Arhitectura microprocesoarelor 2. Microcontrolere

Curs 13

AMP2. MC - 13

Exemplu de problema - DSP

AMP2. MC - 13

Sa se scrie , in limbaj de asamblare ADSP2181, codul pentru generarea si recepția semnalelor analogice utilizate in transmiterea si recepționarea informațiilor intre doua echipamente folosind un modem care asociază unui bit de “1” un semnal sinusoidal de frecventă 400 Hz si unui bit de “0” un semnal sinusoidal de frecventa 200 Hz . Durata semnalelor sinusoidale este de 10 ms. Recepția semnalelor se va face prin numărarea trecerilor prin zero. Se consideră problema sincronizării modemului ca fiind rezolvata.

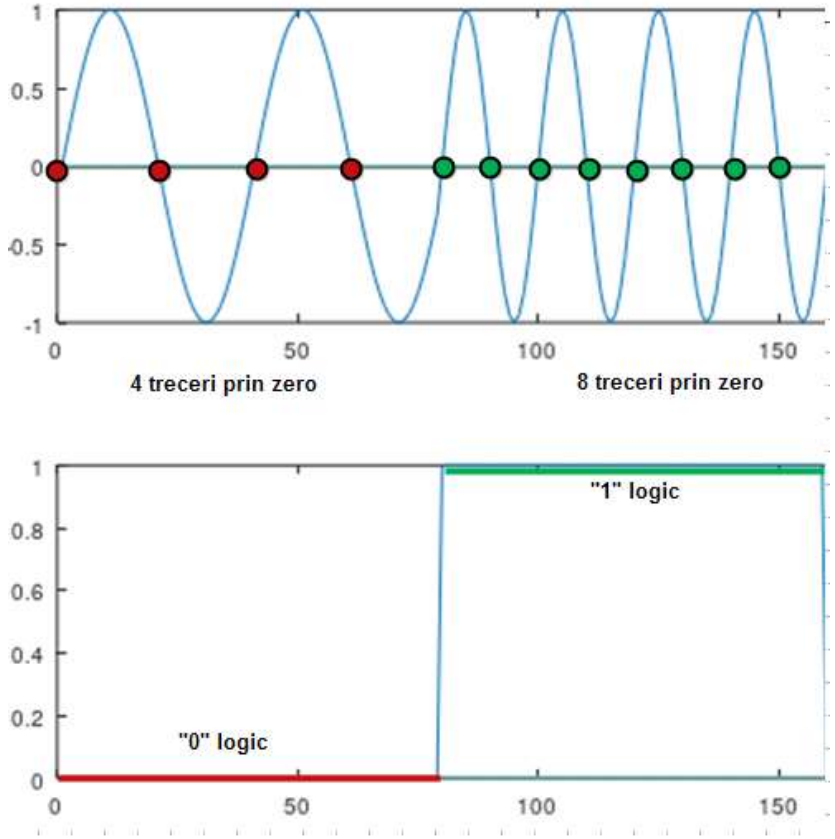
Se cer:

Explicații asupra realizării codului (algoritm, structuri de date utilizate)

Codul in limbaj de asamblare ADSP2181 pentru funcțiile de generare se semnalului analogic si pentru recepția acestuia.

AMP2. MC - 13

Solutie propusa



AMP2. MC - 13

Solutie propusa

Se considera frecventa de esantionare este 8000 Hz.

Transmisie:

- pentru fiecare semnal - trebuie memorata o perioada de 10 ms

Se definesc 2 variabile (buffere circulare) cu dimensiunea 80 de esantioane:

Bufferul y0 – bitul “0” – semnal cu frecventa 200 Hz (2 perioade)

Bufferul y1 – bitul “1” – semnal cu frecventa 400 Hz (4 perioade)

Receptie:

- se numara trecerile prin zero pe durata de 10 ms
- perioada a semnalului sint 2 treceri prin zero
- pentru bitul “0” (200 Hz) - perioada este de 5 ms

  10 ms / 5 ms = doua perioade de semnal si 4 treceri prin zero.

- pentru semnalul de 400Hz - 8 treceri prin zero.

AMP2. MC - 13

Cod posibil

```
l0=80;      // lungimea bufferelor pentru semnalele sinusoidale
m0=1;       // modificatory pentru ambele buffere
l1=0;       // bufferul de iesire nu e circular

ar=0;       // generare semnal pentru bit 0
call gen;   // gen – genereaza semnal sinusoidal la iesire
call rec;   // receptie semnal

ar=1;       // generare semnal pentru bit 1
call gen;   // gen – genereaza semnal sinusoidal la iesire
call rec;   // receptie semnal
```

AMP2. MC - 13

```
 gen:
i1=out;              // i1 index pentru bufferul de iesire
cntr=80;
ar=ar-1;             // ar - bitul
if eq jump s1;
s0:  i0=y0;          // I0 index pentru sinus asociat bitului “0”
jump generate;
s1: i0=y1;           // I0 index pentru sinus asociat bitului “1”
generate:
do gen_sgn until ce;
ax0=dm(i0,m0);   // citeste esantion din y0 sau y1
gen_sgn:
dm(i1,m0)=ax0;   // scrie esantion la iesire
rts;
```

AMP2. MC - 13

```
rec:
i1=out; l1=0; cntr=80;
ax0=0;                  // contor treceri prin zero
ay0=1;                  // increment
ay1=0;                  // valoarea de comparat (zero)
do rec_sgn until ce;
ax1=dm(i1,m0);          // citeste esantionul curent
ar=ax1-ay1;             // compara cu 0
if eq jump count;
jump rec_sgn;
count:
ar=ax0+ay0;             // incrementeaza contor treceri prin zero
rec_sgn: ax0=ar;        // actualizeaza contor treceri prin zero
```

AMP2. MC - 13

// decodeaza bitul

// dupa numarul de treceri prin zero pe durata specificata

```
ay1=4;
ar=ax0-ay1;
if eq jump bit0;          // testeaza valoare contor treceri prin zero
ay1=8;
ar=ax0-ay1;
if eq jump bit1;

ar=0xFFFF;
dm(bit)=ar;               // in caz de eroare bit = 0xFFFF
rts;
```

AMP2. MC - 13

// varaiabila bit - bitul decodat

```
bit0:
ar=0x0000;
dm(bit)=ar;          // bit va contine valoarea bitului
rts;

bit1:
ar=0x0001;
dm(bit)=ar;
rts;
```

Next

Exemplu de problema - MCU (examen)