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.

```
#Arrays von 2KiByte FFh
$buf = ...;
1 = 2048;
                #Anzahl der Bytes
\#CRC-CCITT (CRC-16) \times 16 + \times 12 + \times 5 + 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
                             # shift left
            $crc16 <<= 1;
            $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, OFFFFh
crc1:
             ld
                     a, (de)
                 h
         xor
         ld
                h, a
         rrca
         rrca
         rrca
         rrca
         and
                 0Fh
         xor
                 h
         ld
                h, a
         rrca
         rrca
         rrca
                  af
         push
                 1Fh
         and
         xor
                 l
         ld
                l, a
                 af
         pop
         push
                  af
         rrca
                 0F0h
         and
                 ι
         xor
         ld
                l, a
                 af
         pop
                 0E0h
         and
         xor
                 h
         ld
                h, l
         ld
                l, a
         inc
                 de
         dec
                 bc
         ld
                a, b
         or
                nz, crc1
         jr
         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)

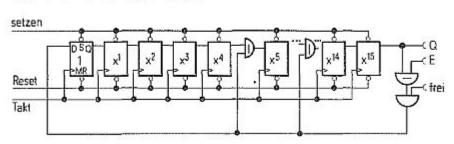
Hardware

aus mc 1984/07

CRC ist die Abkürzung für Cyclic Redundancy Check und so etwas ähnliches wie eine Prüfsumme, darf aber damit nicht verwechselt werden, da die Erzeugung des CRC aufwendiger ist. Dabei werden nicht einfach die einzelnen Bytes aufaddiert, sondern verschiedene Bits gemäß einem sogenannten Generator- Polynom. Es gibt dabei sehr unterschiedliche Vorschriften, jedoch verwendet man bei den gängigen Controllern das vom CCITT definierte Polynom. Es lautet $G(x) = 1 + x^5 + x^12 + x^16$. Daraus kann man eine Schaltung konstruieren, die etwa wie in Bild 16 aussieht. Ein Reset-Eingang sorgt dafür, daß das Schieberegister auf einen definierten Wert gesetzt werden kann. Dann werden der Eingang FREI auf 1 gelegt und zusammen mit einem Takt die Daten an E angelegt. Nach dem Ende des Datenstroms wird FREI auf 0 gelegt, und die CRC-Bytes können aus dem Register geschoben werden. Um nun einen Datenstrom zu testen, wird genauso wie vorher verfahren, nur daß nun auch die CRC-Bytes mitverrechnet werden. Das Ergebnis im Schieberegister muß anschließend 0 sein.

 $G(x)=1+x^5+x^{12}+x^{16}$ (CRC-CCITT)

Bild 16. Erzeugung von CRC-Bytes mit einem Schieberegister



From:

https://hc-ddr.hucki.net/wiki/ - Homecomputer DDR

Permanent link:

https://hc-ddr.hucki.net/wiki/doku.php/cpm/crc?rev=1387210221

Last update: 2013/12/16 16:10

