

APLICAȚII PE PLACA “SIDERAL” TMS320C25

1. Prezentarea plăcii acceleratoare SIDERAL - TMS320C25

Placa acceleratoare, "SIDERAL", folosită pentru experimentările de prelucrare a semnalului vocal este realizată pe baza procesorului de semnal TMS 320C25 (realizată la ITC Software Cluj-Napoca), având schema bloc prezentată în figura 1.

Din schemă se observă că elementul central al schemei este memoria cu dublu acces: din partea DSP-ului și a calculatorului gazdă, IBM-PC. Memoria este împărțită în două secțiuni: memorie de program și memorie de date, conform arhitecturii Harvard a procesorului de semnal. Oscilatorul este realizat extern pentru a putea modifica ușor caracteristicile sale. Pentru realizarea multiplexorului de date/adrese în cadrul memoriei biport s-au folosit buffere cu 3 stări (18286).

Cu ajutorul circuitului generator de stări wait se pot introduce între 0-2 stări de wait la accesarea memoriei. Dintre cele 64 Kcuvinte pentru memoria program și 64Kcuvinte pentru memoria de date s-au implementat doar câte 32 Kcuvinte, suficient pentru multe dintre aplicațiile posibile.

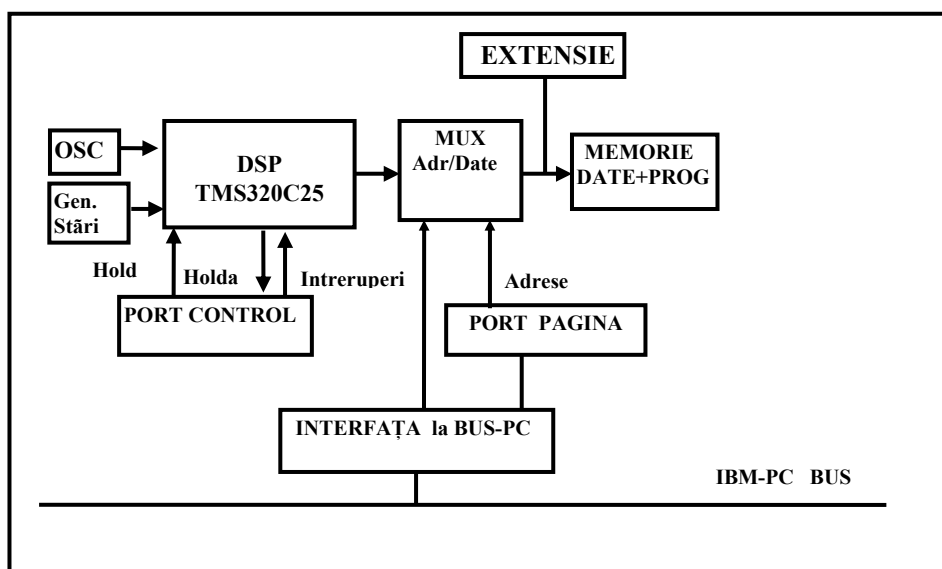


Fig.1. Schema bloc a plăcii acceleratoare cu TMS 320C25

Acest spațiu este paginat folosind adresele superioare, fiind văzut în “felii” de 8 Kocteți de date/program de către PC. Portul de paginare este “văzut” la adresa 311H de către PC.

Multiplexorul de adrese/date către PC și DSP este realizat cu buffere “three-state”, a căror activare este făcută cu semnalul BHOLD livrat de DSP în urma unei cereri de magistrală prin HOLD. Din punctul de vedere al calculatorului gazdă decodificarea memoriei este făcută între A0000H-EFFFFH, rezervată în orice PC pentru memoria video și extensia BIOS. Se recomandă folosirea zonei D0000H-EFFFFH pentru “fereastra” de 8 Kocteți în acord cu adresa fixată de managementul de memorie.

Porturile folosite pentru paginare/comandă sunt plasate în zona 300H-311H. Interruperile DSP către PC se aplică la alegere pe unul dintre nivelele IRQ4-IRQ9 și se pot invalida de către PC punând ENIRQ=0 (pe portul de control). Placa mai conține un conector DIN 2X32 pini pe care s-a realizat extensia pentru achiziția de semnal.

Placa de achiziție semnal facilitează achiziția semnalelor de joasă frecvență în spectrul audio (15 Hz-15kHz). Pe canalul analogic de intrare avem un convertor analog-numeric de 12 biți (MMC 757), cu rata de conversie maximă de 25 kHz, pentru semnale în gama de $\pm 3V$. Citirea celor 12 biți de către procesor se face astfel: biții 11-4 se citesc pe partea low a portului cu adresa 0, iar biții 3-0 pe partea low a portului 1. (Porturile la TMS320C25 sunt pe 16 biți!).

Datele citite sunt în cod complementar (valoarea maximă este 000, iar cea minimă FFFh). Startul unei noi conversii se face printr-o instrucțiune OUT pe portul 1. Pentru ieșirea analogică se folosește un convertor numeric-analogic (DAC 08) cu rezoluția de 8 biți, urmat de un convertor curent-tensiune de ieșire care poate genera un semnal bipolar în gama $10V_{vv}$. Datele se scot pe convertor, pe partea low a portului de ieșire cu adresa 0.

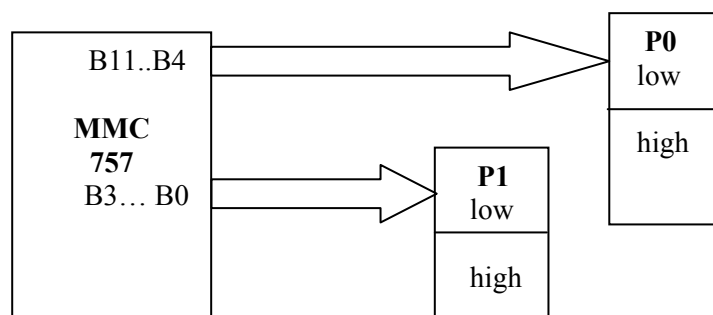


Fig.2 Conectarea convertorului MMC757 la porturi

2. Prezentarea soft-ului pentru dezvoltarea de aplicații

2.1. Asamblorul ASC25

Lansarea asamblorului se face cu comanda:

asmc25 <fișier sursă> [<fișier obiect>] [<fișier listing>]

unde:

- <fișier sursă> este numele fișierului de tip text care conține programul pentru TMS320C25 scris în limbajul de asamblare al procesorului;
- <fișier obiect> este numele fișierului care va conține codul obiect rezultat în urma asamblării programului din fișierul sursă;
- <fișier listing> este numele fișierului de tip text care va conține listingul programului din fișierul sursă.

Funcțiile asamblorului sunt:

- analizarea sintactică a liniilor din fișierul sursă;
- afișarea pe ecran a erorilor detectate în cursul asamblării și numărul liniilor din fișierul sursă în care acestea au apărut;
- generarea codului obiect și depunerea lui în fișierul obiect, dacă este specificat numele acestuia în linia de comandă;
- generarea listingului și depunerea lui în fișierul listing, dacă este specificat numele acestuia în linia de comandă;

Definiții

Identificator este un șir de caractere care începe cu o literă și poate avea până la 32 de caractere; el desemnează etichete sau constante simbolice.

Constantele numerice pot fi:

- binare* când sunt reprezentate printr-un șir de cifre binare terminat cu litera 'b' (de ex.: 011101b);
- zecimale* când sunt reprezentate de un șir de cifre zecimale (de ex. 27);
- hexazecimale* când sunt reprezentate de un șir de cifre hexazecimale terminat cu litera 'h' (ex. 01bh);
- pozitive, caz în care se scrie valoarea numerică absolută a constantei fără nici un semn;
- negative caz în care se scrie valoarea numerică absolută a constantei precedată de semnul '-'.

Expresie este orice combinație de constante numerice și simbolice legate prin operatorii +, -, * și /. Parantezele pot modifica precedența operațiilor.

Comentariu este orice șir de caractere cuprins între simbolul ';' și sfârșitul liniei.

Directive de asamblare

Title

sintaxa: **title** '<șir de caractere>'

descriere: <șir de caractere> este considerat titlul programului și va apărea în antetul listingului;

Equ

sintaxa: <constanta simbolică> **equ** <expresie>

descriere: se evaluează expresia și se atribuie constantei simbolice valoarea ei;

Data

sintaxa: [<etichetă>] **data** <expresie>

descriere: se evaluează expresia și valoarea ei și se utilizează pentru inițializarea locației de memorie corespunzătoare valorii curente a contorului de program, după care contorul de program se incrementează cu 1.

Aorg

sintaxa: [<etichetă>] **aorg** <expresie>

descriere: se evaluează expresia și valoarea ei se atribuie contorului de program, astfel că instrucțiunea imediat următoare va fi asamblată la adresa dată de valoarea expresiei.

End

sintaxa: **End**

descriere: marchează opțional sfârșitul programului sursă.

Phase

sintaxa: [*<eticheta>*] **phase** *<expresie>*
<instrucțiuni >

dephase

descriere: se evaluează expresia și valoarea ei se atribuie contorului de program, astfel că instrucțiunile ce urmează până la **dephase** vor fi asamblate pentru adresa dată de valoarea expresiei; adresa de depunere a codului obținut nu este modificată.

2.2. Programul încărcător LOADRUN

Programul permite încărcarea aplicațiilor .obj rezultate în urma asamblării în memoria plăcii acceleratoare și lansarea în execuție.

Se lansează LOADRUN, iar apoi se cere introducerea numelui fișierului obiect dorit care se va încărca și apoi se va lansa în execuție.

Intreruperea rulării se face prin apăsarea unei taste.

3. Aplicații

3.1. Repetor de semnal

În cele ce urmează se prezintă fișierul listing al aplicației “REPETOR” care preia semnalul de pe intrarea analogică și îl transmite ieșirii analogice.

```

Linie  Adr.  Cod
Sursă  mem. mașină;
0001
0002          ;
0003          ;
0004      4e20  fc      equ      20000
0005      0019  fe      equ      20
0006      00c7  PRD     equ      fc/(4*fe)-1
0007 0000 c800 begin  ldpk 0      ;DP=0
0008 0001 d001      lalk PRD ;ACClow =PRD
      0002 00c7

0009 0003 6003      sac1 3      ;reg. PRD = ACClow (vezi mem.Date)
0010 0004 ca08      lack 8      ;ACClow = masca intr. ptr. timer
0011 0005 4d04      or 4        ;ACClow = ACClow U IMR
0012 0006 6004      sac1 4      ;IMR = ACClow, TINT = 1
0013 0007 ce00      eint        ;validez intreruperile
0014 0008 ff80 stai  b stai     ;astept intrerupere
0015 ;****Rutina de intrerupere TIMER ****
0016 0018          aorg 24      ;.space 16*(24-7);
0017
0018 807f          in 7fh,PA0    ;citeste portul analogic
      ;partea high(b11-b4)
0018 0019 207f      lac 7fh     ;ACClow=adr.7Fh)mem.date
0019 001a ce27      cmpl        ;ACClow = not (ACClow)
0020 001b 607f      sac1 7fh     ;(adr.7Fh)mem.date = ACClow
0021 001c e07f      out 7fh,PA0 ;trimite esantionul citit la P0
0022 001d e170      out 70h,PA1 ;initiez noua achizitie
0023 001e ce00      eint        ;valideaza intreruperi
0024 001f ce26      ret

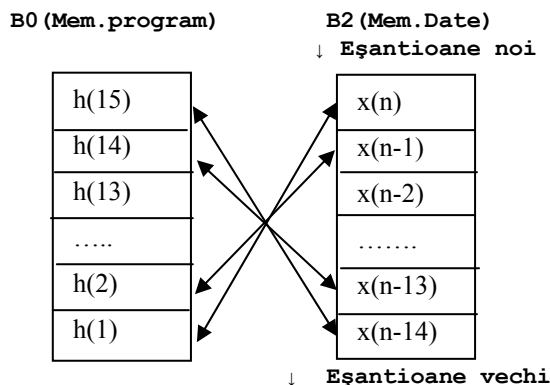
```

3.2. Filtru nerecursiv (FTJ)

Aplicația FTJ.asm implementează un filtru FIR trece jos cu 15 coeficienți cu frecvența de tăiere de $0.05 \cdot F_e$. Implementarea se bazează pe relația:

$$y(n) = \sum_{k=0}^{N-1} h(k+1)x(n-k)$$

unde $N=15$, $h(k)$ sunt coeficienții filtrului, iar $x(n)$ eșantioanele de intrare. Coeficienții s-au calculat cu un program separat și se păstrează în blocul B0 (plasat în memoria program : FF01h...FF0Fh), iar eșantioanele în blocul B2 (în memoria de date 70h...7Eh) pentru a avea acces în același ciclu la ambii operanzi ai înmulțirii.



```

0001          ;FTJ FIR cu 15 coef. Ftaiere = 0.05*Fe
0002          ;
0003          ;
0004      4e20 fc      equ      20000
0005      000f fe      equ      10
0006      014c PRD      equ      fc/(4*fe)-1

0007 0000 c800 begin    ldpk 0
0008 0001 d001          lalk PRD
0009          0002 014c
0009 0003 6003          sac1 3
0010 0004 ca08          lack 8
0011 0005 4d04          or 4
0012 0006 6004          sac1 4
0013 0007 ff80          b init
0014          0008 003e

0014          ;.space 16*24
0015 0018          aorg 24
0016          ; intrerupere TIMER
0017 0018 8070          in 70h,PA0;citesc eșantion nou de la PA0
0018 0019 2870          lac 70h,8 ;Acc=(PA0)* 2^8
0019 001a d006          xork 8000h;scalare valori de la CAN (B11...B4)
0019          001b 8000
0020 001c 6070          sac1 70h
0021 ;reset ACC si P
0022 001d a000          mpyk 0 ;P=0
0023 001e ca00          zac ;Acc=0
0024          ;filtrarea propriu-zisa
0025 001f 5588          larp AR0 ;ARP=0
0026 0020 d000          lrlk AR0,07eh ;AR0=7Eh
0026          0021 007e
0027 0022 cb0e          rptk 14 ;repet de 15 ori
0028 0023 5c90          macd 0ff01h,*- ;vezi ce face MACD!
0028          0024 ff01
0029 0025 ce15          apac ;Acc=Acc+P
0030 0026 6869          sach 69h ;(69h)= AccH
0031 0027 2869          lac 69h,8 ;AccL=(69h)* 2^8
0032 0028 d806          xork 8000h,8 ;Acc=(Acc)⊕ 800000h
0032          0029 8000
0033 002a 6869          sach 69h ;(69h)=AccH

```

```

0034 002b e069      out 69h,PA0      ;PA0=(69h)
0035 002c e169      out 69h,PA1      ;inițiez o nouă conversie
0036 002d ce00      eint
0037 002e ce26      ret
0038 002f fed9      COEF  data 0fed9h ;h15
0039 0030 0102      data 102h   ;h14
0040 0031 044a      data 44ah   ;h13
0041 0032 085d      data 85dh   ;h12
0042 0033 0ca9      data 0ca9h  ;h11
0043 0034 1075      data 1075h  ;h10
0044 0035 1312      data 1312h  ;h9
0045 0036 1400      data 1400h  ;h8
0046 0037 1312      data 1312h  ;h7
0047 0038 1075      data 1075h  ;h6
0048 0039 0ca9      data 0ca9h  ;h5
0049 003a 085d      data 85dh   ;h4
0050 003b 044a      data 44ah   ;h3
0051 003c 0102      data 102h   ;h2
0052 003d fed9      data 0fed9h ;h1
0053 003e ce09      init spm 1   ;PM=1
0054 003f ce04      cnfd          ;B0 in memoria date
0055 0040 5588      larp 0        ;ARP=0
0056 0041 d000      lrlk AR0,0201h ;AR0=201h
0042 0201
0057 0043 cb0e      rptk 14       ;mută coeficienții în
0058 0044 fca0      blkp COEF,*+   ;B0 de la 201h...20fh
0045 002f
0059 0046 ce05      cnfp          ;flag CNF=1,B0 memorie program
0060 0047 ce00      eint          ;validare intreruperi
0061 0048 ff80      stai  b stai    ;astept intreruperi de la timer

```

4. Teme

- Studiați cele două aplicații și observați modul de utilizare al resurselor procesorului TMS320C25 și a plăcii SIDERAL.
- Asamblați și executați aplicațiile vizualizând semnalele la intrare și ieșire.
- Scrieți un program care să genereze un semnal dreptunghiular cu frecvența de 10KHz și amplitudinea de $\pm 3V$.
- Scrieți un program care să genereze un semnal sinusoidal cu frecvența de 2 kHz și amplitudine $\pm 5V$ folosind metoda tabelară sau recursivă.
- Pentru un filtru FIR trece sus cu frecvența de tăiere egală cu $0.1 \cdot F_e$ se dau coeficienții de mai jos: $h1=h15=8.99696E-3$; $h2=h14=-7.86406E-3$; $h3=h13=-3.34992E-2$; $h4=h12=-6.53486E-2$; $h5=h11=-9.8897E-2$; $h6=h10=-0.12856$; $h7=h9=-0.14897$; $h8=0.84375$.

Să se treacă în format Q15 și să se modifice în programul FTJ.asm generând un nou program FTS.asm care se va executa vizualizînd semnalele de intrare și ieșire.

