

Hier nun das erwünschte Beispiel für Rechenfehler auch wegen falschen Carry-Bit-Setzen. Es handelt sich um eine schlichte 16-bit-Subtraktion. Das Beispiel ist ein Auszug aus dem Prog. „Anzeige Regeneration Dieselpartikelfilter“. Da werden u.a. die Werte eines analogen Eingangs (10-bit) ausgelesen und für die Anzeige auf drei 7-Seg-Anzeigen (Abk.: 7SA) aufbereitet. Für die Hunderter und Zehner wird vom Datenwert jeweils dez'100', resp. d'10' solange abgezogen, bis sich ein negatives Ergebnis zeigt. Die Rundendurchläufe repräsentieren dann je den Wert des Hunderters, resp. des Zehners.

Als willkürliches Beispiel wird ein Analog-Wert von d'280' angenommen. Um den aufzubereiten, wären „normal“ drei aufeinander folgende 16-bit-Subtraktionen im Hunderter-Bereich notwendig. Der Ausführlichkeit halber und weil es praktisch im Programm auch so abläuft, werden die insgesamt sechs 8-bit-Subtraktionen dargestellt.

Gerechnet wird: adc minus bit, das Ergebnis wird in adc gespeichert

1. Runde: 280 minus 100 // adc hält h'0118' = d'280' und bit hält h'0064' = d'100'

---

RE 1) 280-100=180

adclow    h'18' / d'24'  
- bitlow   -h'64' / d'100'

---

adclow    h'b4' / d'-180'    Flags: N ja, C nein    //Ergebnis negativ=Decrement high bit nötig

RE 2)    adchigh hielt h'01', nach Decrement nun h'00'

adchigh   h'00' / d'0'  
- bitlow   -h'00' / d'0'

---

adchigh   h'00' / d'0'    Flags: N nein, C ja    // Ergebnis positiv

In Schleife wird Hunderter-Zähler incremiert = Hundert hält d'1'

---

RE3) 180-100=80

adclow    h'b4' / d'180'  
- bitlow   -h'64' / d'100'

---

adclow    h'50' / d'80'    Flags: N nein, C ja    //Ergebnis positiv= kein Decrement high bit

RE 4)

adchigh   h'00' / d'0'  
- bitlow   -h'00' / d'0'

---

adchigh   h'00' / d'0'    Flags: N nein, C ja    // Ergebnis positiv

In Schleife wird Hunderter-Zähler incremiert = Hundert hält d'2'

---

RE 5) 80-100= -180

```
adclow    h'50' / d'80'  
- bitlow  -h'64' / d'100'
```

-----  
adclow h'ec' / d'236' Flags: N ja, C nein //Ergebnis negativ = Decrement high-bit nötig

RE 6) adchigh hielt h'00', nach Decrement nun h'ff' / d'255'

```
adchigh   h'ff' / d'255'  
- bitlow  -h'00' / d'0'
```

-----  
adchigh h'ff' / d'255' Flags: N ja, C ja // Ergebnis positiv

In Schleife wird fälschlich Hunderter-Zähler incremiert=Hundert hält d'3' // RECHENFEHLER !!!

-----  
Die Rechnung 6) , die Subtraction des High-Bytes produziert Fehler am laufenden Bande:

1. aus dem eigentlich negativen Wert des High-Bytes nach dem Decrement, das danach h'ff' hält, wird nun auf einmal ein positiver Wert. Dieses ursprünglich negative h'ff' verursacht den nachfolgenden Kollaps:
2. es werden sowohl das N-Flag als auch das C-Flag gesetzt und das kann nur falsch sein.

Der danach erfolgende Sprung mit BNC bewirkt ein weiteres und damit falsches Hochzählen des Hunderter-Zählers:

Das Rechenergebnis wird falsch und laufend falscher.

3. zusätzlich zum Desaster des falschen Ergebnisses gelangt das Programm in eine Endlosschleife und ist nur noch mit dem weiteren Verfälschen des Hunderter-Zählers beschäftigt.

Somit ist nachgewiesen, dass sowohl das Benutzen des N-Flags im Sprung-Befehl „bra“ als auch dasjenige des C-Flags zu Fehlern führen kann.

Das System des Zweier-Komplements ist über seine Grenzen getreten, rsp. hat solche erhalten. Nur mit Programmier-Tricks, rsp. - Ausflüchten, rsp. Vermeidungs-Strategien kann ein sauberes, genauer gesagt: normales Rechnen ermöglicht werden.

Das angehängte Zip-File enthält 3 Dateien:

- den kompletten vom MPLAB erstellten Ordner „rechnen08“ u.a. mit dem ASM-File
- dieses ASM-File als Text-File
- diesen im Forum eingestellten Beitrag als PDF-File mit den einzelnen Rechenbeispielen, die sich ausgedruckt wohl bequemer nachvollziehen lassen.

Grüße, picass