Anexa Calculul codurilor ciclice

Asa cum s-a precizat in capitolul 4, functiile circuitului de calcul al secventei de control pot fi realizate prin program.

```
//
// fisier crc.h
//
// contine directivele si definitiile utile programelor
// de calcul al codului de control ciclic CRC-CCITT
//

#define CRCCCITT 0x1021
#define NIL 0

#include <stdio.h>
#include <stdlib.h>

typedef unsigned int word;
typedef unsigned char byte;

word calculcrc (word, word, word);
word* tabelcrc (word);

#include "crc.h"

//
// calculeaza noul rest partial pe baza vechiului rest
// (acum), a unui octet de date (data) si a polinomului
// generator (genpoli)
//

word calculcrc (word data, word genpoli, word acum)
{
  int i;
  data <<= 8;
  for (i=8; i>0; i--)
     {
     if ((data^acum) & 0x8000)
       acum = (acum << 1) ^ genpoli;
     else
       acum <<= 1;
     data <<= 1;
     }
  return acum;
}
```

Functia "calculcrc" determina efectul aplicarii procedeului clasic de calcul CRC asupra unui octet de date. Generatorul polinomial (genpoly) si continutul initial al acumulatorului (accum) sint date ca argumente.

Calculul prin program al secventei de control sugereaza posibilitatea de a trata problema nu la nivelul fiecarui bit al unui mesaj, ci la nivelul octetilor. Ideea este de a gasi o modalitate mai simpla prin care, dat fiind un octet de date si continutul vechi al acumulatorului (registrul de deplasare) sa se calculeze noua valoare a acumulatorului. Facind o simulare a operatiilor, se poate constata ca octetul de date se combina doar cu bitii mai semnificativi ai acumulatorului, noua valoare a acumulatorului fiind suma (modulo 2 a) secventei de control corespunzatoare acestei valori combinate si a octetului mai putin semnificativ al acumulatorului. Deoarece valoarea combinata ocupa 8 biti, ea poate avea 256 de valori diferite, secventele de control pentru toate aceste valori putind fi calculate si pastrate intr-un tablou. In acest mod calculul noului acumulator se poate face foarte simplu, prin insumarea modulo 2 a octetului inferior al acumulatorului cu o valoare precalculata, din cea totala.

Functia tabelcrc construieste acest tablou; ea are ca parametri: generatorul polinomial si un pointer la o functie CRC (de ex. calculcrc) si intoarce (unsigned short *) un pointer in tabela.

```
//
// calculeaza tabelul codurilor CRC
//

//
// alcatuieste tabelul codurilor CRC pentru un anumit
// polinom generator (poli); apeleaza "calculcrc" pentru
// toate cele 256 combinatii posibile ale octetului de date
// intoarce un pointer la tabelul de coduri, alocat dinamic
//

word* tabelcrc (word poli )
{
  word* ptabelcrc;
  int i;
  if ((ptabelcrc = (word*) malloc (256 * sizeof (word))) == NIL)
     return NIL;
  for (i=0; i<256; i++)
```

```
    ptabelcrc [i] = calculcrc (i, poli, 0);
  return ptabelcrc;
}
```

Tabelul CRC rezultat in urma executiei acestui program este urmatorul:

0x 0, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
0x1231, 0x 210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
0x2462, 0x3443, 0x 420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
0x3653, 0x2672, 0x1611, 0x 630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
0x48c4, 0x58e5, 0x6886, 0x78a7, 0x 840, 0x1861, 0x2802, 0x3823,
0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x a50, 0x3a33, 0x2a12,
0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x c60, 0x1c41,
0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x e70,
0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
0x1080, 0x a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
0x 2b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
0x34e2, 0x24c3, 0x14a0, 0x 481, 0x7466, 0x6447, 0x5424, 0x4405,
0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
0x26d3, 0x36f2, 0x 691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x 8e1, 0x3882, 0x28a3,
0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x af1, 0x1ad0, 0x2ab3, 0x3a92,
0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x cc1,
0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x ed1, 0x1ef0.

Functia de calcul CRC cu ajutorul tabelului este urmatoarea :

```
#include "crc.h"
#include <io.h>

//
```

```
// calculeaza CRC partial (acum) corespunzator unui octet
// (data) prin utilizarea tabelului codurilor CRC (tabelcrc)
//

void crctabel (word data, word* acum, word* tabelcrc)
  {
  word valcomb;
  valcomb = ((*acum >> 8) ^ data) & 0x00FF;
  *acum = ((*acum & 0x00FF) << 8) ^ tabelcrc [valcomb];
}

//
// calculeaza CRC pentru un fisier al carui nume
// este dat de utilizator; pentru verificare, dupa
// calculul CRC corespunzator fisierului, se continua
// calculul considerind si valoarea CRC gasita;
// rezultatul (acumulator) dupa adaugarea CRC este nul
//
void main() {
...
}
```