Procedimento para cálculo do CRC

Histórico da Revisão

Data	Versão	Descrição	Autor
09/11/2009	1.0	Versão inicial	Glaydstone Gonçalves da Cunha/ Marcus Fortes

TERMO DE CONFIDENCIALIDADE

SUMÁRIO

1. Introdução

Esse documento descreve um exemplo de algoritmo para cálculo do CRC (Cyclic Redundancy Check) tomando como referências o pacote de posição enviado pelo MXT e o ACK da aplicação que recebe os dados desses equipamentos. O calculo do CRC gera um valor que permite a identificação de erros de transmissão a partir da analise do pacote.
O CRC dos pacotes MXT é o CRC CCITT 16.

2. Pacote de posição e ACK de confirmação de posição

2.1. Pacote de posição

O pacote de dados enviado pelo MXT possui o seguinte frame:

Campo	Descrição	Informação
SOF	Início do frame	0x01
DD	Identificador do modelo de equipamento que enviou a mensagem	➢ **MXT-100:** 0xA0 ➢ **MXT-101:** 0xA1 ➢ **MXT-150:** 0xA2 ➢ **MXT-151:** 0xA3
MT	Tipo de mensagem enviada	➢ **Ack:** 0x02 ➢ **Nack:** 0x03 ➢ **Pacote de posição:** 0x31
DEVID	ID do equipamento	ID do módulo
DATA	Informações coletadas pelo equipamento	Coordenadas geográficas, status das entradas, saídas, etc.
CRC	CRC calculado a partir da mensagem atual	Valor do CRC
EOF	Final de frame	0x04

Campos usados para o cálculo do CRC

Tabela 1: Datagrama correspondente ao pacote de posição

2.2. ACK de confirmação da posição

O ACK de confirmação da posição enviada pela aplicação que recebe os dados pelo MXT, deverá possuir o seguinte formato:

Campo	Descrição	Informação
SOF	Início do frame	01
DD	Modelo de equipamento	➢ **MXT-100:** 0xA0 ➢ **MXT-101:** 0xA1 ➢ **MXT-150:** 0xA2 ➢ **MXT-151:** 0xA3
MT	Tipo de mensagem enviada	➢ **Ack:** 0x02 ➢ **Nack:** 0x03 ➢ **Pacote de posição:** 0x31 ➢ **Comando enviado pelo servidor (GPRS):** 0x32
DEVID	Id do módulo	Id do módulo
CRC – POS	CRC da posição enviada pelo módulo	CRC da posição em relação ao qual esse é o ACK de reposta
CRC – ACK	CRC gerado pela aplicação	Valor do CRC
EOF	Final de frame	04

Campos usados para o cálculo do CRC

Tabela 2: Datagrama correspondente ao ACK de confirmação da posição

3. Exemplo de código para o cálculo do CRC

A linguagem utilizada para o exemplo é C++.

Classe principal:

```
#include <string>
#include <algorithm>
#include <fstream>
#include <sstream>
#include <iostream>
#include "crc_ccitt16.hpp"

/* This sample build a CRC_CCITT16 word to use in position ACK command. */

/*Este exemplo demonstra o cálculo de uma String contendo um CRC para utilização em um ACK de posição*/

void main()
{
/*Pacote de posição:

01 a3 31 05 00 50 00 08 06 cb 45 89 f9 18 1c f6 d2 cf Fe 84 62 61 fd 80 23 28 80 02 00 05

03 2e 06 02 00 49 1b f4 27 50 00 83 00 7e 50 53 8d 04

CRC correspondente ao pacote de posição enviado pelo módulo: 538d

Para montar pacote correspondente ao ACK de posição, deve-se extrair do pacote correspondente à posição recebida as informações contendo o tipo de equipamento, tipo de transmissão, ID do módulo e o crc do pacote. Em seguida, calcular o CRC tomando esses dados como base e o inserir no pacote.

ACK de confirmação de recebimento dessa posição – 01 a3 02 05 00 50 00 53 8d 8c 03 04. Esse é um ACK enviado por uma aplicação já funcional.


CRC correspondente ao ack de posição gerado pela aplicação para essa posição: 8c03*/

        unsigned short crc = 0;

        crc = CRC::CRC16_CCITT(crc,0xa3); // Device Type
        crc = CRC::CRC16_CCITT(crc,0x02); // ACK Type Packet
        crc = CRC::CRC16_CCITT(crc,0x05); // ID
        crc = CRC::CRC16_CCITT(crc,0x00); // ID
        crc = CRC::CRC16_CCITT(crc,0x50); // ID
        crc = CRC::CRC16_CCITT(crc,0x00); // ID
        crc = CRC::CRC16_CCITT(crc,0x53); // Param Byte (Position ACK)
        crc = CRC::CRC16_CCITT(crc,0x8d); // Param Byte (Position ACK)

        std::cout << "CRC: " << std::hex << crc << " | CRC Posicao: " << ((crc&0xFF)<<8) + ((crc&0xFF00)>>8);
}
```

Esse exemplo apenas imprime o valor do CRC. Exibe na tela o seguinte texto:

CRC: 038c | CRC Posicao: 8c03

```
#ifndef _CRC_CCITT16_HPP
#define _CRC_CCITT16_HPP
	namespace CRC {
		const unsigned short crc16_ccitt_table[256] = {
			0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
			0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
			0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
			0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
			0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
			0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
			0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
			0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
			0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
			0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
			0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
			0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
			0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
			0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
			0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
			0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
			0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
			0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
			0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
			0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
			0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
			0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
			0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
			0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
			0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
			0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
			0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
			0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
			0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
			0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
			0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
			0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
		}

//Tabela de referência para obtenção dos valores de CRC

//função para cálculo do CRC
		inline unsigned short CRC16_CCITT(unsigned short crc, unsigned char b)

		{
			unsigned char pom;

			pom = b ^ (crc >> 8);
			return ((crc << 8) ^ crc16_ccitt_table[pom]);
		}
	}
#endif // _CRC_CCITT16_HPP
```

A função CRC16_CCITT recebe como parâmetros, em cada chamada realizada na classe principal, o valor do CRC calculado e um fragmento do ACK. Na última chamada, é obtido o valor final do CRC que deverá estar no ACK de confirmação da posição.