

Procedimento para cálculo do CRC

No modo RTU, é incluído na mensagem um error-checking baseado no método CRC que verifica se a mensagem recebida está correta.

O CRC contém dois bytes e é calculado pelo dispositivo transmissor, que anexa o CRC na mensagem.

O dispositivo receptor recalcula o CRC após a recepção da mensagem e compara o valor calculado com o valor recebido. Se os valores não são iguais, a mensagem é descartada.

O algoritmo para cálculo do CRC é:

1. Preencha um registro de 16 bits com 1s (0xFFFF)
2. Faça um OR EXCLUSIVE entre o registro (lsb) e o byte de transmissão
3. Desloque o registro obtido 1 bit à direita
4. Se o bit menos significativo do registro for igual a 1, faça um OR EXCLUSIVE com os seguintes

16 bits:

10100000 00000001

MSB LSB

5. Repita os passos 3 e 4 oito vezes
6. Repita os passos 2,3,4 e 5 para todos os bytes da mensagem
7. O conteúdo final do registro é o valor do CRC que é transmitido no final da mensagem começando com o byte menos significativo.

A seguir uma função para cálculo do CRC escrita em C:

```
void main()
{
    unsigned char buf_485[30];
    union CRC
    { unsigned char c_crc[2];
      unsigned int i_crc;
    } crc;
    crc.i_crc = 0;
    buf_485[0] = 0x1B; // Endereço do Slave
    buf_485[1] = 0x04; // Função MODBUS: Read Input Register
    buf_485[2] = 0x00; // Registro inicial para ler (MSB)
    buf_485[3] = 0x14; // Registro inicial para ler (LSB)
    buf_485[4] = 0x00; // Total de registros para ler (MSB)
    buf_485[5] = 0x02; // Total de registros para ler (LSB)
    crc.i_crc = MB_CalcCRC (6, buf_485, 0xFFFF);
    buf_485[6] = crc.c_crc[0]; // CRC (LSB)
    buf_485[7] = crc.c_crc[1]; // CRC (MSB)
}

#define NUM_SHIFT 8
#define POLINOMIAL 0xA001 // Polinomial constant (RTU mode)
unsigned int MB_CalcCRC (unsigned char num_of_char, unsigned char *ptr_num,
register unsigned int checksum)
{
    char i,j;
    // for all char
    for (i=0; i < num_of_char; i++)
    {
        // execute XOR
        checksum ^= *ptr_num;
        ptr_num++;
        // shift checksum 1 bit right
        for (j=1; j < NUM_SHIFT+1; j++)
        {
            // if LSBit = 0
            if ((checksum & 0x01) == 0)
                checksum >>= 1;
            else
                checksum = (checksum >> 1) ^ POLINOMIAL;
        }
    }
    return(checksum);
}
```