CRC 16 Modbus Berechnung

Kundenfrage:

Ich kommuniziere mit dem UMG über eine RS485 Schnittstelle (Modbus) mit einer SPS. Den Antwortstring möchte ich in der SPS auf Fehlerfreiheit überprüfen. Die letzen beiden Bytes dieses Strings wären, soweit ich in der Dokumentation gelesen habe, die CRC Prüfsumme. Hierzu benötige ich das Generatorpolynom, aus dem die Prüfsumme generiert wird. In der Doku habe ich nichts darüber gefunden. Können Sie mir weiterhelfen?

Antwort:

```
const unsigned char CRC_hi[] = {
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,
0x40,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,
0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,
0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,
0x40,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,
0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,
0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,
0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x01,0xc0,
0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,
0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,
0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x01,0xc0,
0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,
0xc0,0x80,0x41,0x00,0xc1,0x81,0x40,0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,
0x00,0xc1,0x81,0x40,0x01,0xc0,0x80,0x41,0x01,0xc0,0x80,0x41,0x00,0xc1,0x81,
0x40 };
const unsigned char CRC_lo[] = {
0x00,0xc0,0xc1,0x01,0xc3,0x03,0x02,0xc2,0xc6,0x06,0x07,0xc7,0x05,0xc5,0xc4,
0x04,0xcc,0x0c,0x0d,0xcd,0x0f,0xcf,0xce,0x0e,0x0a,0xca,0xcb,0x0b,0xc9,0x09,
0x08,0xc8,0xd8,0x18,0x19,0xd9,0x1b,0xdb,0xda,0x1a,0x1e,0xde,0xdf,0x1f,0xdd,
0x1d,0x1c,0xdc,0x14,0xd4,0xd5,0x15,0xd7,0x17,0x16,0xd6,0xd2,0x12,0x13,0xd3,
0x11,0xd1,0xd0,0x10,0xf0,0x30,0x31,0xf1,0x33,0xf3,0xf2,0x32,0x36,0xf6,0xf7,
0x37,0xf5,0x35,0x34,0xf4,0x3c,0xfc,0xfd,0x3d,0xff,0x3f,0x3e,0xfe,0xfa,0x3a,
0x3b,0xfb,0x39,0xf9,0xf8,0x38,0x28,0xe8,0xe9,0x29,0xeb,0x2b,0x2a,0xea,0xee,
0x2e,0x2f,0xef,0x2d,0xed,0xec,0x2c,0xe4,0x24,0x25,0xe5,0x27,0xe7,0xe6,0x26,
0x22,0xe2,0xe3,0x23,0xe1,0x21,0x20,0xe0,0xa0,0x60,0x61,0xa1,0x63,0xa3,0xa2,
0x62,0x66,0xa6,0xa7,0x67,0xa5,0x65,0x64,0xa4,0x6c,0xac,0xad,0x6d,0xaf,0x6f,
0x6e,0xae,0xaa,0x6a,0x6b,0xab,0x69,0xa9,0xa8,0x68,0x78,0xb8,0xb9,0x79,0xbb,
0x7b,0x7a,0xba,0xbe,0x7e,0x7f,0xbf,0x7d,0xbd,0xbc,0x7c,0xb4,0x74,0x75,0xb5,
0x77,0xb7,0xb6,0x76,0x72,0xb2,0xb3,0x73,0xb1,0x71,0x70,0xb0,0x50,0x90,0x91,
0x51,0x93,0x53,0x52,0x92,0x96,0x56,0x57,0x97,0x55,0x95,0x94,0x54,0x9c,0x5c,
0x5d,0x9d,0x5f,0x9f,0x9e,0x5e,0x5a,0x9a,0x9b,0x5b,0x99,0x59,0x58,0x98,0x88,
0x48,0x49,0x89,0x4b,0x8b,0x8a,0x4a,0x4e,0x8e,0x8f,0x4f,0x8d,0x4d,0x4c,0x8c,
0x44,0x84,0x85,0x45,0x87,0x47,0x46,0x86,0x82,0x42,0x43,0x83,0x41,0x81,0x80,
0x40 };
unsigned short crc16_p(unsigned short crc, void *data, short len)
{
unsigned char p;
while (len--) {
p=(crc >> 8) ^ *(unsigned char *)data++;
crc = ((crc ^ CRC_hi[p]) << 8) | ( CRC_lo[p] );
};
return crc;
}
```

Die Funktion crc16_p berechnet die CRC-Prüfsumme des Puffers data mit der Länge von len Bytes. Als Startwert muss in crc der Wert 0xFFFF übergeben werden.