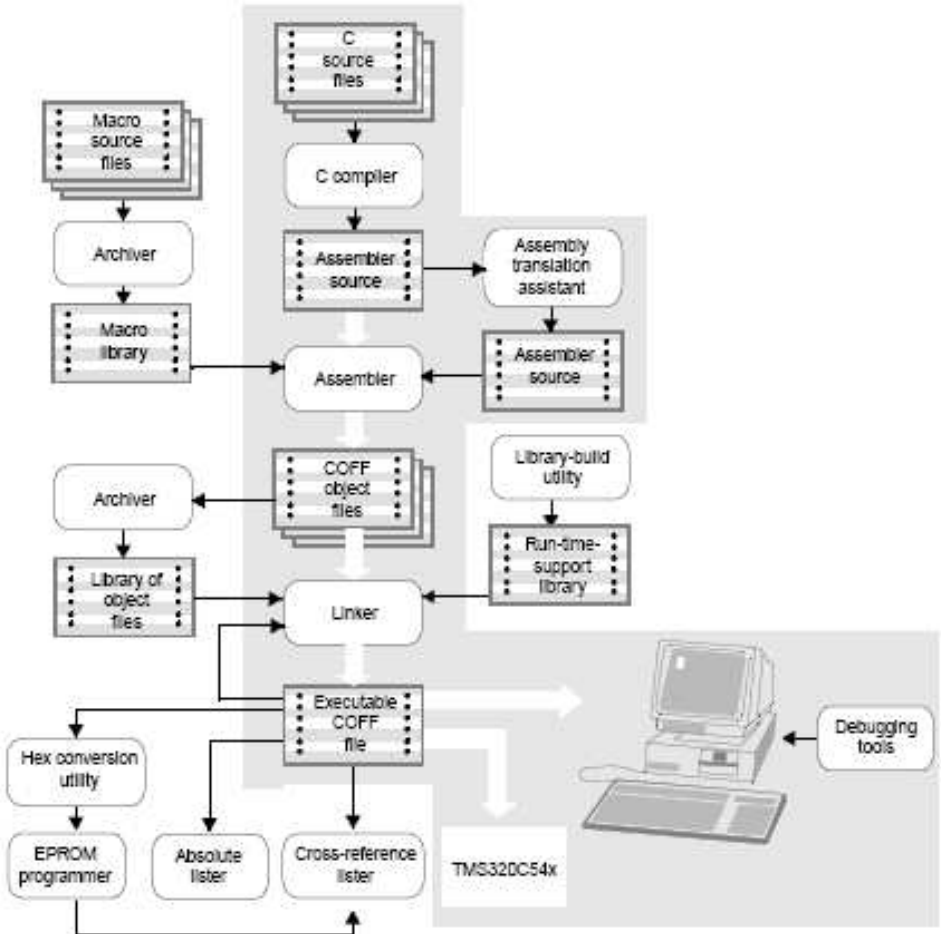


Fig 3 Fluxul de dezvoltare a aplicaţiilor sub Cod Composer Studio

3. Elemente necesare in dezvoltarea unei aplicaţii sub CCS

3.1. Utilizarea Probe-Point-urilor

Probe Point-urile sunt un instrument util pentru dezvoltarea algoritmilor. Pot fi utilizate pentru :

-a transfera date de intrare dintr-un fişier din PC-ul gazdă într-un buffer din DSP pentru a fi utilizate de algoritm

-a transfera date de ieşire dintr-un buffer din DSP într-un fişier din PC-ul gazdă pentru analiză

-a actualiza o fereastră (cum ar fi un grafic cu date). Probe Point-urile seamănă cu breakpoint-urile prin faptul că ambele întrerup programul ce se execută pe DSP pentru a-şi rula acţiunile proprii. Diferenţele dintre ele sunt: Probe Point-urile întrerup programul DSP momentan, fac o singură acţiune şi continuă execuţia programului. Breakpoint-urile întrerup DSP-ul până când execuţia este reluată manual şi fac ca toate ferestrele să fie reactualizate. Probe Point-urile permit intrarea şi iesirea automată în/din fişiere; breakpoint-urile nu permit acest lucru. Se foloseşte de asemenea un breakpoint pentru a actualiza toate ferestrele deschise când se ajunge la acel Probe Point. Aceste ferestre includ şi graficele datelor de intrare şi ieşire.

3.2. Utilizarea elementelor GEL

Code Composer Studio oferă ca modalitate de a modifica o variabilă utilizarea elementelor **"GEL" (**General Extension Language). "**GEL"** este o extensie de limbaj care permite crearea de mici ferestre utilizate la: modificarea valorilor variabilelor declarate în program, scrierea de funcţii care permit configurarea mediului IDE şi a accesului la procesor. Elementele GEL sunt descrise în fişiere cu extensia gel ce se adaugă în fereastra Proiect Wiew în secţiunea GEL files (Click-dreapta ->**Load GEL**)sau cu **File->Load GEL.**

Controlul GEL de tip cursor (slider)

Controlul GEL de tip cursor permite controlul unei variabile prin intermediul unui cursor.

```
slider param_definition( minVal, maxVal, increment, pageIncrement,
paramName)
{
   //corpul controlului
}
```

param_definition - este numele cursorului

minVal -este o valoare întreagă care specifică valaorea minimă a variabilei controlate de cursor

maxVal -este o valoare întreagă care specifică valaorea maximă a variabilei controlate de Slider

increment -este o valoare întreagă care specifică valaorea de increment a variabilei controlate de Slider

pageIncrement - este o valoare întreagă care specifică valoarea de increment a variabilei controlate de Slider atunci când se apasă tastele PageUp şi Page Down

paramName - este numele parametrului ce se utilizează în interiorul funcţiei

Controlul GEL de tip dialog

Acest tip de control permite modificarea unei variabilelor prin intermediul căsuţelor de dialog:

```
dialog funcName( paramName1 "param1 definition", paramName2 "param2
definition", ......)
{
    //corpul controlului
}
```

paramName[1-6] numele parametrilor utilizaţi în corpul funcţiei

"param1 definition" numele parametrilor ce apar tipăriţi în fereastra controlului. Se pot defini maxim 6 parametri.

In exemplul de mai jos se definesc două cotroale **GEL** unul de tip slider şi altul de tip dialog.

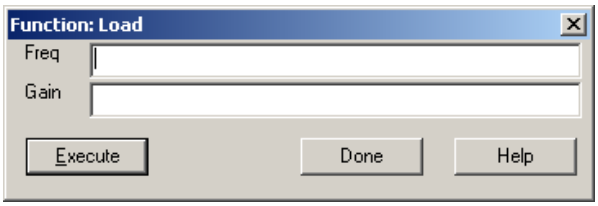
Exemplu de fişier GEL

menuitem "Application Control"

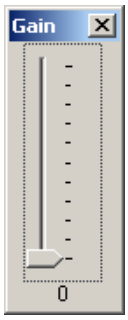
Application control este numele submeniului din meniul GEL care se activează după încărcarea fişierului **gel** .

```
dialog Load(loadParm1 "Freq",loadParm2 "gain")
{
       Freq = loadParm1;
       gain = loadParm2;
}
```

Se defineşte controlul **Gel** de tip dialog cu numele **Load** cu două variabile **loadParam1** şi **LoadParam2** ataşate variabilelor din program **Freq** respectiv **gain.** Variabilele se actualizează după apăsarea butonului Execute.




```
slider Gain(0, 10 ,1, 1, gainParm)
 {
    gain = gainParm;
 }
```

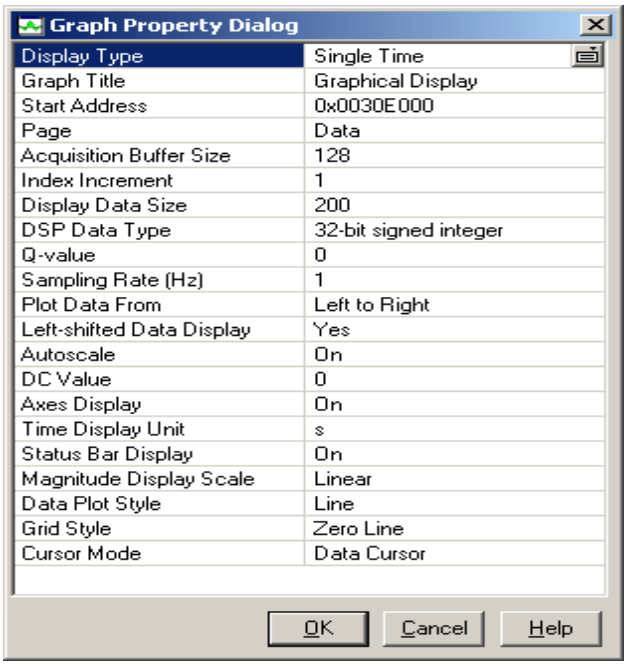


Se defineşte un control GEL de tip slider cu numele **Gain** ce poate modifica variabila gain între limitele: **0** şi **10** cu increment de **1** şi post increment de **1.**

3.3 Vizualizarea grafică a zonelor de memorie – "Freastra Graph"

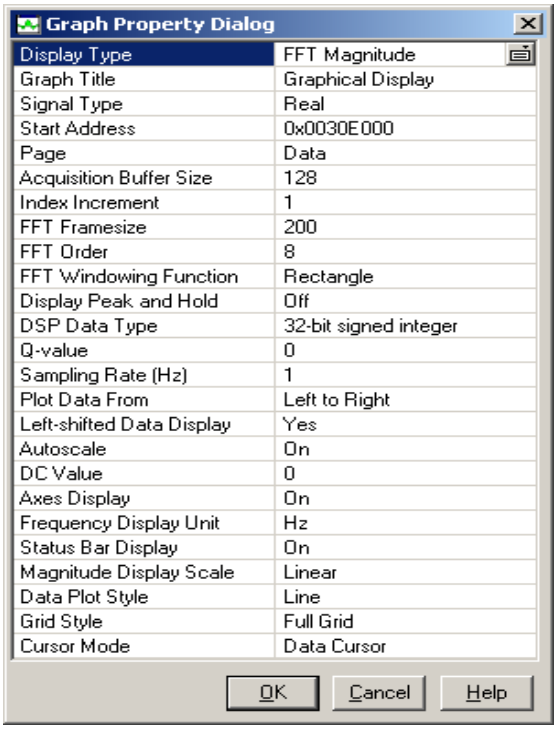
Prin utilizarea ferestrelor de tip Graph putem vizualiza zone de memorie sub formă grafică în domeniul amplitudine-timp sau dublu (dual time), FFT- amplitudine, FFT complex, FFT fază, FFT fază şi amplitudine.
Fereastra de configurare a unui Graph amplitudine-timp cu setările aferente se prezintă mai jos.

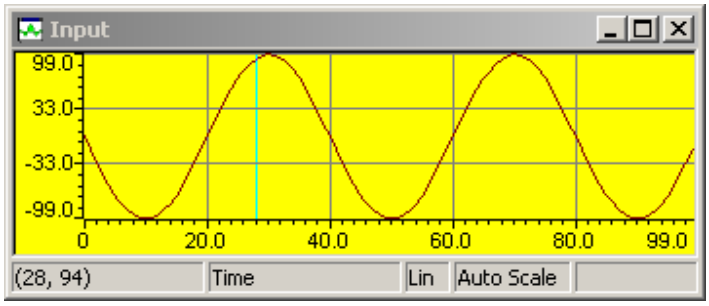
Display Type - domeniul în care se afişează
Graph title - titlul ce se afişează pe grafic
StartAddress - adresa de start
Page –pagina de memorie ce se afişează Program Data sau I/O
AcquisitionBufferSize–mărimea bufferului citit de către Graph
IndexIncrement –rezoluţia afişării
DisplayData Size – numărul de eşantioane afişate
DSP DataType –formatul de afişare a datelor
Q-value – tipul datelor afişate în format fracţionar
Left-shiftedData Display - tipărire cu deplasare spre dreapta
SamplingRate(Hz)-F_{es} în Hz pentru afişarea valorilor pe grafic
Plot data From –tipăreşte datele de la stânga la dreapta sau invers
Autoscale –autoscalare în amplitudine
DC Value-offset-ul
AxesDisplay-afişarea axelor
Time displayUnit - unitatea de masură pe axa timpului: s, ms, µs, eşantioane
StatusBar Display-afişarea barei de stare pe care se tipăresc cordonatele cursorului
Magnitude Display Scale - tipul scalei pentru amplitudine liniar sau logaritmic
Data Plot Style-Stilul de afişare Line sau Bar
Grid Style - Afişează rastrul (Full Grid), numai linia de zero (ZeroLine) sau nimic (No Grid)
Cursor Mode - Fără cursor (No Cursor), Cursor pentru date (Data Cursor), Cursor pentru Zoom (Zoom Cursor)

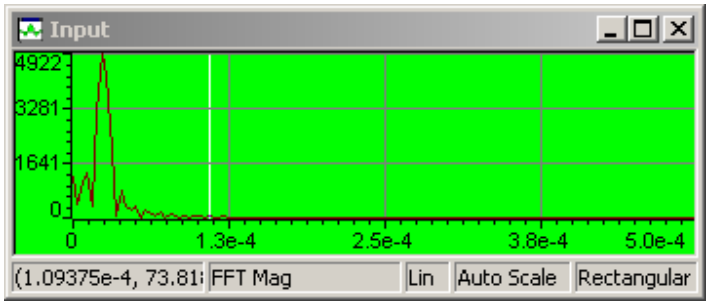


În cazul în care graficul este în domeniul frecvenţă, avem următorii parametri noi:

SignalType-tipul semnalului real sau complex
FFT-Frame size-mărimea ferestrei FFT
FFT Order-ordinul FFT
FFT Windowing Function –tipul ferestrei de ponderare
DisplayPeak and Hold – detectează şi păstrează maximele
Frequency Display Unit –unitatea de afişare pentru frecvenţă: Hz, KHz, MHz







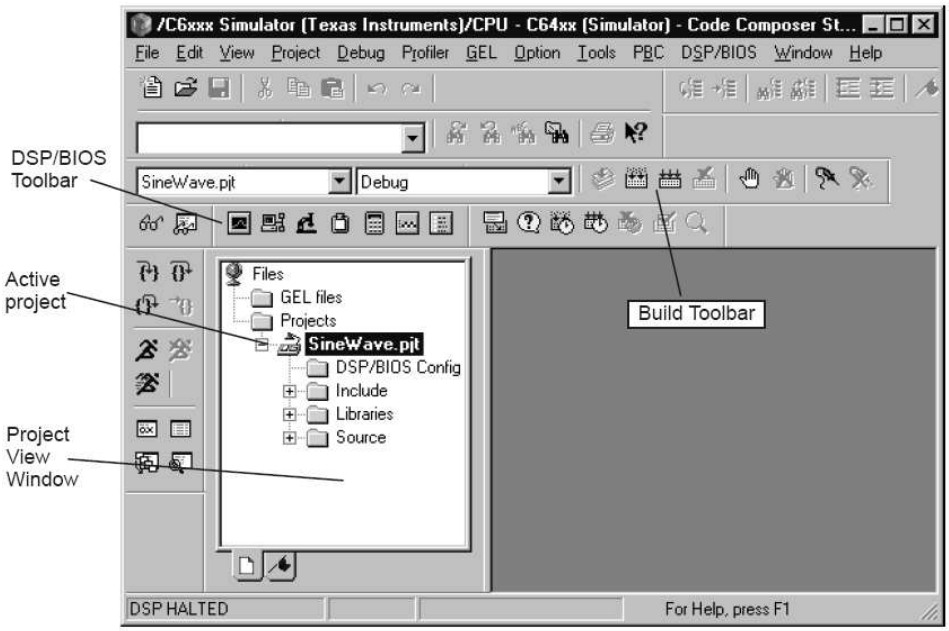
3.4. Exemplu de lucru cu CCS

Aplicaţia prezentată în continuare realizează controlul amplificării unui semnal de intrare.

Pentru a rula aplicaţia este necesar ca **CCS** să fie configurat astfel incat să lucreze cu simulatorul ***C5416 Simulator***. Această configurare se face prin intermediul componentei **Setup CCS.**

În continuare sunt descrişi paşii ce trebuie urmaţi pentru a dezvolta o aplicaţie în **CCS** şi pentru a o rula pe simulator.

1) Se alege meniul **Project -> Open** si se deschide proiectul *sinewave.pjt* din Turorial/sim54x/Sinewave. Proiectul contine fisierele *sine.h, sinewave.cmd, rts500.lib* precum si sursa principală a aplicaţiei *sine.c*.



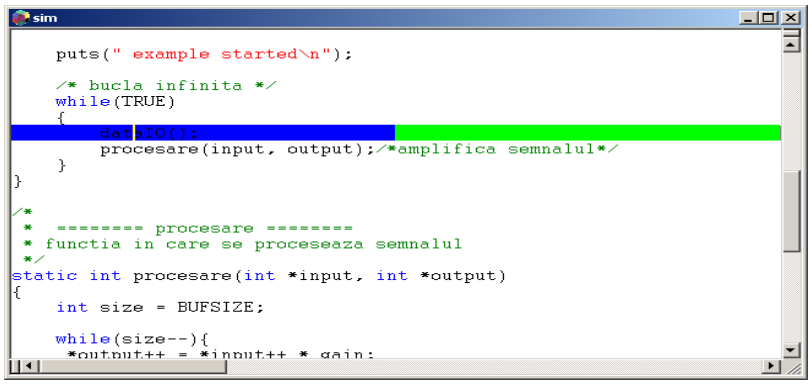
In continuare este prezentata sursa aplicatiei :

```
/*****************************************************************
/*sine.c      ; Acest program foloseste Probe Point-uri pentru a obtine un semnal sinusoidal de intrare, iar apoi acestui semnal i se aplica un factor de castig.
/*****************************************************************
#include <stdio.h>
#include "sine.h"

// variabila de control a castigului
int gain = INITIALGAIN;

// declarare si initializare Buffer IO
BufferContents currentBuffer;

// definire functii
static void processing();          // prelucreaza intrarea si genereaza semnalul de iesire
static void dataIO();              // functia ce se foloseste pentru ProbePoint

void main()
{
        puts("SineWave example started.\n");

   while(TRUE) // bucla continua
   {
      /*  citeste datele de intrare folosind un probe-point conectat la un fisier gazda
          Scrie datele de iesire intr-un graf conectat tot printr-un probe-point */
      dataIO();

      /* pentru a obtine datele de la iesire se aplica castigul asupra datelor de intrare */
      processing();
   }
}
/*Functia urmatoare aplica o transformare de semnal asupra semnalului de intrare pentru a genera semnalul de iesire si are ca parametrii: structura BufferContents ce contine sirurile de intrare/iesire de dimensiune BUFFSIZE; functia nu returneaza nici o valoare */

static void processing()
{
   int size = BUFFSIZE;

   while(size--){                  // aplica castigul asupra intrarii
      currentBuffer.output[size] = currentBuffer.input[size] * gain;
   }
}

/* Functia urmatoare citeste semnalul de intrare si scrie semnalul de iesire procesat folosind ProbePoints; functia nu are parametrii si nu returneaza nici o valoare. */

static void dataIO()
{
   return;
}
```

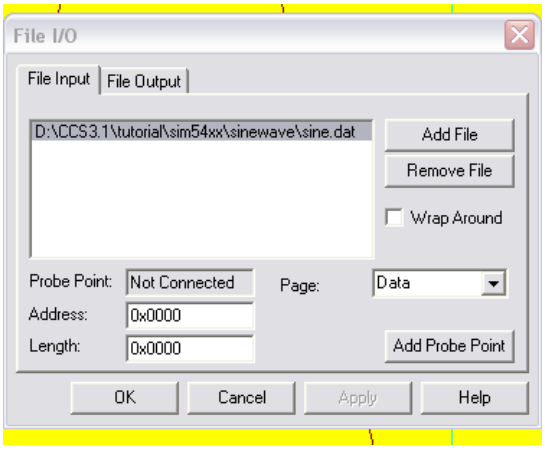
2) Compilaţi fişierele sursă cu **Project->Rebuild All** sau apăsati pe butonul (Rebuild All) din toolbar.
3) Incărcaţi fişierul obiect în memoria DSP din meniul **File->Load Program**. Selectaţi programul pe care tocmai l-aţi reconstruit, **sine.out** şi apăsaţi **Open**.
4) Daţi dublu-click pe fisierul **sine.c** din **Project View.**
5) Pentru a ataşa fisierul ce conţine semnalul ce urmează să fie procesat este necesar să înserăm un ProbePoint (care citeşte date dintr-un fişier de tip *.dat*).

In acest scop poziţionaţi cursorul pe rândul din funcţia *main* unde se apelează funcţia: ***DataIO( )***; Funcţia DataIO rezervă un loc unde se pot face adăugiri mai târziu. Pentru moment vom conecta un "**Probe Point"** care introduce date dintr-un fişier din PC.

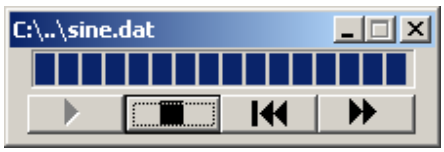
6) Apăsaţi pe butonul (**Toggle Probe Point**) din toolbar. Rândul e evidenţiat pe display.



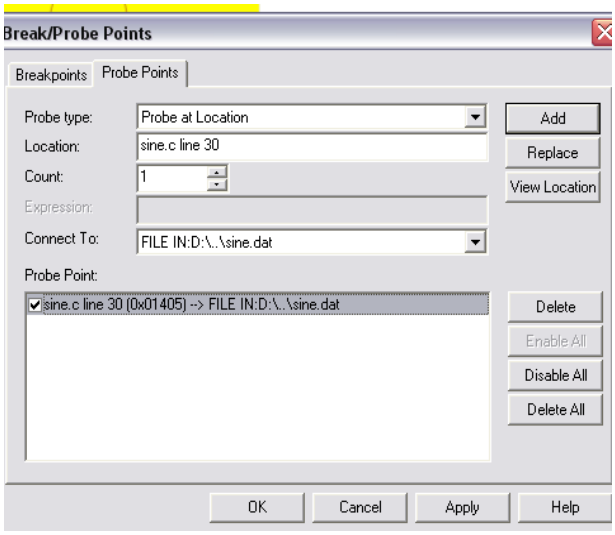
7) Alegeţi **File->File I/O**. Dialogul File I/O apare pentru a se putea selecta fişierele de intrare şi de ieşire.



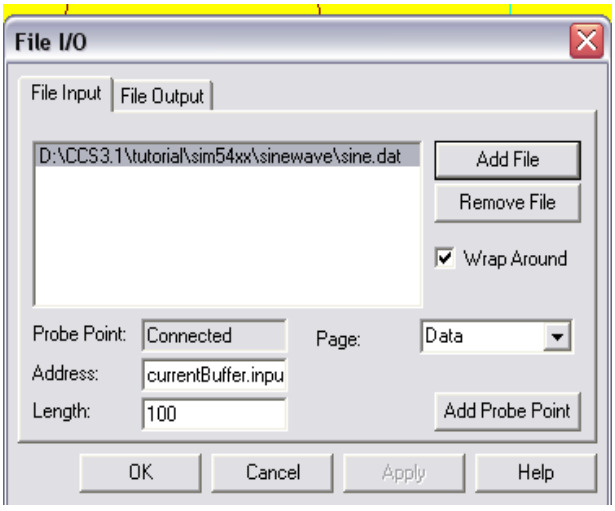
8) În meniul **File Input** apăsaţi pe **Add File**.
9) Alegeţi fişierul *sine.dat*. Observaţi că puteţi selecta formatul datelor în fereastra **Files of type**. Fişierul *sine.dat* conţine valori hexa pentru o formă de undă sinusoidală. Selectaţi **Open** pentru a adăuga acest fişier în lista de dialog **File I/O**. Va apărea o fereastră de control pentru fişierul *sine.dat* (poate fi acoperită de fereastra CCS). Mai târziu, când rulaţi programul, puteţi folosi această fereastră pentru comenzi ca: start, stop, rewind sau fast forward în cadrul fişierului de date.



10) In fereastra **File I/O** urmatorul pas este sa adaugam un **Probe Point** utilizand butonul cu acelasi nume. In fereastra de dialog **BreakProbe Points** validam probe point dand click pe el si ne va arata locatia si selectam **Connect to** alegand fisierul sine.dat. La sfarsit apasati Apply.



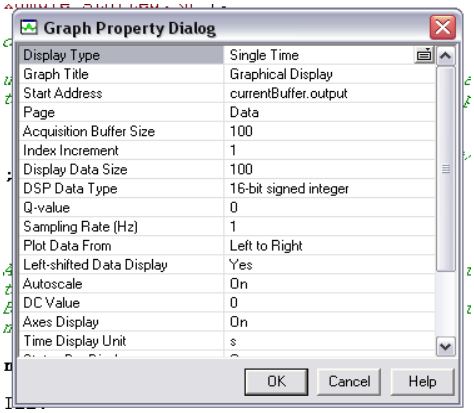
11) In dialogul **File I/O** setati adresa *currentBuffer.input* şi lungimea la 100. De asemenea, bifaţi **Wrap Around** (citirea se face circular). Câmpul de adresă (**Address**) specifică unde trebuie plasate datele din fisier. *currentBuffer.input* e declarat în volume.c ca integer array de valoarea BUFSIZE. Câmpul de lungime (**Length**) indică numărul de eşantioane din fişierul de date care sunt citite de fiecare dată când se ajunge la **Probe Point.** Folosim 100 pentru că aceasta este valoarea fixată pentru constanta BUFSIZE în *volume.h* (0x64). Opţiunea **Wrap Around** face ca CCS să înceapă să citească fişierul de la început după ce a ajuns la sfârşitul lui. Aceasta permite fişierului de date să fie tratat ca un flux continuu de date deşi conţine doar 1000 de valori şi la fiecare **Probe Point** sunt citite câte 100 de valori.



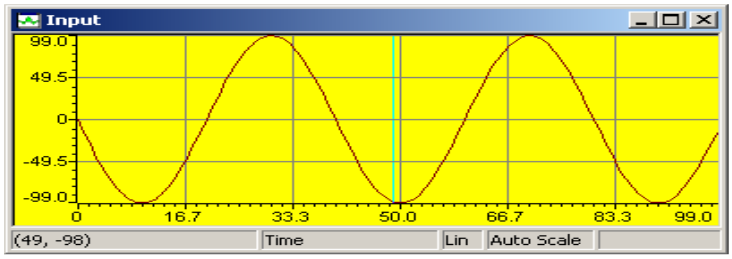
12) **Debug->Run**

13) Selectati **View->Graph->Time/Frequency.**

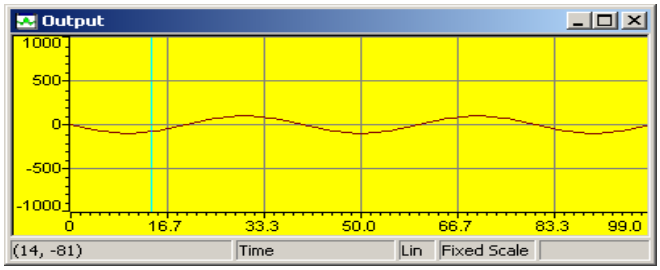
14) In **Graph Property Dialog** modificati caracteristicile: **Graph Title, Start Address, Acquisition Buffer Size, Display Data Size, DSP Data Type, Autoscale,** si **Maximum Y-value** cu valorile din figura.



15) Apăsaţi **OK**. Apare o fereastră grafică pentru bufferul de intrare.



16) Click- dreapta pe fereastra Input şi alegeţi **Clear Display** din meniul care apare.
17) Alegeţi din nou **View->Graph->Time/Frequency.**
18) De data aceasta modificaţi **Graph Title** în Output Buffer şi **Start Address** în *currentBuffer.output*, **Autoscale** OFF .**MaximumY-Value** la 1000.
19) Apăsaţi OK pentru a afişa fereastra grafică Output – care contine semnalul generat.



20) Daţi click-dreapta pe fereastra grafică şi alegeţi **Clear Display** din meniul afişat.
21) Alegeţi **File->Load Gel ->volume.gel**. Acest fisier contine un control GEL de tip slider si unul de tip dialog. Deschideţi fisierul pentru a-i vedea conţinutul.
22) Selectati **Gel-> Application Control -> Gain.**
23) În fereastra **Gain** utilizaţi reglajul pentru a modifica câştigul. În fereastra bufferului de ieşire se modifică amplitudinea. În plus, valoarea variabilei gain din fereastra **Watch** se modifică de câte ori mişcăm reglajul
24) Click (**Halt**) sau apăsaţi **Shift F5** pentru a opri programul.

5. Teme

a) Adăugaţi un control GEL de tip slider pentru a putea modifica castigul.
b) Scrieţi un program care generează un semnal sinusoidal. Vizualizaţi buffer-ul de ieşire prin intermediul unei ferestre Graph.
c) Înlocuiţi controlul cu cursor GEL cu un control Dialog.
d) Modificaţi programul astfel încât semnalul preluat din fişierul de intrare **sine.dat** să fie modulat în amplitudine cu semnal sinusoidal generat în program.
e) Modificaţi **Graph** astfel încât să afişeze semnalul rezultat în domeniul frecvenţă.