Vortrag auf der HAM Radio 2009

In sieben Schritten zum Vor- / Rückwärtszähler

Referent:
Dipl. Ing. (FH) Stefan Lehmann

26.06.2009

Hier werden die sieben Programme, die Gegenstand des Referantes waren, aufgelistet und kurz kommentiert. Der eigentliche Text des Vortrages ist als extra Dokument verfügbar.

Um schnell in die Materie „Microcontroller Programmierung“ einsteigen zu können, bietet es sich an, diesen Microcontroller zuerst als Black Box zu betrachten. Man benötigt nämlich nur ganz wenige Informationen aus dem Datenblatt um mittels einer Hochsprache schnell einfache Programme zu schreiben. Das führt zu schnellen Erfolgserlebnissen, was besonders in der Anfangsphase wichtig ist.

Ein gewisses Grundverständnis für Strom und Spannung, etwas logisches Verständnis und Phantasie sind schon notwendig.

Dieses erste Programm kann sehr gut als Vorlage für eigene Programme dienen, da es bereits ein Grundgerüst besitzt, an das man sich als Programmierer halten sollte. Wenn man nach diesem Muster arbeitet, bleiben die Programme einigermaßen übersichtlich.

Schritt 1

Nachdem man sich im Schaltplan etwas kundig gemacht hat, soll sofort ein Segment der Anzeige zu leuchten gebracht werden. Dazu sucht man sich ein Segment aus und verfolgt die Leitungen bis zum Microcontroller zurück. In unserem Fall ist es das Segment A auf der rechten Stelle der Anzeige.

Dieses Segment ist mit seiner Kathode (Minuspol) an RB6 und mit seiner Anode über den Transistor T4 mit RA0 verbunden. Legt man an RB6 0V an und steuert T4 mit 5V an, dann leuchtet dieses Segment auf.

Startet man nun dieses Programm, wird man zum Erstaunen feststellen, dass nicht nur das gewünschte Segment, sondern auch ander A-Segmente leuchten.

Was ist hier passiert?

Die Antwort lesen Sie bei Schritt 2.

```
'*****************************************************************************
'                              Vor- und Rückwärtszähler
'                                  F_R_Counter.BAS
'
'                                 Stand: 24.06.2009
'
'                                    1. Schritt
'
'(c) Ing.Büro Lehmann, Fürstenbergstraße 8a, 77756 Hausach, www.iL-online.de
'*****************************************************************************

'Historie

'Besonderheiten
'Auf der PICUP-Platine müssen die Steckbrücken (Jumper) wie folgt gesteckt sein:
'JP1A und JP1B  jeweils 2-3   -> externer Oszillator (Quarz)
'JP2A und JP2B  jeweils 2-3   -> T3 an RA1  und T2 an RA2
'

'die beiden nachfolgenden Schalter sind notwendig, wenn das PicKit2 als
'Programmiergerät verwendet wird.
        $LIST  /M_OBJ
        $LIST  OBJ2HEX

'hier wird der Prozessortyp und dessen Spezifikationen definiert
        DEFINE DEVICE 16F628A
        CONFIG CMCFG7
        CONFIG HS_OSC
        CONFIG MCLR_INT

        XTAL   4.096

'hier folgen die Variablendeklarationen (8, 16 und ggf. 32 Bit)


'hier werden die Bitvariablen definiert


'hier werden die Konstanten definiert

'-----------------------------------------
'ab hier wird Code für den Prozessor erzeugt
'-----------------------------------------
'falls Unterprogramme verwendet werden, werden diese zuerst übersprungen

cold:
        GOTO      start

'hier stehen ggf. die Unterprogramme


'hier beginnt das Hauptprogramm
start:
        LOW       RB,6                'Segment A
        HIGH      RA,0                '1. Digit (T4)

        END
```

Schritt 2

Warum leuchtet nicht nur ein A-Segment auf?

Dazu etwas Hintergrundwissen:

1. Wird ein Microcontroller eingeschaltet, werden nur ganz wenige Dinge definiert. Im Allgemeinen haben die Speicher also irgend einen Wert gespeichert, der nicht Null sein muss.
2. Alle Pins werden i.d.R. auf Eingang gestellt. Sie sind somit hochohmig und ein von außen angelegtes Signal kann den Zustand High (5V) oder Low (0V) annehmen.

In diesem Fall hat Punkt 2 einen Einfluss. Die npn-Transistoren haben in ihrem Emitterzweig die LED liegen. Dadurch verschiebt sich das Potenzial an der Basis, so dass der Transistor leitend wird. In dieser Schaltung muss somit die jeweilige Basis der Transistoren explizit auf Low geschaltet werden.

```
'*****************************************************************************
'                              Vor- und Rückwärtszähler
'                                  F_R_Counter.BAS
'
'                                 Stand: 24.06.2009
'
'                                    2. Schritt
'
'(c) Ing.Büro Lehmann, Fürstenbergstraße 8a, 77756 Hausach, www.iL-online.de
'*****************************************************************************

'Historie
'Schritt2: weil mehr wie ein Segment leuchten, werden die übrigen explizit
'          abgeschaltet.

'Besonderheiten
'Auf der PICUP-Platine müssen die Steckbrücken (Jumper) wie folgt gesteckt sein:
'JP1A und JP1B  jeweils 2-3   -> externer Oszillator (Quarz)
'JP2A und JP2B  jeweils 2-3   -> T3 an RA1  und T2 an RA2

'die beiden nachfolgenden Schalter sind notwendig, wenn das PicKit2 als
'Programmiergerät verwendet wird.
         $LIST  /M_OBJ
         $LIST  OBJ2HEX

'hier wird der Prozessortyp und dessen Spezifikationen definiert
         DEFINE DEVICE 16F628A
         CONFIG CMCFG7
         CONFIG HS_OSC
         CONFIG MCLR_INT

         XTAL   4.096

'hier folgen die Variablendeklarationen (8, 16 und ggf. 32 Bit)


'hier werden die Bitvariablen definiert


'hier werden die Konstanten definiert

'-----------------------------------------
'ab hier wird Code für den Prozessor erzeugt
'-----------------------------------------
'falls Unterprogramme verwendet werden, werden diese zuerst übersprungen

cold:
         GOTO      start

'hier stehen ggf. die Unterprogramme


'hier beginnt das Hauptprogramm
start:
         LOW       RB,6                'Segment A
         HIGH      RA,0                '2. Digit (T3) einschalten
         LOW       RA,1                '4. Digit (T1) ausschalten
         LOW       RA,2                '3. Digit (T2) ausschalten
         LOW       RA,4                '1. Digit (T4) ausschalten

         END
```

Schritt 3

Hier soll das gewünschte Segment blinken, um auch ganz sicher zu gehen, dass es unser Programm ist was dieses Segment ansteuert. Aus Schritt 1 haben wir die Erfahrung gemacht, dass es auch andere Ursachen gibt, die u.U. auch fehlinterpretiert werden können.

Der Blinkvorgang wird einfach so realisiert, dass man das Segment einschaltet, eine gewisse Zeit (hier 0,5 Sekunden) wartet, das Segment ausschaltet und erneut 0,5 Sekunden wartet. Anschließend verzweigt man an den Anfang, wo das Segment erneut eingeschaltet wird.

Beim Ein- und Ausschalten reicht es, wenn man den Transistor schaltet. Das Potenzial an der Kathode der LED kann unverändert bleiben.

```
'*****************************************************************************
'                              Vor- und Rückwärtszähler
'                                  F_R_Counter.BAS
'
'                                 Stand: 24.06.2009
'
'                                    3. Schritt
'
'(c) Ing.Büro Lehmann, Fürstenbergstraße 8a, 77756 Hausach, www.iL-online.de
'*****************************************************************************

'Historie
'Schritt3: nun soll das Segment blinken
'Schritt2: weil mehr wie ein Segment leuchten, werden die übrigen explizit
'          abgeschaltet.

'Besonderheiten
'Auf der PICUP-Platine müssen die Steckbrücken (Jumper) wie folgt gesteckt sein:
'JP1A und JP1B  jeweils 2-3   -> externer Oszillator (Quarz)
'JP2A und JP2B  jeweils 2-3   -> T3 an RA1  und T2 an RA2

'die beiden nachfolgenden Schalter sind notwendig, wenn das PicKit2 als
'Programmiergerät verwendet wird.
         $LIST  /M_OBJ
         $LIST  OBJ2HEX

'hier wird der Prozessortyp und dessen Spezifikationen definiert
         DEFINE DEVICE 16F628A
         CONFIG CMCFG7
         CONFIG HS_OSC
         CONFIG MCLR_INT

         XTAL   4.096

'hier folgen die Variablendeklarationen (8, 16 und ggf. 32 Bit)


'hier werden die Bitvariablen definiert


'hier werden die Konstanten definiert

'-----------------------------------------
'ab hier wird Code für den Prozessor erzeugt
'-----------------------------------------
'falls Unterprogramme verwendet werden, werden diese zuerst übersprungen

cold:
         GOTO      start

'hier stehen ggf. die Unterprogramme


'hier beginnt das Hauptprogramm
start:
         LOW       RA,1                '?. Digit (T?) ausschalten
         LOW       RA,2                '?. Digit (T?) ausschalten
         LOW       RA,4                '?. Digit (T?) ausschalten
loop:
         LOW       RB,6                'Segment A
         HIGH      RA,0                '1. Digit (T4) einschalten
```

```
WAIT      500                 'Segment bleibt 500 ms ein
LOW       RA,0                '1. Digit (T4) ausschalten
WAIT      500                 'die Ausphase ist auch 500 ms lang
GOTO      LOOP                'das Ganze läuft nun endlos

END
```

Schritt 4

Nun soll auf der letzten Stelle die Zahlen von 0 bis 9 hochgezählt werden. Dazu benötigen wir eine Zählvariable (Count) sowie eine Hilfsvariable (Muster). In der Variablen Count zählen wir von 0 bis 9 hoch. Da diese Variablen aber 8 Bit besitzen, kann man bis 255 hochzählen. Wir brechen den Zählvorgang aber bei der Zahl 10 ab, und setzen die Variable wieder auf Null.

Die zweite Variable (Muster) benötigen wir um aus der Zahl in der Count-Variable ein Muster zu machen, das wir auf die Anzeige geben. Dieses Muster muss entsprechend der Zuordnung zwischen den Segmenten A bis G, sowie dem Dezimalpunkt und den Portanschlüssen erstellt werden.

Im Praxisheft 19 ist diese Abbildung zu finden.
So geht RB6 an das Segment A, RB7 an Segment B, RB1 an Segment C, usw. Dabei ist die Reihenfolge der Zuordnung beliebig. Man wählt letztendlich eine Zuordnung, bei der das Platinenlayout einfach wird. Die richtige Zuordnung ist in der Software ein Kinderspiel. Die entsprechende Programmsequenz lautet:

```
'bcd2seg wandelt eine Zahl zwischen 0 und 9 in ein Bitmuster für die
'Siebensegmentanzeige um
bcd2seg:
    IF      Count=0 THEN LET Muster = %00100001
    IF      Count=1 THEN LET Muster = %01111101
    IF      Count=2 THEN LET Muster = %00010011
    IF      Count=3 THEN LET Muster = %00011001
    IF      Count=4 THEN LET Muster = %01001101
    IF      Count=5 THEN LET Muster = %10001001
    IF      Count=6 THEN LET Muster = %10000001
    IF      Count=7 THEN LET Muster = %00111101
    IF      Count=8 THEN LET Muster = %00000001
    IF      Count=9 THEN LET Muster = %00001001
    RETURN
```

Das Muster wird für 0,5 Sekunden angezeigt, anschließend wird der Zähler um eins erhöht. Wird der Wert 10 erreicht, sorgt die folgende Anweisung dafür, dass Count auf Null gesetzt wird.

```
    IF      Count=10 THEN LET Count=0  'Zähler zählt nur bis 9
```

```
'*****************************************************************************
'                              Vor- und Rückwärtszähler
'                                  F_R_Counter.BAS
'
'                                 Stand: 24.06.2009
'
'                                    4. Schritt
'
'(c) Ing.Büro Lehmann, Fürstenbergstraße 8a, 77756 Hausach, www.iL-online.de
'*****************************************************************************

'Historie
'Schritt4: es wird auf de hintersten Stelle von 0 bis 9 hochgezählt
'          zum ersten Mal wird eine Variable definiert, ein Unterprogamm
'          zur Zeichenumsetzung verwendet und der TRIS-Befehl eingesetzt
'Schritt3: nun soll das Segment blinken
'Schritt2: weil mehr wie ein Segment leuchten, werden die übrigen explizit
'          abgeschaltet.

'Besonderheiten
'Auf der PICUP-Platine müssen die Steckbrücken (Jumper) wie folgt gesteckt sein:
'JP1A und JP1B  jeweils 2-3   -> externer Oszillator (Quarz)
'JP2A und JP2B  jeweils 2-3   -> T3 an RA1  und T2 an RA2

'die beiden nachfolgenden Schalter sind notwendig, wenn das PicKit2 als
'Programmiergerät verwendet wird.
         $LIST  /M_OBJ
         $LIST  OBJ2HEX

'hier wird der Prozessortyp und dessen Spezifikationen definiert
         DEFINE DEVICE 16F628A
         CONFIG CMCFG7                'interene Komparatoren abschalten, damit
                                      'die Pins als normale Ein-/Ausgänge arbeiten
         CONFIG HS_OSC                'Wenn ein Quarz verwendet wird
         CONFIG MCLR_INT              'der MCLR-Pin soll als IO-Pin arbeiten
         CONFIG WDT_OFF               'verhindert, dass nach 2,3 Sek ein Reset
                                      'erzeugt wird

         XTAL   4.096

'hier folgen die Variablendeklarationen (8, 16 und ggf. 32 Bit)
         DEFINE    Count = AUTO AS BYTE          'Zählervariable
         DEFINE    Muster = AUTO AS BYTE         'Segmentmuster

'hier werden die Bitvariablen definiert


'hier werden die Konstanten definiert

'-----------------------------------------
'ab hier wird Code für den Prozessor erzeugt
'-----------------------------------------
'falls Unterprogramme verwendet werden, werden diese zuerst übersprungen

cold:
         GOTO      start

'hier stehen ggf. die Unterprogramme

'bcd2seg wandelt eine Zahl zwischen 0 und 9 in ein Bitmuster für die
'Siebensegmentanzeige um
```

```
bcd2seg:
          IF        Count=0 THEN LET Muster = %00100001
          IF        Count=1 THEN LET Muster = %01111101
          IF        Count=2 THEN LET Muster = %00010011
          IF        Count=3 THEN LET Muster = %00011001
          IF        Count=4 THEN LET Muster = %01001101
          IF        Count=5 THEN LET Muster = %10001001
          IF        Count=6 THEN LET Muster = %10000001
          IF        Count=7 THEN LET Muster = %00111101
          IF        Count=8 THEN LET Muster = %00000001
          IF        Count=9 THEN LET Muster = %00001001
          RETURN

'hier beginnt das Hauptprogramm
start:
          LOW       RA,1                '?. Digit (T?) ausschalten
          LOW       RA,2                '?. Digit (T?) ausschalten
          LOW       RA,4                '?. Digit (T?) ausschalten
          LET       Count = 0           'Zähler mit 0 initialisieren
          TRIS      RB,0                'alle Pins von RB arbeiten als Ausgänge
loop:
          GOSUB     bcd2seg             'Zahl in Count umwandeln in Seg.muster
          OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen

          HIGH      RA,0                '1. Digit (T4) einschalten

          WAIT      500                 'Segment bleibt 500 ms ein

          LOW       RA,0                '1. Digit (T4) ausschalten
          LET       Count=Count+1       'Zähler hochzählen lassen
          IF        Count=10 THEN LET Count=0  'Zähler zählt nur bis 9

          GOTO      LOOP                'das Ganze läuft nun endlos

          END
```

Schritt 5

Der Zähler wird nun auf vier Stellen erweitert. Dazu müssen neue Variablen definiert werden. Es sind dies: Count0, Count1, Count2 und Count3. Um weiterhin das Unterprogramm zur Umsetzung des Zahlenwertes in des Siebensegmentmuster benutzen zu können, wird es leicht modifiziert und eine weitere Variable (Wert) definiert. Nun übergibt man vor dem Unterprogrammaufruf die zu konvertierende Stelle der Variablen Wert. Im Unterprogramm haben wir somit nur noch diese Variable zu verarbeiten. Das Ergebnis wird weiterhin in der Variablen Muster zurückgegeben.

Würde man die Anzeigezeit weiterhin bei 0,5 Sekunden belassen, würde jetzt jede einzelne Stelle für diese Zeit angezeigt. Ein stehendes Bild entsteht somit nicht. Das ist das Wesen des Multiplexbetriebs. Man muss die einzelnen Stellen so schnell umschalten, dass die Trägheit des Auges nur noch ein Gesamtbild erkennen kann. Die Einschaltzeiten der einzelnen Stellen werden somit stark verkürzt. Jedoch muss die einzelne Stelle eine gewisse Mindestzeit eingeschaltet sein, damit auch die LED genügend hell scheinen. Bei einer 4-stelligen Anzeige sind 5 Millisekunden ein guter Kompromiss.

Der 4-stellige Zähler arbeitet wie folgt:
Hat die niederwertigste Stelle den Wert 10 erreicht, wird sie, wie bisher auch, auf Null gesetzt. Dann aber wird die nächst höhere Stelle weitergezählt und auch auf den Wert 10 geprüft. Das erfolgt solange, bis alle Stellen bearbeitet sind.

```
'*************************************************************************
'                            Vor- und Rückwärtszähler
'                                F_R_Counter.BAS
'
'                               Stand: 24.06.2009
'
'                                  5. Schritt
'
'(c) Ing.Büro Lehmann, Fürstenbergstraße 8a, 77756 Hausach, www.iL-online.de
'*************************************************************************

'Historie
'Schritt5: nun arbeiten auch die anderen Stellen und werden dabei hoch-
'          gezählt. Die Pause ist kleiner, damit der Zähler schneller läuft
'          Es müssen weitere Variablen für den Zähler verwendet werden (DEFINE)
'          Auch der Multiplexbetrieb wird nun verwendet
'          Allerdings läuft der Zähler nun deutlich schneller, ohne dass man
'          darauf Einfluss hat. Denn ändert man die 5ms beginnt es u.U. zu flimmern
'          Dies ist allerdings nicht so tragisch, da das Endprodukt nicht
automatisch
'          hochzählen soll.
'Schritt4: es wird auf de hintersten Stelle von 0 bis 9 hochgezählt
'          zum ersten Mal wird eine Variable definiert, ein Unterprogamm
'          zur Zeichenumsetzung verwendet und der TRIS-Befehl eingesetzt
'Schritt3: nun soll das Segment blinken
'Schritt2: weil mehr wie ein Segment leuchten, werden die übrigen explizit
'          abgeschaltet.

'Besonderheiten
'Auf der PICUP-Platine müssen die Steckbrücken (Jumper) wie folgt gesteckt sein:
'JP1A und JP1B  jeweils 2-3   -> externer Oszillator (Quarz)
'JP2A und JP2B  jeweils 2-3   -> T3 an RA1  und T2 an RA2

'die beiden nachfolgenden Schalter sind notwendig, wenn das PicKit2 als
'Programmiergerät verwendet wird.
         $LIST  /M_OBJ
         $LIST  OBJ2HEX

'hier wird der Prozessortyp und dessen Spezifikationen definiert
         DEFINE DEVICE 16F628A
         CONFIG CMCFG7                 'interene Komparatoren abschalten, damit
                                       'die Pins als normale Ein-/Ausgänge arbeiten
         CONFIG HS_OSC                 'Wenn ein Quarz verwendet wird
         CONFIG MCLR_INT               'der MCLR-Pin soll als IO-Pin arbeiten
         CONFIG WDT_OFF                'verhindert, dass nach 2,3 Sek ein Reset
                                       'erzeugt wird

         XTAL   4.096

'hier folgen die Variablendeklarationen (8, 16 und ggf. 32 Bit)
         DEFINE    Count0 = AUTO AS BYTE         'Zählervariable (1er-Stelle)
         DEFINE    Count1 = AUTO AS BYTE         'nächste Stelle des Zähler (10er)
         DEFINE    Count2 = AUTO AS BYTE         '100er Stelle
         DEFINE    COUNT3 = AUTO AS BYTE         '1000er Stelle

         DEFINE    Wert = AUTO AS BYTE           'für die Parameterübergabe an
BCD2SEG
         DEFINE    Muster = AUTO AS BYTE         'Segmentmuster

'hier werden die Bitvariablen definiert
```

```
'hier werden die Konstanten definiert

'-----------------------------------------
'ab hier wird Code für den Prozessor erzeugt
'-----------------------------------------
'falls Unterprogramme verwendet werden, werden diese zuerst übersprungen

cold:
         GOTO      start

'hier stehen ggf. die Unterprogramme

'bcd2seg wandelt eine Zahl zwischen 0 und 9 in ein Bitmuster für die
'Siebensegmentanzeige um
bcd2seg:
         IF        Wert=0 THEN LET Muster = %00100001
         IF        Wert=1 THEN LET Muster = %01111101
         IF        Wert=2 THEN LET Muster = %00010011
         IF        Wert=3 THEN LET Muster = %00011001
         IF        Wert=4 THEN LET Muster = %01001101
         IF        Wert=5 THEN LET Muster = %10001001
         IF        Wert=6 THEN LET Muster = %10000001
         IF        Wert=7 THEN LET Muster = %00111101
         IF        Wert=8 THEN LET Muster = %00000001
         IF        Wert=9 THEN LET Muster = %00001001
         RETURN

'hier beginnt das Hauptprogramm
start:
         LOW       RA,1                '?. Digit (T?) ausschalten
         LOW       RA,2                '?. Digit (T?) ausschalten
         LOW       RA,4                '?. Digit (T?) ausschalten
         LET       Count0 = 0          'Zähler mit 0 initialisieren
         LET       Count1 = 0          'auch die übrigen Stellen auf 0 setzen
         LET       Count2 = 0
         LET       Count3 = 0

         TRIS      RB,0                'alle Pins von RB arbeiten als Ausgänge
loop:
         LET       Wert = Count0       'jetzt muss gemultiplext werden
         GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
         OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
         HIGH      RA,0                '1. Digit (T4) einschalten
         WAIT      5                   '5ms warten. Zeit entscheidet über Flackern
der Anzeige
         LOW       RA,0                '1. Digit (T4) ausschalten

         LET       Wert = Count1       'jetzt die nächste Stelle
         GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
         OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
         HIGH      RA,1                '2. Digit (T3) einschalten
         WAIT      5                   '5ms warten
         LOW       RA,1                '2. Digit (T3) ausschalten

         LET       Wert = Count2       'jetzt die 100er-Stelle
         GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
         OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
         HIGH      RA,2                '3. Digit (T2) einschalten
         WAIT      5                   '5ms warten
         LOW       RA,2                '3. Digit (T2) ausschalten
```

```
        LET     Wert = Count3       'jetzt die 1000er-Stelle
        GOSUB   bcd2seg             'Zahl in Wert umwandeln in Seg.muster
        OUTPUT  RB,Muster           'dieses Muster an Port RB anlegen
        HIGH    RA,4                '4. Digit (T1) einschalten
        WAIT    5                   '5ms warten
        LOW     RA,4                '4. Digit (T1) ausschalten


        LET     Count0=Count0 + 1   'Zähler hochzählen lassen
        IF      Count0<10 THEN GOTO loop    'kein Überlauf, keine neue Stelle
notwendig

        LET     Count0=0            '1er-Zähler wieder auf 0 setzen, dafür 10er
erhöhen
        LET     Count1=Count1 + 1
        IF      Count1<10 THEN GOTO loop     '10er Stelle kein Überlauf

        LET     Count1 = 0          '10er-Stelle zurück auf 0
        LET     Count2 = Count2 + 1 '100er-Steller erhöhen
        IF      Count2<10 THEN GOTO loop

        LET     Count2 = 0          '100-Stelle auf 0
        LET     Count3 = Count3 + 1 '1000-er Stelle hochzählen
        IF      Count3<10 THEN GOTO loop

        LET     Count3=0            'wenn hier angekommen, ist der komplette
Zähler
                                    'einmal durchgelaufen

        GOTO    LOOP                'das Ganze läuft nun endlos

        END
```

Schritt 6

Jetzt wird die Start-Taste abgefragt und bei einer steigenden Flanke der Zähler um 1 erhöht. Eine Bitdefinition erleichtert die Tastenabfrage. Solange die Taste nicht gedrückt ist, liegt am entsprechenden Pin Low-Pegel. Dieser Zustand wird in der Variablen Taster1 gespeichert. Drückt man jetzt auf TA1, wechselt der Pegel an RA3 von Low auf High. Es erfolgt noch die Überprüfung der Variablen Taster1. Ist deren Wert 0, dann haben wir einen Pegelwechsel vorliegen und der Zähler wird hochgezählt. Bei diesem Vorgang wird auch Taster1 auf 1 gesetzt, so dass ein weiteres Erkennen eines High-Pegels an diesem Pin nicht nochmal zum Erhöhen des Zählers führt. Erst wenn die Taste wieder losgelassen wurde, wechselt der Wert von Taster1 auf 0. Erst jetzt kann eine Flanke wieder erkannt werden.

```
'*****************************************************************************
'                              Vor- und Rückwärtszähler
'                                  F_R_Counter.BAS
'
'                                 Stand: 24.06.2009
'
'                                    6. Schritt
'
'(c) Ing.Büro Lehmann, Fürstenbergstraße 8a, 77756 Hausach, www.iL-online.de
'*****************************************************************************

'Historie
'Schritt6: jetzt wird der Tastendruck erfasst und bei steigender Flanke an RA3
'          hochgegezählt. Erstmal wird eine Bitvariable definiert die mit dem
'          Eingang RA3 (TA1, Start) verknüpft ist. Zusätzlich eine Merker-
'          Variable (Taster1) die gesetzt wird, sobald eine steigende Flanke
'          entdeckt und der Zähler erhöht wurde. Zurückgesetzt wird sie erst,
'          wenn die Taste wieder losgelassen wurde. 'Schritt5: nun arbeiten auch die
anderen Stellen und werden dabei hoch-
'          gezählt. Die Pause ist kleiner, damit der Zähler schneller läuft
'          Es müssen weitere Variablen für den Zähler verwendet werden (DEFINE)
'          Auch der Multiplexbetrieb wird nun verwendet
'          Allerdings läuft der Zähler nun deutlich schneller, ohne dass man
'          darauf Einfluss hat. Denn ändert man die 5ms beginnt es u.U. zu flimmern
'          Dies ist allerdings nicht so tragisch, da das Endprodukt nicht
automatisch
'          hochzählen soll.
'Schritt4: es wird auf de hintersten Stelle von 0 bis 9 hochgezählt
'          zum ersten Mal wird eine Variable definiert, ein Unterprogamm
'          zur Zeichenumsetzung verwendet und der TRIS-Befehl eingesetzt
'Schritt3: nun soll das Segment blinken
'Schritt2: weil mehr wie ein Segment leuchten, werden die übrigen explizit
'          abgeschaltet.

'Besonderheiten
'Auf der PICUP-Platine müssen die Steckbrücken (Jumper) wie folgt gesteckt sein:
'JP1A und JP1B  jeweils 2-3   -> externer Oszillator (Quarz)
'JP2A und JP2B  jeweils 2-3   -> T3 an RA1  und T2 an RA2

'die beiden nachfolgenden Schalter sind notwendig, wenn das PicKit2 als
'Programmiergerät verwendet wird.
         $LIST  /M_OBJ
         $LIST  OBJ2HEX

'hier wird der Prozessortyp und dessen Spezifikationen definiert
         DEFINE DEVICE 16F628A
         CONFIG CMCFG7                'interene Komparatoren abschalten, damit
                                      'die Pins als normale Ein-/Ausgänge arbeiten
         CONFIG HS_OSC                'Wenn ein Quarz verwendet wird
         CONFIG MCLR_INT              'der MCLR-Pin soll als IO-Pin arbeiten
         CONFIG WDT_OFF               'verhindert, dass nach 2,3 Sek ein Reset
                                      'erzeugt wird

         XTAL   4.096

'hier folgen die Variablendeklarationen (8, 16 und ggf. 32 Bit)
         DEFINE    Count0 = AUTO AS BYTE         'Zählervariable (1er-Stelle)
         DEFINE    Count1 = AUTO AS BYTE         'nächste Stelle des Zähler (10er)
         DEFINE    Count2 = AUTO AS BYTE         '100er Stelle
         DEFINE    COUNT3 = AUTO AS BYTE         '1000er Stelle
```

```
        DEFINE    Wert = AUTO AS BYTE           'für die Parameterübergabe an
BCD2SEG
        DEFINE    Muster = AUTO AS BYTE         'Segmentmuster
        DEFINE    Taster1 = AUTO AS BYTE        'Merkt sich den Zustand der Taste 1

'hier werden die Bitvariablen definiert
        DEFINE    Taste1 = RA,3                 'Taste zum Hochzählen

'hier werden die Konstanten definiert

'-----------------------------------------
'ab hier wird Code für den Prozessor erzeugt
'-----------------------------------------
'falls Unterprogramme verwendet werden, werden diese zuerst übersprungen

cold:
        GOTO      start

'hier stehen ggf. die Unterprogramme

'bcd2seg wandelt eine Zahl zwischen 0 und 9 in ein Bitmuster für die
'Siebensegmentanzeige um
bcd2seg:
        IF        Wert=0 THEN LET Muster = %00100001
        IF        Wert=1 THEN LET Muster = %01111101
        IF        Wert=2 THEN LET Muster = %00010011
        IF        Wert=3 THEN LET Muster = %00011001
        IF        Wert=4 THEN LET Muster = %01001101
        IF        Wert=5 THEN LET Muster = %10001001
        IF        Wert=6 THEN LET Muster = %10000001
        IF        Wert=7 THEN LET Muster = %00111101
        IF        Wert=8 THEN LET Muster = %00000001
        IF        Wert=9 THEN LET Muster = %00001001
        RETURN

'hier beginnt das Hauptprogramm
start:
        LOW       RA,1                '?. Digit (T?) ausschalten
        LOW       RA,2                '?. Digit (T?) ausschalten
        LOW       RA,4                '?. Digit (T?) ausschalten
        LET       Count0 = 0          'Zähler mit 0 initialisieren
        LET       Count1 = 0          'auch die übrigen Stellen auf 0 setzen
        LET       Count2 = 0
        LET       Count3 = 0
        IF        Taste1=0 then LET Taster1=0 ELSE LET Taster1=1

        TRIS      RB,0                'alle Pins von RB arbeiten als Ausgänge
loop:
        LET       Wert = Count0       'jetzt muss gemultiplext werden
        GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
        OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
        HIGH      RA,0                '1. Digit (T4) einschalten
        WAIT      5                   '5ms warten. Zeit entscheidet über Flackern
der Anzeige
        LOW       RA,0                '1. Digit (T4) ausschalten

        LET       Wert = Count1       'jetzt die nächste Stelle
        GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
        OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
        HIGH      RA,1                '2. Digit (T3) einschalten
        WAIT      5                   '5ms warten
```

```
         LOW       RA,1                '2. Digit (T3) ausschalten

         LET       Wert = Count2       'jetzt die 100er-Stelle
         GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
         OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
         HIGH      RA,2                '3. Digit (T2) einschalten
         WAIT      5                   '5ms warten
         LOW       RA,2                '3. Digit (T2) ausschalten

         LET       Wert = Count3       'jetzt die 1000er-Stelle
         GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
         OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
         HIGH      RA,4                '4. Digit (T1) einschalten
         WAIT      5                   '5ms warten
         LOW       RA,4                '4. Digit (T1) ausschalten

'hier erfolgt die Überprüfung von Taster_Start (Taste1). Wird eine steigende Flanke
erkannt
'wird der Zähler erhöht. Das Signal kann auch über den Optokoppler 2 (Klemmen Start)
kommen
'Zuerst wird geprüft, ob die Taste ein 0-Pegel bringt, sonst wäre sie gedrückt
'Eine Merkervariable (Taster1) sorgt dafür, dass wirklich nur die Flanke zählt.
Dieser Merker
'wird durch die Zählroutine gesetzt und wird nur durch einen LOW-Pegel an der Taste
zurückgesetzt
         IF        Taste1=0 THEN GOTO tast_aus
tast_ein:
         IF        Taster1=0 THEN GOTO flanke_da  'Flanke erkannt -> zählen
         GOTO      loop                'Taste auf 1 aber keine Flanke
tast_aus:
         LET       Taster1=0           'Taste nicht gedrückt, Zustand merken und zu
loop
         GOTO      loop

flanke_da:
         LET       Taster1=1           'Merker setzen
         LET       Count0=Count0 + 1   'Zähler hochzählen lassen
         IF        Count0<10 THEN GOTO loop    'kein Überlauf, keine neue Stelle
notwendig

         LET       Count0=0            '1er-Zähler wieder auf 0 setzen, dafür 10er
erhöhen
         LET       Count1=Count1 + 1
         IF        Count1<10 THEN GOTO loop     '10er Stelle kein Überlauf

         LET       Count1 = 0          '10er-Stelle zurück auf 0
         LET       Count2 = Count2 + 1 '100er-Steller erhöhen
         IF        Count2<10 THEN GOTO loop

         LET       Count2 = 0          '100-Stelle auf 0
         LET       Count3 = Count3 + 1 '1000-er Stelle hochzählen
         IF        Count3<10 THEN GOTO loop

         LET       Count3=0            'wenn hier angekommen, ist der komplette
Zähler
                                       'einmal durchgelaufen

         GOTO      LOOP                'das Ganze läuft nun endlos

         END
```

Das 7. Programm beinhaltet nun alle gewünschten Funktionen. Die Tastenabfrage für die Stop-Taste (Taste 2) ist ähnlich der für Taste 1 aufgebaut. Der Abwärtszähler funktioniert nach folgendem Schema:

Zuerst wird die letzte Stelle erniedrigt. Hat die Variable Count0 nun den Wert 255, dann wurde 0 um 1 verringert. In diesem Fall soll aber an dieser Stelle der Wert 9 erscheinen und die nächst höhere Stelle um eins erniedrigt werden. So wird Stelle für Stelle überprüft. Damit ist gewährleistet, dass nach dem Zählerstand 0000 auf 9999 gesprungen wird.

Der Zähler kann nicht nur über die beiden Tasten gesteuert werden, sondern auch über die beiden Impulseingänge START und STOP. Legt man hier ein Rechtecksignal an, dann wird der Zähler entsprechend der steigenden Flanke hoch- bzw. runterzählen.

Hier kann man prüfen, bis zu welcher Frequenz dieser Zähler richtig arbeitet. Denn gemäß dem Programm muss zuerst die komplette Anzeige angesteuert werden bevor die Tastenabfrage erfolgt. Die Anzeigenroutine benötigt aber eine Mindestzeit (hier 4 x 5 ms). Erfolgt der Flankenwechsel an den Impulseingängen schneller, werden manche Impulse nicht mehr erkannt und somit nicht gezählt.

```
'*****************************************************************************
'                              Vor- und Rückwärtszähler
'                                  F_R_Counter.BAS
'
'                                 Stand: 24.06.2009
'
'                                    7. Schritt
'
'(c) Ing.Büro Lehmann, Fürstenbergstraße 8a, 77756 Hausach, www.iL-online.de
'*****************************************************************************

'Historie
'Schritt7: nun wird die Taste 2 (Stop) abgefragt und der Zähler erniedrigt
'Schritt6: jetzt wird der Tastendruck erfasst und bei steigender Flanke an RA3
'          hochgegezählt. Erstmal wird eine Bitvariable definiert die mit dem
'          Eingang RA3 (TA1, Start) verknüpft ist. Zusätzlich eine Merker-
'          Variable (Taster1) die gesetzt wird, sobald eine steigende Flanke
'          entdeckt und der Zähler erhöht wurde. Zurückgesetzt wird sie erst,
'          wenn die Taste wieder losgelassen wurde. 'Schritt5: nun arbeiten auch die
anderen Stellen und werden dabei hoch-
'          gezählt. Die Pause ist kleiner, damit der Zähler schneller läuft
'          Es müssen weitere Variablen für den Zähler verwendet werden (DEFINE)
'          Auch der Multiplexbetrieb wird nun verwendet
'          Allerdings läuft der Zähler nun deutlich schneller, ohne dass man
'          darauf Einfluss hat. Denn ändert man die 5ms beginnt es u.U. zu flimmern
'          Dies ist allerdings nicht so tragisch, da das Endprodukt nicht
automatisch
'          hochzählen soll.
'Schritt4: es wird auf de hintersten Stelle von 0 bis 9 hochgezählt
'          zum ersten Mal wird eine Variable definiert, ein Unterprogamm
'          zur Zeichenumsetzung verwendet und der TRIS-Befehl eingesetzt
'Schritt3: nun soll das Segment blinken
'Schritt2: weil mehr wie ein Segment leuchten, werden die übrigen explizit
'          abgeschaltet.

'Besonderheiten
'Auf der PICUP-Platine müssen die Steckbrücken (Jumper) wie folgt gesteckt sein:
'JP1A und JP1B  jeweils 2-3   -> externer Oszillator (Quarz)
'JP2A und JP2B  jeweils 2-3   -> T3 an RA1  und T2 an RA2

'die beiden nachfolgenden Schalter sind notwendig, wenn das PicKit2 als
'Programmiergerät verwendet wird.
         $LIST  /M_OBJ
         $LIST  OBJ2HEX

'hier wird der Prozessortyp und dessen Spezifikationen definiert
         DEFINE DEVICE 16F628A
         CONFIG CMCFG7                'interene Komparatoren abschalten, damit
                                      'die Pins als normale Ein-/Ausgänge arbeiten
         CONFIG HS_OSC                'Wenn ein Quarz verwendet wird
         CONFIG MCLR_INT              'der MCLR-Pin soll als IO-Pin arbeiten
         CONFIG WDT_OFF               'verhindert, dass nach 2,3 Sek ein Reset
                                      'erzeugt wird

         XTAL   4.096

'hier folgen die Variablendeklarationen (8, 16 und ggf. 32 Bit)
         DEFINE    Count0 = AUTO AS BYTE         'Zählervariable (1er-Stelle)
         DEFINE    Count1 = AUTO AS BYTE         'nächste Stelle des Zähler (10er)
         DEFINE    Count2 = AUTO AS BYTE         '100er Stelle
         DEFINE    COUNT3 = AUTO AS BYTE         '1000er Stelle
```

```
        DEFINE    Wert = AUTO AS BYTE           'für die Parameterübergabe an
BCD2SEG
        DEFINE    Muster = AUTO AS BYTE         'Segmentmuster
        DEFINE    Taster1 = AUTO AS BYTE        'Merkt sich den Zustand der Taste 1
        DEFINE    Taster2 = AUTO AS BYTE        'Merkt sich den Zustand der Stop-
Taste (Taste2)

'hier werden die Bitvariablen definiert
        DEFINE    Taste1 = RA,3                 'Taste zum Hochzählen
        DEFINE    Taste2 = RA,5                 'Taste zum Abwärtszählen

'hier werden die Konstanten definiert

'-----------------------------------------
'ab hier wird Code für den Prozessor erzeugt
'-----------------------------------------
'falls Unterprogramme verwendet werden, werden diese zuerst übersprungen

cold:
        GOTO      start

'hier stehen ggf. die Unterprogramme

'bcd2seg wandelt eine Zahl zwischen 0 und 9 in ein Bitmuster für die
'Siebensegmentanzeige um
bcd2seg:
        IF        Wert=0 THEN LET Muster = %00100001
        IF        Wert=1 THEN LET Muster = %01111101
        IF        Wert=2 THEN LET Muster = %00010011
        IF        Wert=3 THEN LET Muster = %00011001
        IF        Wert=4 THEN LET Muster = %01001101
        IF        Wert=5 THEN LET Muster = %10001001
        IF        Wert=6 THEN LET Muster = %10000001
        IF        Wert=7 THEN LET Muster = %00111101
        IF        Wert=8 THEN LET Muster = %00000001
        IF        Wert=9 THEN LET Muster = %00001001
        RETURN

'hier beginnt das Hauptprogramm
start:
        LOW       RA,1                '?. Digit (T?) ausschalten
        LOW       RA,2                '?. Digit (T?) ausschalten
        LOW       RA,4                '?. Digit (T?) ausschalten
        LET       Count0 = 0          'Zähler mit 0 initialisieren
        LET       Count1 = 0          'auch die übrigen Stellen auf 0 setzen
        LET       Count2 = 0
        LET       Count3 = 0
        IF        Taste1=0 then LET Taster1=0 ELSE LET Taster1=1

        TRIS      RB,0                'alle Pins von RB arbeiten als Ausgänge
loop:
        LET       Wert = Count0       'jetzt muss gemultiplext werden
        GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
        OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
        HIGH      RA,0                '1. Digit (T4) einschalten
        WAIT      5                   '5ms warten. Zeit entscheidet über Flackern
der Anzeige
        LOW       RA,0                '1. Digit (T4) ausschalten

        LET       Wert = Count1       'jetzt die nächste Stelle
```

```
        GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
        OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
        HIGH      RA,1                '2. Digit (T3) einschalten
        WAIT      5                   '5ms warten
        LOW       RA,1                '2. Digit (T3) ausschalten

        LET       Wert = Count2       'jetzt die 100er-Stelle
        GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
        OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
        HIGH      RA,2                '3. Digit (T2) einschalten
        WAIT      5                   '5ms warten
        LOW       RA,2                '3. Digit (T2) ausschalten

        LET       Wert = Count3       'jetzt die 1000er-Stelle
        GOSUB     bcd2seg             'Zahl in Wert umwandeln in Seg.muster
        OUTPUT    RB,Muster           'dieses Muster an Port RB anlegen
        HIGH      RA,4                '4. Digit (T1) einschalten
        WAIT      5                   '5ms warten
        LOW       RA,4                '4. Digit (T1) ausschalten

'hier erfolgt die Überprüfung von Taster_Start (Taste1). Wird eine steigende Flanke
erkannt
'wird der Zähler erhöht. Das Signal kann auch über den Optokoppler 2 (Klemmen Start)
kommen
'Zuerst wird geprüft, ob die Taste ein 0-Pegel bringt, sonst wäre sie gedrückt
'Eine Merkervariable (Taster1) sorgt dafür, dass wirklich nur die Flanke zählt.
Dieser Merker
'wird durch die Zählroutine gesetzt und wird nur durch einen LOW-Pegel an der Taste
zurückgesetzt
        IF        Taste1=0 THEN GOTO tast1_aus
tast1_ein:
        IF        Taster1=0 THEN GOTO flanke_da_h  'Flanke erkannt -> hochzählen
        GOTO      loop                'Taste auf 1 aber keine Flanke
tast1_aus:
        LET       Taster1=0           'Taste nicht gedrückt, Zustand merken und
nächste Taste prüfen

'hier Taste 2 prüfen
        IF        Taste2=0 THEN GOTO tast2_aus
tast2_ein:
        IF        Taster2=0 THEN GOTO flanke_da_l   'Flanke erkannt, abwärtszählen
        GOTO      loop                'Taste 2 noch immer auf 1, somit ignorieren
tast2_aus:
        LET       Taster2=0           'Taste 2 nicht gedrückt, Zustand merken und
zu loop
        GOTO      loop

'hier wird hochgezählt
flanke_da_h:
        LET       Taster1=1           'Merker setzen
        LET       Count0=Count0 + 1   'Zähler hochzählen lassen
        IF        Count0<10 THEN GOTO loop    'kein Überlauf, keine neue Stelle
notwendig

        LET       Count0=0            '1er-Zähler wieder auf 0 setzen, dafür 10er
erhöhen
        LET       Count1=Count1 + 1
        IF        Count1<10 THEN GOTO loop     '10er Stelle kein Überlauf

        LET       Count1 = 0          '10er-Stelle zurück auf 0
        LET       Count2 = Count2 + 1 '100er-Steller erhöhen
```

```
         IF        Count2<10 THEN GOTO loop

         LET       Count2 = 0          '100-Stelle auf 0
         LET       Count3 = Count3 + 1 '1000-er Stelle hochzählen
         IF        Count3<10 THEN GOTO loop

         LET       Count3=0            'wenn hier angekommen, ist der komplette
Zähler
                                       'einmal durchgelaufen

         GOTO      LOOP                'damit das Ganze endlos läuft

'hier wird abwärtsgezählt
flanke_da_1:
         LET       Taster2=1           'Merker setzen
         LET       Count0=Count0 - 1   'Zähler runterzählen lassen
         IF        Count0<>255 THEN GOTO loop    'kein Unterlauflauf, keine neue
Stelle notwendig

         LET       Count0=9            '1er-Zähler wieder auf 9 setzen, dafür 10er
erniedrigen
         LET       Count1=Count1 - 1
         IF        Count1<>255 THEN GOTO loop     '10er Stelle kein Unterlauf

         LET       Count1 = 9          '10er-Stelle zurück auf 9
         LET       Count2 = Count2 - 1 '100er-Steller erniedrigen
         IF        Count2<>255 THEN GOTO loop

         LET       Count2 = 9          '100-Stelle auf 9
         LET       Count3 = Count3 - 1 '1000-er Stelle erniedrigen
         IF        Count3<>255 THEN GOTO loop

         LET       Count3=9            'wenn hier angekommen, ist der komplette
Zähler
                                       'heruntergelaufen

         GOTO      LOOP                'damit das Ganze endlos läuft

         END
```