CRC-Berechnung

In diversen U880-Programmen, z.B. EPROM-Software, wird oftmals eine Prüfsumme ausgegeben. Dabei handelt es sich fast immer um eine 16 BIT-CRC-Prüfsumme, d.h. ein 17-Bit-Polynom, nach Standard CCITT:

CRC-CCITT (CRC-16) x^16 + x^12 + x^5 + 1

s. Wikipedia

Als Startwert wird eigentlich immer 0FFFFh genommen.

In Perl kann man die CRC so berechnen (nicht optimiert, reine Umsetzung des Polynoms!). Die Und-Verknüpfung mit 0x8000 erfolgt zur Maskierung des Hi-Bits 15; Die Und-Verknüpfung mit 0xFFFF ist nötig, um das Ergebnis als 16Bit-Zahl zu belassen.

```
$buf = ....;    #Arrays von 2KiByte FFh
$len = 2048;    #Anzahl der Bytes

#CRC-CCITT (CRC-16) x16 + x12 + x5 + 1
$POLY = 0b_0001_0000_0010_0001; # das 17. Bit (x^16) entfällt,
                                # da nur mit 16 Bit gearbeitet wird

#Startwert
$crc16 = 0xFFFF;

for ($i=0;$i<$len;$i++) {
    my $byte = ord(substr($buf,$i,1));     # nächstes Byte aus Buffer holen

    $byte = $byte * 0x100;                # in 16 Bit wandeln
    for (0..7) # 8 Bits pro Byte
    {
        if (($byte & 0x8000) ^ ($crc16 & 0x8000)) {
        # wenn die Hi-Bits unterschiedlich sind, dann
            $crc16 <<= 1;        # shift left
            $crc16 ^= $POLY;    # XOR-Verknüpfung mit CRC-Poly
            $crc16 &= 0xFFFF;    # beschränken auf 16 Bit
        } else {
        # ansonsten nächstes Bit ohne Verküpfung
            $crc16 <<= 1;        # shift left
            $crc16 &= 0xFFFF;    # beschränken auf 16 Bit
        }
        $byte <<= 1;       # shift left, nächstes Bit
        $byte &= 0xFFFF;
    }
}

# Ausgabe
printf "CRC = %.4X\n", $crc16;
```

Normalerweise werden CRC-Polynome mit reverser Bit-Reihenfolge berechnet; auch die einzelnen Bytes werden in umgekehrter Reihenfolge abgearbeitet. Und richtig optimal wird es erst mit vorbrechneten Tabellen...

In Assembler sieht die CRC-Routine wie folgt aus. Die Berechnung ist optimiert und erfolgt tetradenweise. (Der Code stammt aus der Z9001-EPROM-Software)

in: DE = Startadr., BC = Länge
out: HL = CRC

```
crc:        ld    hl, 0FFFFh
crc1:       ld    a, (de)
        xor    h
        ld    h, a
        rrca
        rrca
        rrca
        rrca
        and    0Fh
        xor    h
        ld    h, a
        rrca
        rrca
        rrca
        push    af
        and    1Fh
        xor    l
        ld    l, a
        pop    af
        push    af
        rrca
        and    0F0h
        xor    l
        ld    l, a
        pop    af
        and    0E0h
        xor    h
        ld    h, l
        ld    l, a
        inc    de
        dec    bc
        ld    a, b
        or    c
        jr    nz, crc1
        ret
```

s.a.

- http://www.robotrontechnik.de/html/forum/thwb/showtopic.php?threadid=3846
- http://www.ac1-info.de/literatur/fa_86_11.htm (Berechnung nach SDLC, mit Bit-Schieberegister)