Das Taster-Interrupt-Problem

Es war eine ganz einfache Aufgabenstellung.

Ich hatte ein Messgerät für CO2 und Sauerstoff gebaut, die Werte wurden auf einem LCD angezeigt. Da ich noch einen Barometersensor hatte, der seit langem auf seinen Einsatz wartete, wurde dieser mit eingebaut. Mit einer kleinen Zusatzberechnung konnte sogar noch ein Höhenmesser mit integriert werden. Aber nun musste der Anzeigemodus gewechselt werden können, da nicht alles auf dem LCD gleiczeitig seinen Platz fand.

Kein Problem, ein Taster sollte für diese Aufgabe genügen.
Eine Variable "Displaymode" würde bei jedem Tastendruck hochgezählt werden (modulo 3, da es 3 Displaymodes geben sollte), und entsprechend die Ausgabe steuern.

Da es im Hauptprogramm etwas längere Zeit brauchte für Messen, Berechnen usw. schien es mir keine gute Idee, mit Polling zu arbeiten. Man könnte schnell einen Tastendruck verpassen.

Also Interruptsteuerung.

Nun gibt es aber ein Problem. Wenn der Taster prellt, werden mehrere Interrupts ausgelöst und die Variable Displaymode entsprechend auf den falschen Wert gesetzt.

Entprellen sollte eigentlich kein großes Problem darstellen, es ist ja eine Standardaufgabe bei Mikrocontrollern. Man muß eigentlich nur ein wenig warten bis der Zustand stabil ist.
Aber Warten in einer Interrupt-Service-Routine (ISR)? Damit hat man den ganzen Controller blockiert bis es weiter geht. Ich habe bei den ersten Tests mit dieser Methode experimentiert, aber gefallen hat mir die Lösung nicht.

Dann gibt es aber noch eine weiter Sache die Schwierigkeiten macht: es kann passieren dass während der Abarbeitung des Interrupts durch das Prellen ein weiterer Interrupt ausgelöst wird, so daß die ISR mehrmals aufgerufen wird.
Ganz naiv habe ich zuerst versucht, den Interrupt während der ISR zu sperren. dies nützt aber nichts, denn wenn ein Interrupt auftritt, wird das Ereignis im Interruptflag gespeichert bis der Interrupt abgearbeitet wird.

Also, die Sache ist doch ein wenig komplizierter als ich zunächst angenommen hatte.

In so einem Fall gibt es zwei Möglichkeiten: im Internet (oder altmodisch in Büchern) recherchieren ob schon jemand eine Lösung beschrieben hat, oder selbst eine Lösung finden.
Da meine Recherche nicht sehr erfolgreich war, habe ich zu der zweiten Methode gegriffen.

Hierzu wurde das Programm einmal soweit abgespeckt, daß nur noch das übrig blieb was das Problem verursachte.

Es war klar: das wichtigste war, daß das Interruptflag vor dem Verlassen der ISR gelöscht wurde, und zwar erst ganz zum Schluß.

Ein wenig Herumprobieren zeigte, daß dies meist funktionierte, aber nicht immer.

Als sichere Vorgehensweise ergab sich diese Strategie in der ISR (bei einem Taster nach GND, mit Pullup, z.B. an PIND.3):

1. Warten bis der Taster losgelassen wird
   ```
   Do
   ```

```
Loop Until Pind.3 = 1
```

2. Noch einige Millisekunden warten bis das Prellen vorbei ist.
   `Waitms 10`

3. Ein Flag setzen das im Hauptprogramm ausgewertet werden kann.
   `Buttonflag = 1`

4. Das Interruptflag im GIFR-Register zurücksetzen.
   Achtung: dies geschieht durch Schreiben einer 1 (!) :
   `sbi Gifr, 7` für INT1 (PinD.3)
   `sbi Gifr, 6` für INT0 beim Mega8

Im Hauptprogramm wird dann das Flag abgefragt und, wenn es gesetzt ist die entsprechende Aktion ausgeführt. Hier darf man natürlich nicht vergessen, das Flag auch wieder rückzusetzen.

```
Do
     'If button was pushed, do something
     If Buttonflag = 1 Then
         Buttonflag = 0
         Toggle Portd.7
         Incr Btmp
         Print Btmp
     End If

      '.... other actions
Loop
```

Hier ein funktionierendes Testprogramm, bei dem einfach ein Zähler bei jedem Tastendruck hochzählt:

```
$crystal = 8000000
$regfile = "m8def.dat"
Baud = 9600

'switch to ground (with pullup) on interrupt , falling edge
Config Pind.3 = Input
Portd.3 = 1
On Int1 Switchdisplaymode
Config Int1 = Falling
Enable Int1
Enable Interrupts

Dim Btmp As Byte
Dim Buttonflag As Bit

Config Portd.7 = Output     'LED as indicator

'-----------------------------------------------------------------------
Do
     'If button was pushed, do something
     If Buttonflag = 1 Then
         Buttonflag = 0             ' reset flag
         Toggle Portd.7             ' toggle LED
         Incr Btmp                  ' increment + print counter
         Print Btmp
     End If
Loop
```

```
'-----------------------------------------------------
Switchdisplaymode:

      ''Disabling INT is not necessary

         'Looping as long as key is pressed is good,
         'as it avoids reaction to bouncing on rising edge
        Do
        Loop Until Pind.3 = 1

      'Wait a little bit until situation is stable
      Waitms 10


      'Do what you have to do, in this case set Flag
      Buttonflag = 1

     'Reset INT1 flag in GIFR register to forget INT1
     'orruring during execution of INT1  service routine
     sbi Gifr, 7
Return
```