Drehimpulsgeber

(Dekodierung mit PIC-Mikrocontroller)

Autor: Buchgeher Stefan

Letzte Bearbeitung: 2. März 2004

Inhaltsverzeichnis

1. Grundlegendes zum Drehimpulsgeber

Bild 1: Bauformen von Drehimpulsgebern

Ein Drehimpulsgeber ist ein Bauteil, welcher beim Drehen Impulse erzeugt. Diese Impulse sind von der Drehrichtung abhängig. Weiters besitzt ein Drehimpulsgeber meist noch eine Druck-Taster-Funktion. Mit einem Drehimpulsgeber kann daher eine einfache Werteingabe oder Menüauswahl erfolgen. Durch Drehen kann ein Wert oder ein Menü ausgewählt werden und mit einem Druck auf die Achse erfolgt die Bestätigung dieses Wertes oder des Menüs. Drehimpulsgeber sind bei Conrad in zwei Ausführungen erhältlich (siehe Bild 1: liegend, Bestell-Nr. 705586 oder stehend, Bestell-Nr. 705594).

Die hier gezeigten Drehimpulsgeber besitzen 5 Pins (2 Pins für die Taster-Funktion und 3 Pins für die „eigentlichen“ Drehimpulse).
Dreht man die Achse, so entsteht an 2 der 3 Pins ein Taktmuster. Diese beiden Pins dienen dabei als Schalter, welche je nach Stellung der Achse entweder Öffnen oder Schließen. Der dritte (mittlere) Pin ist der gemeinsame Bezugspin und wird daher extern mit Masse verbunden.

Die Drehachse besitzt pro Umdrehung 30 leichte Rastpositionen. Pro Umdrehung erzeugen beide „Schalter“ 15 Rechteckimpulse, welche um etwa 6 ms zueinander verschoben sind. Das folgende Bild zeigt das Taktmuster der beiden internen „Schalter“ bei einer Drehung gegen den Uhrzeigersinn (CCW = Counter Clock Wise).

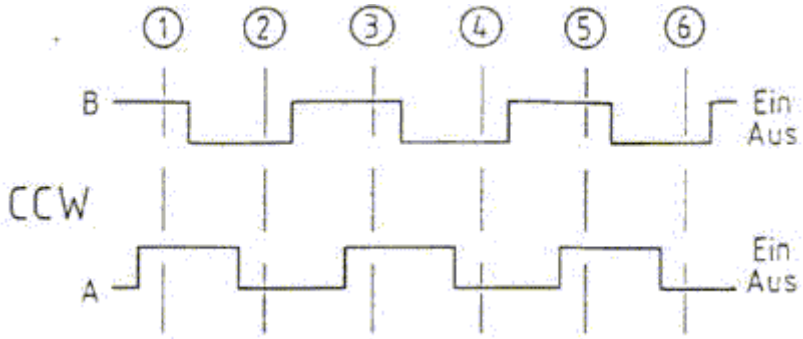


Bild 2: Taktmuster bei einer Drehung gegen den Uhrzeigersinn

Die eingekreisten Ziffern deuten die Rasterstellung an, in der sich die Achse befindet. Die Bezeichnungen „Ein“ bzw. „Aus“ bedeuten dass die inneren „Schalter“ Pin A bzw. Pin B gegenüber dem Bezugspin (Masse) entweder geschlossen oder geöffnet sind.

Das folgende Bild zeigt das Taktmuster bei einer Drehung im Uhrzeigersinn. (Hier liest man das obige Bild einfach nur von rechts nach links)

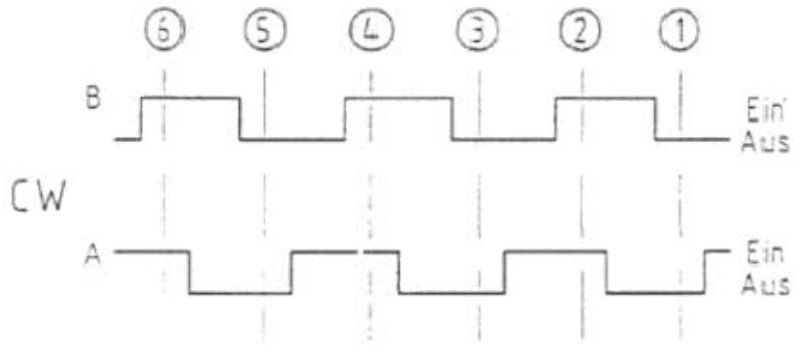


Bild 3: Taktmuster bei einer Drehung im Uhrzeigersinn

Bei genauer Betrachtung der beiden Bilder fällt auf, dass es eine eindeutige Unterscheidung zwischen einer Drehung gegen den Uhrzeigersinn (Linksdrehung) und einer Drehung im Uhrzeigersinn (Rechtsdrehung) gibt:

- Bei einer Drehung gegen den Uhrzeigersinn (Bild 2) ändert der Drehimpulsgeber zuerst Pin B. Ca. 6 ms später Pin A.
- Bei einer Drehung im Uhrzeigersinn (Bild 3) ändert der Drehimpulsgeber zuerst Pin A. Ca. 6 ms später Pin B.

Weiters gilt:

- Befindet sich der Drehimpulsgeber in einer Rasterposition, so haben die Pins A und B den selben Pegel (entweder beide Low-Pegel oder beide High-Pegel, siehe Bilder 2 und 3).

2. Hardware

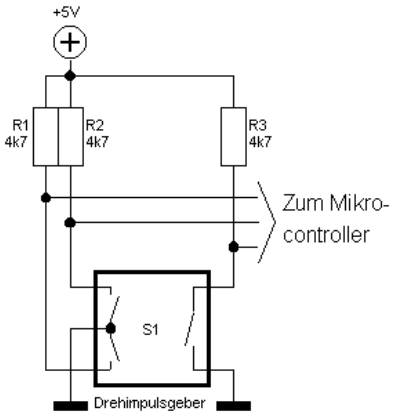


Bild 4: Hardware zur Dekodierung

Für die Auswertung (Dekodierung) werden neben dem Drehimpulsgeber (S1) nur 3 Pull-Up-Widerstände (je 4k7) und 3 Portpins eines Mikrocontrollers benötigt. Wird die Taster-Funktion (der Achse) nicht benötigt, so kann der Widerstand R3 entfallen. (Es werden dann nur 2 Pull-Up-Widerstände und 2 freie Portpins eines Mikrocontrollers verwendet). Welche Mikrocontroller-Familie verwendet wird ist eigentlich nebensächlich. Hier beschränke ich mich aber zurzeit auf die PIC-Mikrocontroller-Familie, da ich für andere Familien keine Entwicklungsumgebung besitze!

3. Software

Auf die Taster-Funktion wird bei der folgende Software und der Softwarebeschreibung nicht eingegangen, da dieser wie ein ganz normaler Taster funktioniert.

3.1. Das Dekodierverfahren

Die beiden Eingänge des Drehimpulsgeber werden zyklisch (ca. alle 4 ms) vom Unterprogramm INKRROUTINE abgefragt. Als Zeitbasis für den 4-ms-Takt dient der Timer-0-Interrupt. Dieser kann je nach Anwendung entweder so eingestellt werden, dass er alle 4 ms einen Interrupt auslöst, oder er wird so eingestellt dass er z. B. alle 500µs oder alle 1ms einen Interrupt erzeugt und ein Zählregister zählt die notwendige Anzahl an Interruptaufrufen bis zu einer Zeitbasis von 4 ms. Hier in dieser Dokumentation wird der Einfachheit halber die erstere Variante verwendet.
Die Aufgabe des Unterprogramms INKRROUTINE besteht darin eine Drehung des Drehimpulsgebers aufgrund der im Abschnitt 1 (Grundlegendes zum Drehimpulsgeber) gewonnen Erkenntnisse zu dekodieren und je nach Richtung entweder das Flag INKRLINKS oder INKRRECHTS im Register INKRSTATUS zu setzen. Weiters beinhaltet das Register INKRSTATUS noch den Zustand der Eingänge des Drehimpulsgebers 4 ms vor dem Aufruf des Unterprogramms INKRROUTINE. Mit Hilfe des aktuellen Zustands und des Zustands vor 4 ms kann nun erkannt werden, ob der Drehimpulsgeber gedreht wurde und in welche Richtung diese Drehung erfolgte.
Beim Demonstrationsbeispiel (siehe Abschnitt 4) erfolgt der Aufruf des Unterprogramms INKRROUTINE in der ISR. Man kann sagen dass diese zyklische Tätigkeit im Hintergrund abläuft. Das Hauptprogramm prüft nun ständig die beiden Flags INKRLINKS und INKRRECHTS des Register INKRSTATUS. Ist eines der beiden Flags gesetzt, so wurde eine Drehung in diese Richtung erkannt und die gewünschten Aktionen können entweder mit einem Unterprogramm oder direkt im Hauptprogramm eingeleitet werden. Logischerweise können die Flags INKRLINKS und INKRRECHTS nie gleichzeitig gesetzt sein. Im Demonstrationsbeispiel (siehe Abschnitt 4) wandert eine LED einer LED-Reihe beim Drehen abhängig von der Drehrichtung entweder nach links oder nach rechts.

3.2. Benötigte Register und Maske

Für die Dekodierung einer Drehung des Drehimpulsgebers werden neben einigen internen Register (SFR, **S**pezielle **F**unktions-**R**egister) noch folgende Register benötigt:

- INKRSTATUS: (das schon angesprochene zentrale Register der Drehimpulsgeberdekodierung, siehe auch Abschnitt 3.4)
- TEMP1: Hilfsregister zum Zwischenspeichern der aktuellen Pegeln am Drehimpulsgeber
- TEMP2: Hilfsregister zur Dekodierung

Die Maske MASKEINKR wird zum Ausmaskieren der beiden Eingänge des Drehimpulsgebers am Port A benötigt

3.3. Initialisierung (Unterprogramm „INIT“)

Die Portpins an die der Drehimpulsgeber angeschlossen ist müssen als Eingang definiert werden. Welcher Port (A, B, C, D...) verwendet wird ist nicht von Bedeutung. Wichtig ist nur, dass die beiden Eingänge am selben Port angeschlossen sind. Es ist

auch möglich dass sich die Eingänge an einem Port nicht „nebeneinander“ befinden. (Eingang A kann zum Beispiel am Portpin RA2 angeschlossen werden, während der Eingang B am Portpin RA5) angeschlossen ist.

In der Initialisierungsroutine muss für den erste Aufruf des Unterprogramms INKRROUTINE der aktuelle Zustand des Ports (an dem der Drehimpulsgeber angeschlossen ist) gelesen, die relevanten Bits ausmaskiert und in das Register INKRSTATUS kopiert werden.

Der folgende Programmausschnitt zeigt eine mögliche Initialisierungsroutine. Der Drehimpulsgeber ist hier am Port A an den Bits 3 und 4 angeschlossen. Die schwarz hervorgehobenen Stellen sind für die Dekodierung des Drehimpulsgebers notwendig.

```
INIT        clrf  TMR0             ;Timer0 auf 0 voreingestellt

            bsf   STAT,RP0         ;Registerseite 1
            movlw b'00000011'      ;interner Takt, Vorteiler = 16 an TMR0
            movwf OPTREG
            movlw MASKEINKR        ;Port A: Bits 3 und 4: Eingang
            movwf TRISA            ;        Bits 0 bis 2: Ausgang (unbenutzt)
            movlw b'00000000'      ;Port B: Ausgang
            movwf TRISB

            bcf   STAT,RP0         ;Registerseite 0

            movf  PORTA,w          ;Den aktuellen Zustand von Port A einlesen
            andlw MASKEINKR        ;  die Bits des Inkrementalgebers
            movwf INKRSTATUS       ;  ausmaskieren und in INKRSTATUS sichern

            movlw b'00001000'      ;Zu Beginn soll eine der mittleren LEDs
            movwf PORTB            ;  leuchten

            return
```

3.4. Unterprogramm „INKRROUTINE“

Das Unterprogramm „INKRROUTINE“ ist für die Dekodierung die wichtigste Komponente .

Diese Routine wird ca. alle 4 ms aufgerufen

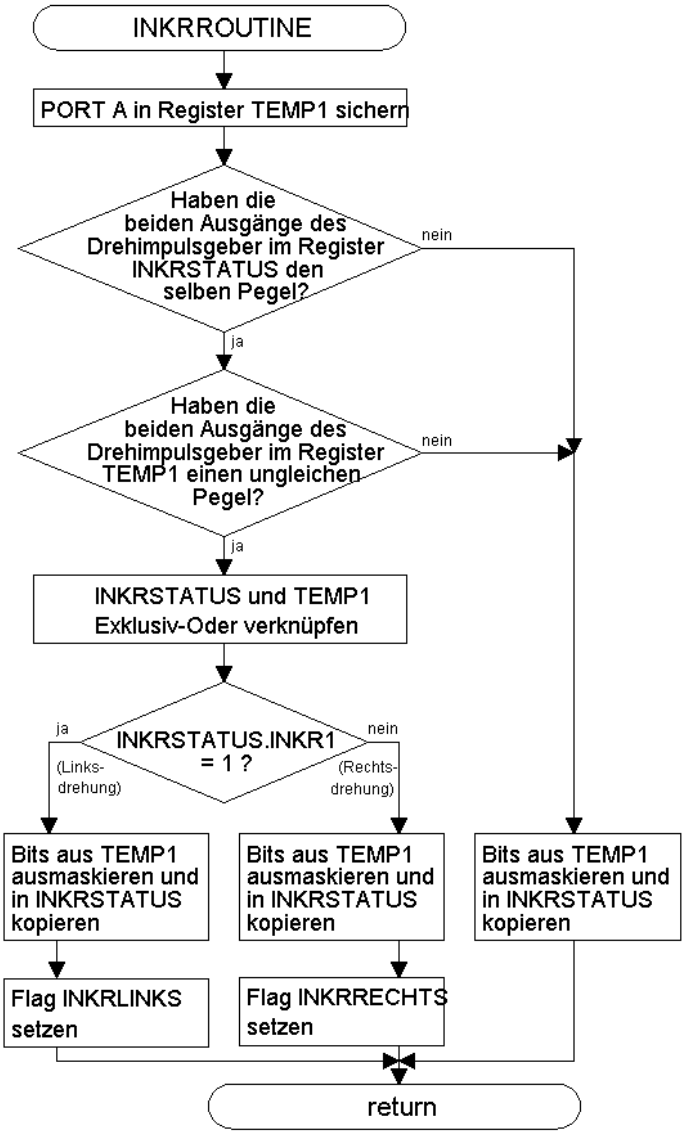
Aufgabe:
Ermittelt ob und in welche Richtung der Drehimpulsgeber (= Inkrementalgeber) gedreht wurde. Das Ergebnis befindet sich dann im Register INKRSTATUS. Dieses Register besitzt die folgenden Bits (Flags):

- INKR1ALT, INKR2ALT: Beinhalten den Zustand des Drehimpulsgeber für den nächsten Aufruf dieses Unterprogramms.
- INKRLINKS: Wird vom Unterprogramm „INKRROUTINE“ gesetzt, wenn eine Drehung nach links des Drehimpulsgebers dekodiert wurde.
- INKRRECHTS: Wird vom Unterprogramm „INKRROUTINE“ gesetzt, wenn eine Drehung nach rechts des Drehimpulsgebers dekodiert wurde.

Allgemeines:

- Der Inkrementalgeber ist (so wie auch bei Tasten) am Port mit zwei Pull-Up-Widerstand angeschlossen.
- Die Eingänge des Drehimpulsgebers müssen hier am selben Port angeschlossen sein, welche Bits verwendet werden ist jedoch beliebig und müssen nicht "nebeneinander" liegen. (Eingang A kann z.B. am Port RA2 liegen, während der Eingang B dem Portpin RA5 zugeordnet ist)

Vorgehensweise:



- Port A einlesen und im Register TEMP1 sichern.
- Sind die beiden Bits des "alten" Zustandes ungleich, zum Ende des Unterprogramms springen. (Das Register TEMP2 dient zur Überprüfung, ob die Bits gleich (bzw. ungleich) sind
- Sind die beiden Bits des "alten" Zustandes gleich, testen, ob die beiden Bits des „neuen“ Zustands ungleich sind. (Auch hier dient das Register TEMP2 zur Überprüfung, ob die Bits gleich (bzw. ungleich) sind
- Sind die beiden Bedingungen erfüllt (Bits des alten Zustands gleich und Bits des neuen Zustands ungleich) so wurde der Drehimpulsgeber verdreht. Als nächstes gilt es herauszufinden in welche Richtung der Drehimpulsgeber gedreht wurde. Dazu den alten Zustand (Register INKRSTATUS) mit dem neuen Zustand (TEMP1) Exklusiv-Oder-Verknüpfen. Je nachdem welches der beiden Bits nun gesetzt ist erfolgte entweder eine Drehung nach links oder eine Drehung nach rechts (Anmerkung: Beide Bits können nicht gesetzt sein. Dies wurde ja im vorgehenden Schritt überprüft!)
- Vom Register TEMP1 nur die Eingänge an denen der Drehimpulsgeber angeschlossen ist ausmaskieren und in das Register INKRSTATUS kopieren und je nach ermittelter Drehrichtung das Flag INKRLINKS oder INKRRECHTS setzen.

Anmerkung:

Die temporären Register TEMP1 und TEMP2 werden hier nur zum Zwischenspeichern (TEMP1) bzw. als Hilfsregister (TEMP2) benötigt. Sie können daher auch woanders verwendet werden.

Hier das Unterprogramm:

```
INKRROUTINE movf  PORTA,w     ;PORT A in das Hilfsregister TEMP1 kopieren
            movwf TEMP1

            ;Testen ob die beiden Bits des "alten" Zustands gleich sind
            clrf  TEMP2
            btfsc INKRSTATUS,INKR2ALT
            bsf   TEMP2,INKR1ALT

            movf  INKRSTATUS,w
            xorwf TEMP2,f

            btfsc TEMP2,INKR1ALT
            goto  UPINKRWEITER ;INKR1 ist ungleich zu INKR2 -> folgende
                               ;  Befehle ueberspringen

            ;Testen, ob die beiden Bits des "neuen" Zustands ungleich sind
            clrf  TEMP2
            btfsc TEMP1,INKR2
            bsf   TEMP2,INKR1

            movf  TEMP1,w
            xorwf TEMP2,f
            btfss TEMP2,INKR1
            goto  UPINKRWEITER ;INKR1 ist gleich zu INKR2 -> folgende
                               ;  Befehle ueberspringen

            ;Ermittlung in welche Richtung gedreht wurde
            movf  TEMP1,w     ;TEMP1 und INKRSTATUS
            xorwf INKRSTATUS,f ;  Exklusiv-Oder verknuepfen
            btfss INKRSTATUS,INKR1
            goto  UPINKRRE

UPINKRLI    movf  TEMP1,w     ;Bei einer ermittelten Linksdrehung die
            andlw MASKEINKR   ;  Eingaenge an denen der Drehimpulsgeber
            movwf INKRSTATUS  ; angeschlossen ist ausmaskieren und in das
                              ;  Register INKRSTATUS kopieren
            bsf   INKRSTATUS,INKRLINKS ;  Flag INKRLINKS setzen
            goto  UPINKRENDE

UPINKRRE    movf  TEMP1,w     ;Bei einer ermittelten Rechtsdrehung die
            andlw MASKEINKR   ;  Eingaenge an denen der Drehimpulsgeber
            movwf INKRSTATUS  ; angeschlossen ist ausmaskieren und in das
                              ;  Register INKRSTATUS kopieren
            bsf   INKRSTATUS,INKRRECHTS   ;  Flag INKRRECHTS setzen
            goto  UPINKRENDE

            ;Bei Nichterfuellung der ersten beiden Bedingungen (Bits des alten
Zustands
            ;gleich und Bits des neuen Zustands ungleich)
UPINKRWEITER movf TEMP1,w     ;Eingaenge an denen der Drehimpulsgeber
            andlw MASKEINKR   ;  angeschlossen ist ausmaskieren und in das
            movwf INKRSTATUS  ;  Register INKRSTATUS kopieren

UPINKRENDE  return
```

4. Demonstrationsbeispiel

Das folgende Beispiel dient nur zur Demonstration der Auswertesoftware. Sie zeigt eine mögliche Einbindung des für die Dekodierung wichtigen Unterprogramms „INKRROUTINE“.

4.1. Hardware

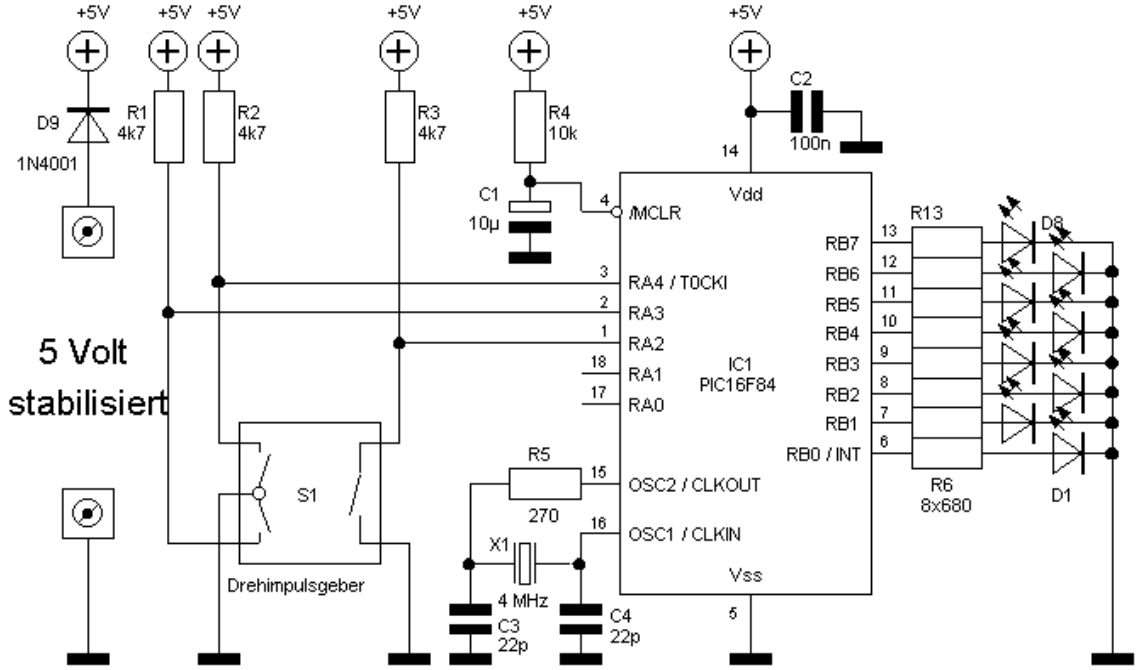


Bild 5: Schaltung der Demonstration des Drehimpulsgebers

Die Drehrichtung des Drehimpulsgeber (S1) wird hier vom PIC-Mikrocontroller PIC16F84 (IC1) an den Portpins RA3 und RA4 mit je einem Pull-Up-Widerstand (R1 und R2) erfasst und ausgewertet. Die Taster-Funktion der Achse wird mit dem Pull-Up-Widerstand R3 am Portpin RA2 angeschlossen (In der Software wird diese Taster–Funktion allerdings noch nicht ausgewertet!).

Die Drehbewegung des Drehimpulsgebers wird optisch mit Leuchtdioden (D1 bis D8 mit den Vorwiderständen R6 bis R13) am Port B angezeigt. Zu Beginn leuchtet eine der mittleren Leuchtdiode. Durch Drehen am Drehimpulsgeber wandert dieser Leuchtpunkt in die jeweilige Drehrichtung.

Für die Takterzeugung dient eine Standardbeschaltung bestehend aus einem 4-MHz-Quarz (X1), zwei Kondensatoren (C3, C4) und einem Widerstand (R5).
Das RC-Glied (R4, C1) erzeugt einen definierten Reset beim Anlegen der Betriebsspannung.

Die Diode D9 verhindert eine Zerstörung des Mikrocontrollers bei einer Falschpolung der Betriebsspannung. Als Betriebsspannung muss eine stabile 5-V-Spannungsquelle verwendet werden.

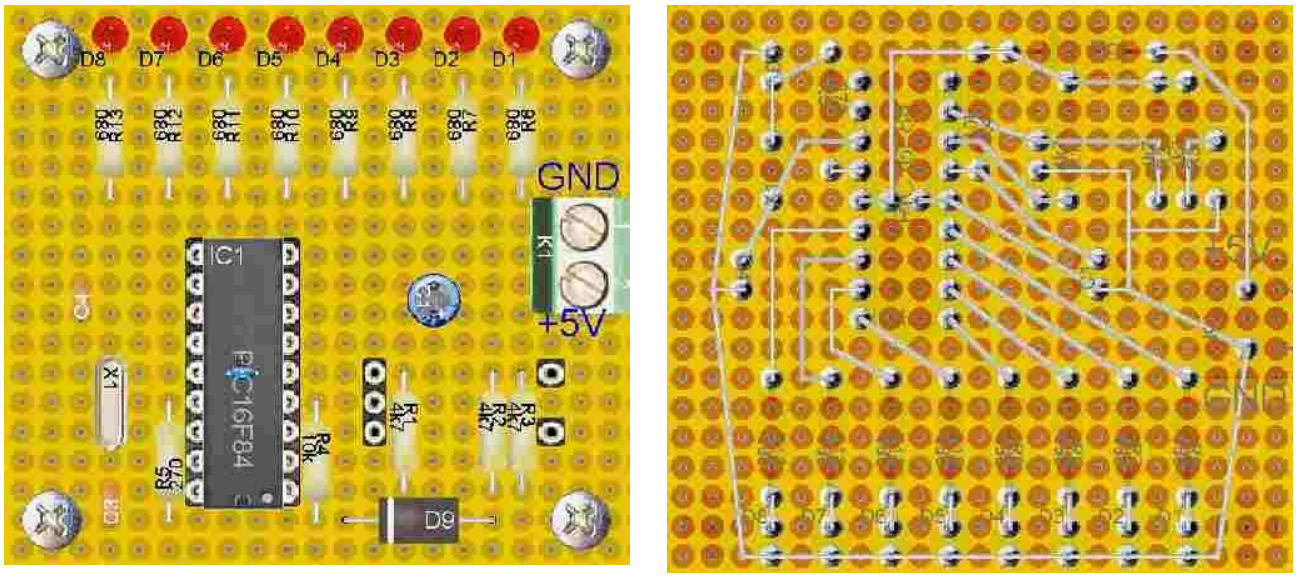


Bild 6: Bauteil- und Lötseite einer möglichen Lochrasterplatine

4.2. Software

```
;*****************************************************************************************
;** Testprogramm zur Werteingabe mit einem Drehimpulsgeber (Inkrementalgeber)            **
;**                                                                                      **
;** Hardware: + Prozessor: PIC16F84, Takt (4 MHz, Genauigkeit ist nicht so wichtig!)     **
;**            + Inkrementalgeber z.B.: Conrad Best.Nr.: 705594 (stehend                 **
;**                                                      705586 (liegend)                **
;**           + 2 Pull-Up-Widerstaende (je 4k7)                                          **
;**           + 1 Pull-Up-Widerstand (4k7) fuer die Taster-Funktion der Achse (Diese Funktion **
;**             wird hier jedoch nicht ausgewertet!)                                     **
;**                                                                                      **
;** Entwickler: Buchgeher Stefan                                                         **
;** Entwicklungsbeginn der Software: 31. Juli 2003                                       **
;** Funktionsfaehig seit: 31. Juli 2003                                                  **
;** Letzte Bearbeitung: 2. Maerz 2004                                                    **
;*****************************************************************************************


               List    p=PIC16F84

;******************************** Register (in Registerbank 0) ********************************
TMR0                    equ     1               ;Timer0-Register
STAT                    equ     3               ;Statusregister
PORTA                   equ     5               ;PortA-Register
PORTB                   equ     6               ;PortB-Register
INTCON                  equ     0B              ;Interrupt-Register


;******************************** Register (in Registerbank 1) ********************************
OPTREG                  equ     1               ;OPTION-Register
TRISA                   equ     5               ;Richtungsregister PortA
TRISB                   equ     6               ;Richtungsregister PortB


;******************************** Eigene Register (in Registerbank 0) *************************
INKRSTATUS              equ     10              ;Statusregister zur Dekodierung des
                                                ;  Drehimpulsgebers

TEMP1                   equ     11              ;Allgemeines Hilfsregister 1
TEMP2                   equ     12              ;Allgemeines Hilfsregister 2

ISR_STAT_TEMP           equ     13              ;Zwischenspeicher des Statusregister der ISR
ISR_w_TEMP              equ     14              ;Zwischenspeicher des Arbeitsregister der ISR


;******************************** Bits in Registern der Registerbank 0 ************************
;Register STAT
```

```
C                       equ     0               ;Carrybit
RP0                     equ     5               ;Seitenauswahlbit 0

;Register INTCON
T0IF                    equ     2               ;Timer0-Interruptflag


;******************************** Bits in den eigenen Registern der Registerbank 0 ************
;Port A
INKR1                   equ     3
INKR2                   equ     4

;INKRSTATUS
INKRLINKS               equ     6
INKRRECHTS              equ     7
INKR1ALT                equ     3               ;Achtung: muessen dieselben Bitzuordnungen
INKR2ALT                equ     4               ;  wie am Port A besitzen


;******************************** Ziele der Registeroperationen ******************************
w                       equ     0               ;w ist Zielregister
f                       equ     1               ;f ist Zielregister


;******************************** Masken *****************************************************
MASKEINKR               equ     b'11000'        ;zum Ausmaskieren der Inkrementalgeber-
                                                ;  Eingaenge


;***************************** Konfigurations-Bits *******************************************
_lp_osc         equ     h'3FFC'
_xt_osc         equ     h'3FFD'
_hs_osc         equ     h'3FFE'
_rc_osc         equ     h'3FFF'

_wdt_off        equ     h'3FFB'
_wdt_on         equ     h'3FFF'

_pwrt_off       equ     h'3FFF'
_pwrt_on        equ     h'3FF7'

_cp_off         equ     h'3FFF'
_cp_on          equ     h'000F'

        __config        _hs_osc & _wdt_off & _pwrt_off & _cp_off

                ORG     0x000
                goto    Beginn
                ORG     0x004
                goto    ISR


;***************************** ISR - Timer0 **************************************************

;*********************************************************************************************
;** Interrupt Service Routine:                                                              **
;** Diese ISR wird ca. alle 4 ms (genauer: alle 4,096 ms) aufgerufen. Die Genauigkeit dieser **
;** Zeit ist hier allerdings nicht so wichtig.                                              **
;**                                                                                         **
;** Aufgabe:                                                                                **
;**     + w-Register (=Arbeitsregister) und Status-Register zwischenspeichern.              **
;**     + Unterprogramm INKRROUTINE aufrufen. Dieses Unterprogramm ist fuer die Auswertung  **
;**       (Dekodierung) der Drehimpulsgebers (Inkrementalgeber) zustaendig                  **
;**     + Das Timer-Interrupt-Flag T0IF wieder löschen                                      **
;**     + Das w-Register und das Status-Register wiederherstellen.                          **
;*********************************************************************************************
ISR
PUSH            movwf   ISR_w_TEMP              ;w-Register retten
                swapf   STAT,w                  ;Statusregister
                movwf   ISR_STAT_TEMP           ;  retten

                ;Beginn der eigentlichen ISR-routine
                call    INKRROUTINE
                ;Ende der eigentlichen ISR-routine

                bcf     INTCON,T0IF             ;T0-Interruptflag loeschen

POP             swapf   ISR_STAT_TEMP,w         ;Status-Register
                movwf   STAT                    ;  und
```

```
                swapf   ISR_w_TEMP,f          ;  w-Register
                swapf   ISR_w_TEMP,w          ;  wieder herstellen

                retfie


;****************************** Unterprogramme *************************************************

;***********************************************************************************************
;** Initialisierung des Prozessor:                                                            **
;**     + TMR0-Vorteiler mit 16 (-> Aufruf der Timer0-ISR alle 4,096 ms)                      **
;**     + Ports: Port A: RA3 und RA4: Eingaenge (Inkrementalgeber)                            **
;**                      RA0 bis RA2: Ausgaenge (unbenutzt)                                   **
;**              Port B: Ausgaenge                                                            **
;**     + Aktuellen Zustand am Port A einlesen, nur die notwendigen Bits ausmaskieren und in  **
;**       das Register INKRSTATUS kopieren                                                    **
;**     + Die mittlere LED am Port B aktivieren                                               **
;***********************************************************************************************
INIT            clrf    TMR0                  ;Timer0 auf 0 voreingestellt

                bsf     STAT,RP0              ;Registerseite 1
                movlw   b'00000011'           ;interner Takt, Vorteiler = 16 an TMR0
                movwf   OPTREG
                movlw   MASKEINKR             ;Port A: Bits 3 und 4: Eingang
                movwf   TRISA                 ;        Bits 0 bis 2: Ausgang (unbenutzt)
                movlw   b'00000000'           ;Port B: Ausgang
                movwf   TRISB

                bcf     STAT,RP0              ;Registerseite 0

                movf    PORTA,w               ;Den aktuellen Zustand von Port A einlesen
                andlw   MASKEINKR             ;... die Bits des Inkrementalgebers
                movwf   INKRSTATUS            ;... ausmaskieren und in INKRSTATUS sichern

                movlw   b'00001000'           ;Zu Beginn soll eine der mittleren LEDs
                movwf   PORTB                 ;  leuchten

                return


;***********************************************************************************************
;** INKRROUTINE                                                                               **
;** Diese Routine wird ca. alle 4 ms aufgerufen                                               **
;**                                                                                           **
;** Aufgabe: Ermittelt ob und in welche Richtung der Drehimpulsgeber (= Inkrementalgeber)     **
;**     gedreht wurde. Das Ergebnis befindet sich dann im Register INKRSTATUS. Dieses         **
;**     Register besitzt die folgenden Bits (Flags):                                          **
;**       + INKR1ALT, INKR2ALT: Beinhalten den Zustand des Drehimpulsgeber fuer den naechsten **
;**                             Aufruf dieses Unterprogramms.                                 **
;**       + INKRLINKS:          Wird von diesem Unterprogramm gesetzt, wenn eine Drehung nach  **
;**                             links des Drehimpulsgebers dekodiert wurde.                   **
;**       + INKRRECHTS:         Wird von diesem Unterprogramm gesetzt, wenn eine Drehung nach  **
;**                             rechts des Drehimpulsgebers dekodiert wurde.                  **
;**                                                                                           **
;** Allgemeines: Der Inkrementalgeber ist (so wie auch bei Tasten) am Port A mit zwei Pull-   **
;**     Up-Widerstand angeschlossen.                                                          **
;**     Die Eingaenge des Drehimpulsgebers muessen hier am selben Port angeschlossen sein,    **
;**      welche Bits verwendet werden ist jedoch beliebig und muessen nicht "nebeneinander"   **
;**      liegen. (Eingang A kann z.B. am Port RA2 liegen, waehrend der Eingang B dem Portpin  **
;**      RA5 zugeordnet ist)                                                                  **
;**                                                                                           **
;** Vorgehensweise:                                                                           **
;**     + Port A einlesen und im Register TEMP1 sichern.                                      **
;**     + Sind die beiden Bits des "alten" Zustandes ungleich, zum Ende des Unterprogramms    **
;**       springen. (Das Register TEMP2 dient zur Ueberpruefung, ob die Bits gleich (bzw. un- **
;**       gleich) sind                                                                        **
;**     + Sind die beiden Bits des "alten" Zustandes gleich, testen, ob die beiden Bits des   **
;**       "neuen" Zustands ungleich sind. (Auch hier dient das Register TEMP2 zur Ueber-      **
;**       pruefung, ob die Bits gleich (bzw. ungleich) sind                                   **
;**     + Sind die beiden Bedingungen erfuellt (Bits des alten Zustands gleich und Bits des   **
;**       neuen Zustands ungleich) so wurde der Drehimpulsgeber verdreht. Als naechstes gilt  **
;**       es herauszufinden in welche Richtung der Drehimpulsgeber gedreht wurde. Dazu den    **
;**       alten Zustand (Register INKRSTATUS) mit dem neuen Zustand (TEMP1) Exklusiv-Oder-    **
;**       verknuepfen. Je nachdem welches der beiden Bits nun gesetzt ist erfolgte entweder   **
;**       eine Drehung nach links oder eine Drehung nach rechts (Anmerkung: Beide Bits        **
;**       koennen nicht gesetzt sein. Dies wurde ja im vorgehenden Schritt ueberprueft!)      **
;**     + Vom Register TEMP1 nur die Eingaenge an denen der Drehimpulsgeber angeschlossen ist **
;**       ausmaskieren und in das Register INKRSTATUS kopieren und je nach ermittelter Dreh-  **
;**       richtung das Flag INKRLINKS oder INKRRECHTS setzen.                                 **
```

```
;**                                                                                          **
;** Anmerkung: Die temporaeren Register TEMP1 und TEMP2 werden hier nur zum Zwischenspeichern **
;**    (TEMP1) bzw. als Hilfsregister (TEMP2) benoetigt. Sie koennen daher auch woanders     **
;**    verwendet werden.                                                                     **
;**********************************************************************************************
INKRROUTINE    movf   PORTA,w              ;PORT A in das Hilfsregister TEMP1 kopieren
               movwf  TEMP1

               ;Testen ob die beiden Bits des "alten" Zustands gleich sind
               clrf   TEMP2
               btfsc  INKRSTATUS,INKR2ALT
               bsf    TEMP2,INKR1ALT

               movf   INKRSTATUS,w
               xorwf  TEMP2,f

               btfsc  TEMP2,INKR1ALT
               goto   UPINKRWEITER         ;INKR1 ist ungleich zu INKR2 -> folgende
                                           ;  Befehle ueberspringen

               ;Testen, ob die beiden Bits des "neuen" Zustands ungleich sind
               clrf   TEMP2
               btfsc  TEMP1,INKR2
               bsf    TEMP2,INKR1

               movf   TEMP1,w
               xorwf  TEMP2,f
               btfss  TEMP2,INKR1
               goto   UPINKRWEITER         ;INKR1 ist gleich zu INKR2 -> folgende
                                           ;  Befehle ueberspringen

               ;Ermittlung in welche Richtung gedreht wurde
               movf   TEMP1,w              ;TEMP1 und INKRSTATUS
               xorwf  INKRSTATUS,f         ;  Exklusiv-Oder verknuepfen
               btfss  INKRSTATUS,INKR1
               goto   UPINKRRE

UPINKRLI       movf   TEMP1,w              ;Bei einer ermittelten Linksdrehung die
               andlw  MASKEINKR            ;  Eingaenge an denen der Drehimpulsgeber
               movwf  INKRSTATUS           ;  angeschlossen ist ausmaskieren und in das
                                           ;  Register INKRSTATUS kopieren
               bsf    INKRSTATUS,INKRLINKS ;  Flag INKRLINKS setzen
               goto   UPINKRENDE

UPINKRRE       movf   TEMP1,w              ;Bei einer ermittelten Rechtsdrehung die
               andlw  MASKEINKR            ;  Eingaenge an denen der Drehimpulsgeber
               movwf  INKRSTATUS           ;  angeschlossen ist ausmaskieren und in das
                                           ;  Register INKRSTATUS kopieren
               bsf    INKRSTATUS,INKRRECHTS ;  Flag INKRRECHTS setzen
               goto   UPINKRENDE

               ;Bei Nichterfuellung der ersten beiden Bedingungen (Bits des alten Zustands
               ;gleich und Bits des neuen Zustands ungleich)
UPINKRWEITER   movf   TEMP1,w              ;Eingaenge an denen der Drehimpulsgeber
               andlw  MASKEINKR            ;  angeschlossen ist ausmaskieren und in das
               movwf  INKRSTATUS           ;  Register INKRSTATUS kopieren

UPINKRENDE     return


;**********************************************************************************************
;** INKRLINKSDREHUNG                                                                         **
;** Diese Routine wird nach einer Drehung nach links des Inkrementalgebers aufgerufen        **
;**                                                                                          **
;** Aufgabe:                                                                                 **
;** Das am Port B gesetzte Bit (die am Port B leuchtende LED) um eine Position nach links    **
;** schieben. War vor dem Aufruf dieses Unterprogramms das Bit 0 (LED 0 leuchtete) gesetzt,  **
;** so ist nun nicht Bit 7 sondern das Carry-Flag gesetzt (Ueberlauf). -> durch ein weiteres **
;** nach links schieben wird die LED 7 bzw. das Bit 7 von Port B gesetzt.                    **
;** Zum Schluss das Anforderungsflag loeschen                                                **
;**********************************************************************************************
INKRLINKSDREHUNG
               bcf    STAT,C               ;Carry-Flag loeschen
               rlf    PORTB,f              ;das gesetzte Bit (die leuchtende LED) um eine
                                           ;  Position nach links schieben

               btfsc  STAT,C               ;Bei einem Ueberlauf (Carry-Flag gesetzt) nach
               rlf    PORTB,f              ;  einmal nach links schieben. Nun leuchtet
                                           ;  die LED 7
```

```
                bcf     INKRSTATUS,INKRLINKS  ;Anforderungsflag zuruecksetzen
                return


;*************************************************************************************************
;** INKRRECHTSDREHUNG                                                                           **
;** Diese Routine wird nach einer Drehung nach rechts des Inkrementalgebers aufgerufen          **
;**                                                                                             **
;** Aufgabe:                                                                                    **
;** Das am Port B gesetzte Bit (die am Port B leuchtende LED) um eine Position nach rechts      **
;** schieben. War vor dem Aufruf dieses Unterprogramms das Bit 7 (LED 7 leuchtete) gesetzt,     **
;** so ist nun nicht Bit 0 sondern das Carry-Flag gesetzt (Ueberlauf). -> durch ein weiteres    **
;** nach rechts schieben wird die LED 0 bzw. das Bit 0 von Port B gesetzt.                      **
;** Zum Schluss das Anforderungsflag loeschen                                                   **
;*************************************************************************************************
INKRRECHTSDREHUNG
                bcf     STAT,C                ;Carry-Flag loeschen
                rrf     PORTB,f               ;das gesetzte Bit (die leuchtende LED) um eine
                                              ;  Position nach rechts schieben

                btfsc   STAT,C                ;Bei einem Ueberlauf (Carry-Flag gesetzt) nach
                rrf     PORTB,f               ;  einmal nach rechts schieben. Nun leuchtet
                                              ;  die LED 0

                bcf     INKRSTATUS,INKRRECHTS ;Anforderungsflag zuruecksetzen
                return


;******************************** Hauptprogramm ***************************************************

;*************************************************************************************************
;** Aufgaben des Hauptprogramms:                                                                **
;**     + Aufrufen von INIT (Initialisierungen                                                  **
;**     + Interrupt freigeben                                                                   **
;**     + In einer Endlosschleife staendig die Flags INKRLINKS und INKRRECHTS im Register       **
;**       abfragen:                                                                             **
;**        + Wenn das Flag INKRLINKS im Register INKRSTATUS gesetzt ist, so wurde der Dreh-     **
;**          impulsgeber (Inkrementalgeber) nach links gedreht -> Unterprogramm                 **
;**          INKRLINKSDREHUNG aufrufen. (Hier erfolgen die Aktivitaeten, welche bei einer       **
;**          Drehung des Drehimpulsgebers nach links gewuenscht sind. Zum Beispiel vermindern   **
;**          eines Zaehlwerts um 1)                                                             **
;**        + Wenn das Flag INKRRECHTS im Register INKRSTATUS gesetzt ist, so wurde der Dreh-    **
;**          impulsgeber (Inkrementalgeber) nach rechts gedreht -> Unterprogramm                **
;**          INKRRECHTSDREHUNG aufrufen. (Hier erfolgen die Aktivitaeten, welche bei einer      **
;**          Drehung des Drehimpulsgebers nach links gewuenscht sind. Zum Beispiel erhoehen     **
;**          eines Zaehlwerts um 1)                                                             **
;*************************************************************************************************
Beginn          call    INIT                  ;Initialisierungen
                movlw   b'10100000'           ;Timer0 freigeben durch
                movwf   INTCON                ;Setzen von GIE und T0IE im INTCON Register

HPSCHLEIFE      btfsc   INKRSTATUS,INKRLINKS  ;Ist das Flag INKRLINKS im Register INKRSTATUS
                call    INKRLINKSDREHUNG      ;  gesetzt das Unterprogramm INKRLINKSDREHUNG
                                              ;  aufrufen

                btfsc   INKRSTATUS,INKRRECHTS ;Ist das Flag INKRRECHTS im Register INKRSTATUS
                call    INKRRECHTSDREHUNG     ;  gesetzt das Unterprogramm INKRRECHTSDREHUNG
                                              ;  aufrufen
                goto    HPSCHLEIFE

                end
```

4.3. Anmerkungen zur Software

Die Software besteht im Wesentlichen aus einem kurzen Hauptprogramm, einer Interrupt-Service-Routine (kurz ISR) und den Unterprogrammen INIT, INKRROUTINE, INKRRECHTSDREHUNG und INKRLINKSDREHUNG.

Die ISR wird alle 4ms aufgerufen und besitzt nur die Aufgabe das Unterprogramm INKRROUTINE zyklisch (eben alle 4 ms) aufzurufen. Dieses Unterprogramm wurde bereits im Abschnitt 3.4 (Unterprogramm „INKRROUTINE“) ausführlich beschrieben.

Bei eigenen Anwendungen muss in diesem Unterprogramm **nur** der verwendete Port angepasst werden.
Das Hauptprogramm besteht nach der Initialisierung (Unterprogramm INIT) und der Interruptfreigabe nur mehr aus einer Endlosschleife. Diese Endlosschleife prüft also ununterbrochen die beiden Flags INKRLINKS und INKRRECHTS. Wird vom Unterprogramm INKRROUTINE eines dieser beiden Flags gesetzt – am Drehimpulsgeber wurde also gedreht – so wird das entsprechende Unterprogramm (INKRRECHTSDREHUNG bzw. INKRLINKSDREHUNG) aufgerufen. In diesen beiden Unterprogrammen erfolgen nun die Anweisungen, die bei einer Drehung (in diese Richtung) erfolgen sollen. Diese beiden Unterprogramme sind daher projektspezifisch. Also in jeder Anwendung unterschiedlich. Hier soll zur Demonstration nur das gesetzte Bit am Port B entweder nach links oder nach rechts geschoben werden.

Das Unterprogramm INIT dient zur Initialisierung des Controllers. Hier werden u.a. die Ports konfiguriert (Port dient als Eingang oder als Ausgang), der oder die Timer eingestellt usw. Dieses Unterprogramm ist vom Controllertyp abhängig und je nach Anwendung mehr oder weniger umfangreich. Hier muss zusätzlich der Startzustand des Drehimpulsgebers am Port A eingelesen werden. Siehe auch Abschnitt 3.3 (Initialisierung)