CRC 16 USB

O CRC USB tem por objetivo aumentar a robustez do envio de dados, em específico na transmissão de dados por USB. Ele funciona de forma a gerar um valor (16 bit no caso) a partir de determinado dado, valor esse que pode ser verificado no receptor.

Seu funcionamento se dá da seguinte forma:

**Calculo CRC**

1. Utilizasse de uma variável **crc** que conterá o resultado do cálculo. A mesma é iniciada com 0xffff;
2. Utiliza-se de uma variável **polGerador** que contém os 16 bits LSB do polinômio gerador x^16 x^15 x^2 x^1 ou 11000000000000101. Logo **polGerador** = 1000000000000101;
3. O dado recebido é revertido byte a byte;
4. Se o MSB do **crc** é igual ao MSB do dado:
   1. Realiza-se o shitf do **crc** à esquerda.
5. Se não:
   1. Realiza-se o shitf do **crc** à esquerda;
   2. Faz-se o **crc** receber a operação xor entre o **crc** e o **polGerador.**
6. Verifica-se se já se encontra no fim do dado. Se não, volta para o passo 4.
7. Inverte-se os bits do **crc**, e encerra-se o programa.

**Exemplo**

A sequência 00000000000000010000001000000011 é tratada da seguinte forma:

1. Utilizasse de uma variável **crc** que conterá o resultado do cálculo. A mesma é iniciada com 0xffff;
2. Utiliza-se de uma variável **polGerador** que contém os 16 bits LSB do polinômio gerador x^16 x^15 x^2 x^1 ou 11000000000000101. Logo **polGerador** = 1000000000000101;
3. O dado recebido é revertido byte a byte gerando 00000000100000000100000011000000;
4. A variável **crc** tem seu MSB em 1 e o dado tem seu MSB em 0 portanto:
   1. Realiza-se o shitf do **crc** à esquerda – **crc** recebe 1111111111111110;
   2. Faz-se a operação de xor entre o **crc** e o **polGerador** gerando um **crc** igual à 0111111111111011**.**
5. A variável **crc** tem seu MSB em 0 e o dado tem seu MSB em 0 portanto:
   1. Realiza-se o shitf do **crc** à esquerda – **crc** recebe 1111111111110110;

...

1. (Trigésimo segundo byte) A variável **crc** tem o seu MSB em 0 e o dado tem o seu MSB em 1:
   1. Realiza-se o shitf do **crc** à esquerda – **crc** recebe 1000100010100100;
   2. Faz-se a operação de xor entre o **crc** e o **polGerador** gerando um **crc** igual à 0000100010100001**.**
2. Encontra-se o fim dos dados;
3. Invertesse o **crc** bit a bit gerando 1111011101011110;
4. Encerra-se o programa.

**Verificação**

1. Realiza-se o cálculo do crc da mesma forma que descrito no **Cálculo CRC**.
2. Compara-se o resultado com o encontrado no arquivo.
3. Imprime o status do arquivo.
4. Encerra o programa.

O algoritmo funciona exatamente como descrito no artigo dado como fonte bibliográfica.