Praktikumsprotokoll Mikrorechentechnik II – „Echtzeitsteuerung eines Wechselstromstellers mit einem Mikrocontroller“

Gruppe 50:
Marcel Junige, Fabian Kurz
Martin Laabs, Lars Lindenmüller

18. April 2005

Inhaltsverzeichnis

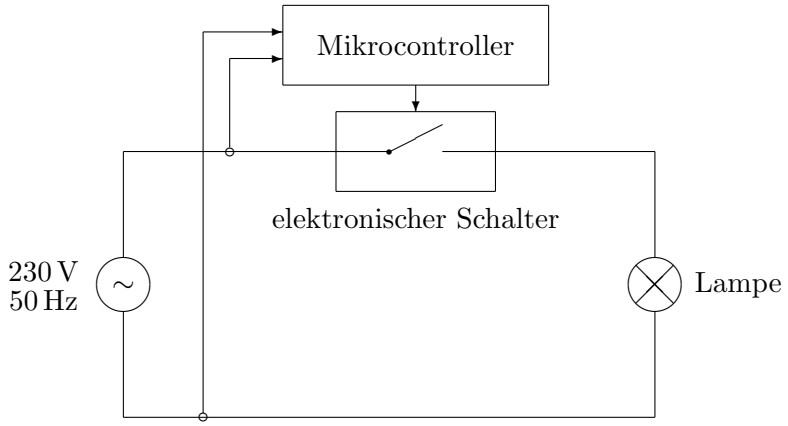

Abbildung 1: Schematische Darstellung des Wechselstromstellers

1 Aufgabenstellung

Aufgabe des Praktikums war die Realisierung eines Wechselstromstellers mit Hilfe eines Mikrocontrollers. Dieser arbeitet nach dem Prinzip des Phasenanschnitts: Der Wechselspannung wird erst nach Erreichen eines einstellbaren Steuerwinkels α durch Zünden eines Triacs eingeschaltet. Es ergibt sich ein von α abhängiger Effektivwert der Spannung, der der Beziehung

$$U_{\text{eff}} = U \cdot \sqrt{\frac{1}{\pi} \int_{\vartheta=\alpha}^{\pi} \left(\sqrt{2}\sin\vartheta\right)^2 \, d\vartheta} = U \cdot \sqrt{\frac{1}{\pi}\left(\pi - \alpha - \frac{1}{2}\sin(2\alpha)\right)} \tag{1}$$

folgt. Sobald der notwendige Haltestrom des Triacs unterschritten wird sperrt dieser wieder. Bei rein ohmscher Last erfolgt dies unmittelbar vor dem Spannungsnulldurchgang, jedoch treten in der Praxis meist ohmsch-induktive Lasten (Lampen, Motoren, ...) auf, bei denen der Stromverlauf gegenüber dem Spannunsgverlauf zeitlich verschoben ist. Der minimale Steuerwinkel α wird durch die Überlappung des Stromes in die nächste Halbwelle begrenzt.

Bild 1 stellt die Versuchsanordnung schematisch dar. Am Mikrokontroller sind Strom- und Spannungsdurchgangssensoren sowie ein Ausgang zur Zündung des Triacs angeschlossen. Zusätzlich sind zwei Taster zur Einstellung des Phasenwinkels und zwei LEDs als Statusanzeige vorhanden. Im Versuch wurde als ohmsc-induktive Last eine Glühbirne benutzt.

2 Zeitgraph

Zur Verdeutlichung der Funktionsweise wurde der zeitliche Verlauf aller relevanten Signale des Wechselstromstellers in Abbildung 2 für einen Einstellwinkel α von 45 ° und eine ohmsch-induktive Last dargestellt.

Die vier Signale sind von oben nach unten: Ausgang des Strom-Nulldurchgangsdetektors ($U_{I=0}$), Ausgang des Spannungs-Nulldurchgangsdetektors ($U_{I=0}$), Zündsignal des Triacs (U_{Triac}), Ausgangsstrom (I_a) und Ausgangsspannung (U_a).

Bezüglich der Echtzeitanforderung handelt es sich um „weiche“ Echtzeit, da das Verpassen eines Zündimpulses oder Nulldurchgangs i.d.R. lediglich zu einem verminderten Nutzen, nicht jedoch zu einem Totalausfall oder einer Beschädigung des Systems führt; nach einer Halbwelle wird wieder synchronisiert.

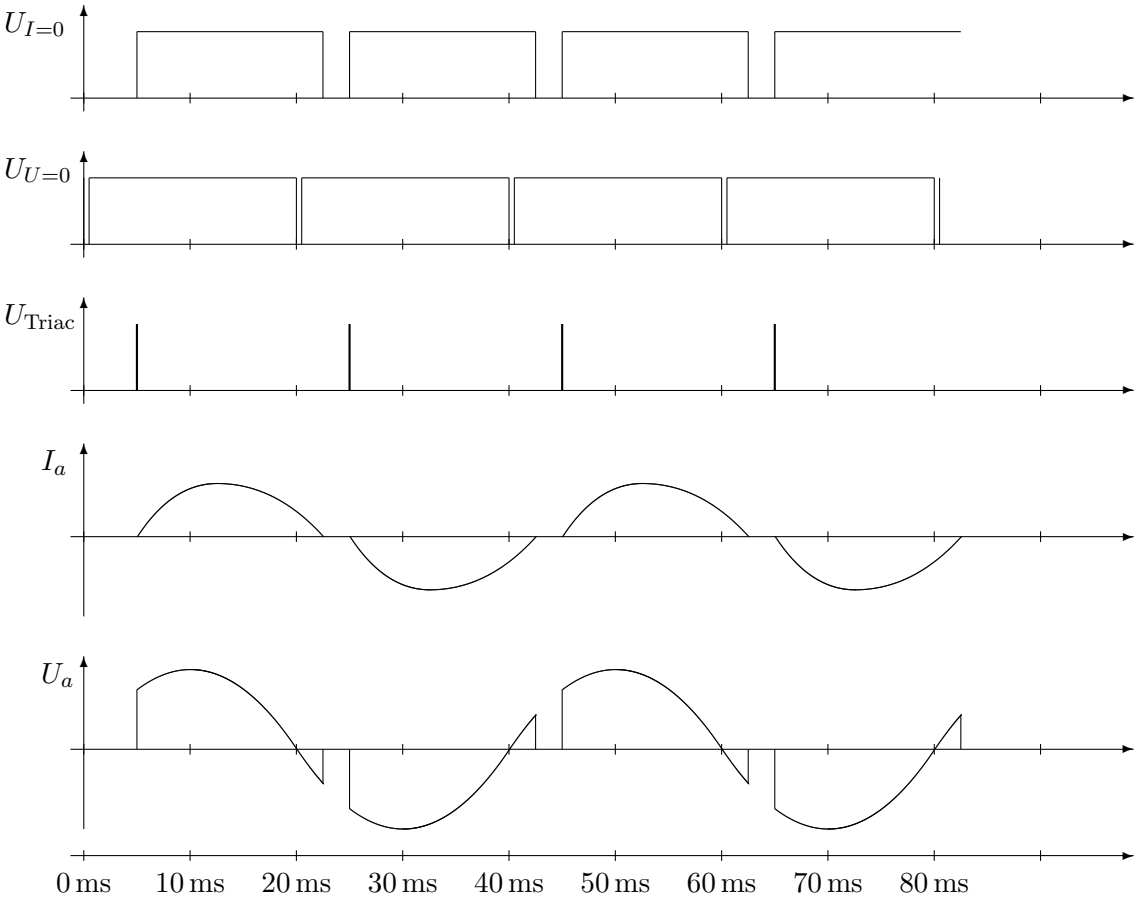

Abbildung 2: Zeitgraph der Signale am Wechselstromsteller

3 Programmablaufplan

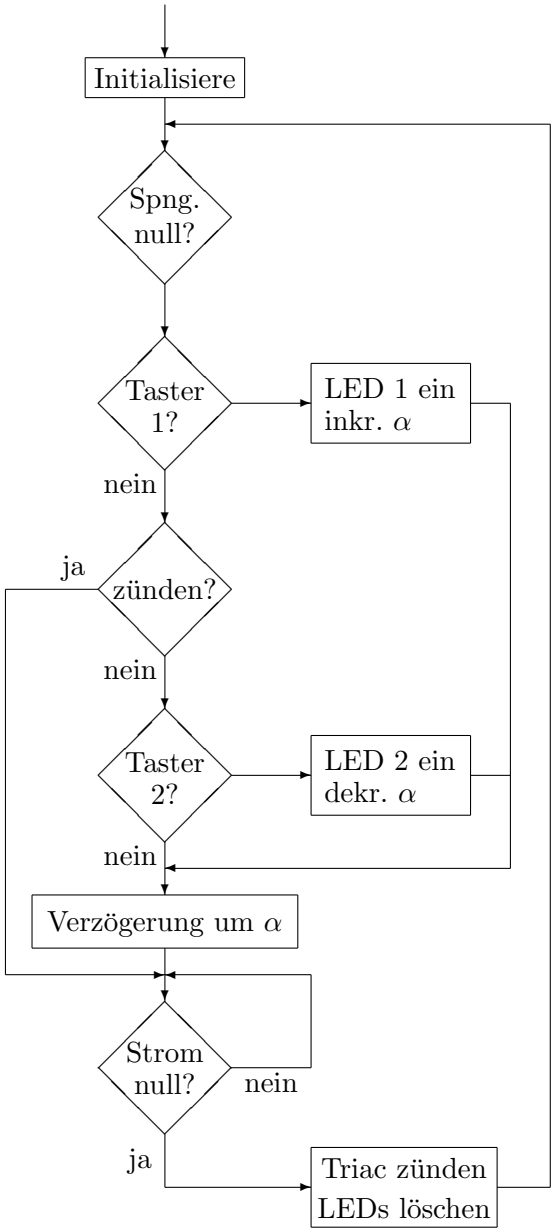

Abbildung 3: Flußdiagramm

Abbildung 3 beschreibt den Programmablauf. Am Anfang werden die Initialisierungen des Mikrocontrollers vorgenommen: Variablen für den Zündwinkel und die Verzögerungsschleife werden definiert und initialisiert, der Timer kalibriert, die Options-Register für die IO-Pins gesetzt (um zu definieren welche Pins Eingänge, welche Ausgänge sind) etc.

Danach beginnt die Hauptschleife. Es wird der nächste Spannungsnulldurchgang abgewartet. Nachdem dieser erfolgte, werden die Taster zur Veränderung des Zündwinkels α abgefragt und ggf. der Wert in einer temporären Variable inkrementiert oder dekrementiert und die passende LED eingeschaltet. Dabei wird überprüft und sichergestellt, daß der neue Winkel noch innerhalb des zulässigen Bereiches liegt, also kein Überlauf oder Unterlauf der Variable stattfindet.

Bevor der Triac gezündet werden darf, wird eine Verzögerungsschleife durchlaufen, deren Länge von α abhängt. Diese wird jedoch übersprungen, wenn nach der Abfrage des ersten Tasters (dunkler) bereits die Bedingung zum Zünden gegeben ist. Für den Fall, daß nach der Verzögerungsschleife immernoch Strom fließt wird noch auf den nächsten Stromnulldurchgang gewartet

bevor letztendlich gezündet wird. Nach erfolgter Zündung werden die LEDs ausgeschaltet und das Programm springt zurück zum Anfang. Das Programm wiederholt sich nun mit dem eventuell veränderten Winkel α.

4 Implementierung

Das Programm wurde in Assembler für den 16C505 Mikrocontroller geschrieben. Dieser hat einen sehr überschaubaren Befehlssatz, der jedoch für diese Anwendung völlig ausreichend ist.

Es folgt der kommentierte Quelltext

```
;*********************************************************************
; DATEINAME phasenanschnitt_1.asm
;*********************************************************************
; PROZESSOR PIC 16C505
; PINBELEGUNG
;   Vcc 1-*-14 Vss
;       2   13 Spannungsnulldurchgangserfassung
;       3   12 Stromnulldurchgangserfassung
;       4   11 LED1
;       5   10 LED2
;       6   9  Taster1
; Triac 7   8  Taster2
;*********************************************************************
; Praktikumsversuch     "Echtzeitsteuerung eines Wechselstromstellers
;                         mit einem PIC Mikrocontroller"
;
; Gruppe 50:            Marcel Junige, Fabian Kurz
;                       Martin Laabs, Lars Lindenmüller
;*********************************************************************
LIST P=16C505             ; Festlegung des verwendeten Prozessors
#include <P16c505.inc>    ; Verwendung von vorgegebenen Variablennamen
                          ; für den Mikrocontroller

;****** Variablen Definitionen ***************************************
wait    equ 08h           ; Hilfsregister für Unterprogramm "warten"
zuend   equ 09h           ; In diesem Register steht der Zündwinkel,
                          ; absolut, als vielfaches von 38cyc
zneu    equ 0Ah           ; Register für etwaigen neuen Zündwinkel

;****** BIT Definitionen *********************************************
#DEFINE _Unull PORTB,0      ; Spannungsnulldurchgang
#DEFINE _Inull PORTB,1      ; Stromnulldurchgang
#DEFINE _triac PORTC,3      ; Ansteuerung Triac
#DEFINE _Taster1 PORTC,1    ; dunkler
#DEFINE _Taster2 PORTC,2    ; heller
#DEFINE _LED1 PORTB,2       ; LED1
#DEFINE _LED2 PORTC,0       : LED2

;****** Start Vector *************************************************
    ORG 01FFh ; Reset Adresse
GOTO MAIN
```

```
    ORG 0000h ; Start Adresse
GOTO MAIN ; Sprung zum Hauptprogramm

;##### Hauptprogramm ################################################

;****** Initialisiere PIC ***********************************************
MAIN
    MOVWF OSCCAL        ; Timer kalibrieren
    CLRWDT              ; Clear Watchdog Timer
    MOVLW b'00000001'
    OPTION              ; setzt OPTION Register mit folgenden Werten
                        ; Bit7 wake-up on pin-change,
                        ; Bit6 weak pull-ups
                        ; Bit5 Timer0 on internal instruction clock,
                        ; Bit3 prescaler to Timer0
                        ; Bit0-2: Prescaler rate 1:4
    MOVLW b'11111011'
    TRIS PORTB          ; Setzt Pins des PortB(RB0..RB5) auf
    MOVLW b'11110110'   ; 1=Input(high-impedance), 0=Output
    TRIS PORTC          ; entsprechend PortC(RC0..RC5)
    CLRF PORTB
    CLRF PORTC

    ;zneu und zuend mit 200 initialisieren
    movlw d'200'
    movwf zneu
    movwf zuend

Nulldurchgang1    ; Zu Beginn auf den Spannungsnulldurchgang synchronisieren.
    btfsc _Unull
    goto Nulldurchgang1
Spannungssync1
    btfss _Unull
    goto Spannungssync1

;****** MAIN ************************************************************
START
    btfsc _Taster1    ; Testen ob Taster1 (dunkler) gedrückt wurde
    goto dunkler      ; Falls ja: "dunkler"-Routine starten

    movf zuend,0      ; zuend in akku kopieren (Z-Test) -> falls 0 wird
    btfsc STATUS,Z    ; Z-Flag gesetzt, dann SOFORT zünden
    goto Zuendtriac   ; dadurch minimale Verzögerung (maximale Helligkeit)

    btfss _Taster2    ; Testen ob Taster2 (heller) gedrückt wurde
    goto verzoeg      ; Falls nicht: Normale Verzögerungsroutine
    bsf _LED2         ; Falls doch: LED2 an
    decf zneu         ; und zneu um eins erniedrigen, normal weiter
    goto verzoeg      ; Anmerkung: Keine Routine nötig um zu erkennen, ob
                      ; zneu schon 0 ist, wird ja zu Beginn schon gemacht
```

```
dunkler
    bsf _LED1          ; LED1 leuchten lassen
    incf zneu          ; zneu incrementieren
    btfsc STATUS,Z     ; Falls Überlauf stattfand
    goto verzoeg       ; nicht weiter
    movlw h'FF'        ; sondern zneu
    movwf zneu         ; mit FFh beschreiben.

verzoeg
    movf zneu,0        ; neue Phase in Akkumulator schreiben
    movwf zuend        ; und uebernehmen
verzoeg2               ; innere Verzoegerungsschleife
    movf zuend,0       ; Testen, ob schon gezündet werden muss.
    btfsc STATUS,Z     ; Wenn zuend nicht Null ist naechsten Befehl skippen
    goto Zuendtriac
    movlw d'6'
    movwf wait
    call warten
    decf zuend         ; zuend decrementieren
    goto verzoeg2      ; und verzoeg2 erneut aufrufen.

Zuendtriac
    btfsc _Inull       ; Sicherheit: Falls noch Strom fließt (_Inull high)
    goto Zuendtriac    ; KEINE Zündung, sonst auf Stromabriss warten. 2cyc
                       ; Verlust werden in Kauf genommen.
    bsf _triac         ; Setze Zuendimpuls
    movlw d'255'
    movwf wait         ; warten
    call warten
    bcf _triac         ; Zuendimpuls zuende

Nulldurchgang          ; Am Ende auf den Spannungsnulldurchgang synchronisieren.
    btfsc _Unull
    goto Nulldurchgang
    ; Hier sollte genug Zeit sein eventuell gesetzte LEDs zu löschen (2cyc)
    bcf _LED1
    bcf _LED2
Spannungssync
    btfss _Unull
    goto Spannungssync
    goto START     ; ...und Abarbeitung von oben wiederholen.

; Unterprogramm "warten" realisiert eine einfache Warteschleife
; Der Wert, von dem heruntergezählt werden soll ist in "wait" zu schreiben.
warten
    decfsz wait,1
    goto warten
    retlw 0h
    END
;*****************************************************************
```

5 Zusatzaufgabe

Als Zusatzaufgabe war ein Programm zu realisieren, welches die angeschlossene Glühbirne im Sekundentakt blinken lässt. Dazu ließ sich ein Großteil des Quelltextes der eigentlichen Aufgabe wiederverwenden, mit dem Unterschied daß eine zusätzliche Variable `sec` eingeführt wurde, die von 100 bei jedem Nulldurchgang herunterzählt. Falls die Laufvariable bei Null angekommen ist, wird der Wert für α zwischen 0 und 255 hin- und hergeschaltet.

Im Quelltext muss also bei der Initialisierung noch eingefügt werden:

```
sec     equ 0Bh     ; Register für 1 sec Blinken

movlw d'100'        ; Halbwelle = 10ms, 100 * 10ms = 1s
movwf sec
```

Und der Anfang der Hauptschleife nach `START` wird alles bis zur Verzögerungsschleife durch folgenden Code ersetzt:

```
    movf sec,0
    btfsc STATUS,Z  ; Testen, ob sec schon Null ist
    goto toggle     ; wenn ja, toggeln
    decf sec
    goto verzoeg

toggle
    movf zneu,0
    xorlw 0FFh      ; zneu zwischen 0 und 255 toggeln
    movwf zneu
    movlw d'100'    ; und sec neu beschreiben
    movwf sec
```

Selbstverständlich wäre das Problem einfacher zu lösen, jedoch war das Programm zum Phasenanschnitt schon vorhanden und somit war die Modifikation der einfachste Weg.

6 Probleme

Das zu Hause vorbereitete Programm zeigte nicht auf Anhieb die gewünschte Funktion. Es stellte sich heraus, daß dies darauf zurückzuführen war, daß bei der Übernahme des neue Zündwinkels das Mnemonic `movwf` falsch geschrieben war (`movfw`). Dies wurde jedoch unglücklicherweise beim Assemblieren nicht bemängelt und blieb somit vorerst unbemerkt. Die folgende Fehlersuche gestaltete sich etwas langwierig und es wurden etliche Veränderungen am Programm gemacht, die sich letztendlich als unnötig herausstellten.

Nachdem dieser Fehler jedoch behoben war, funktionierte der Wechselstromsteller wie gefordert. Durch den während der Fehlersuche eingefügten sehr langen Zündimpuls konnten sogar entgegen den Erwartungen auch sehr kleine Zündwinkel eingestellt werden. Bei einem kurzen Zündimpuls ist dies problematisch, da der notwendige Haltestrom des Triacs eventuell nicht erreicht wird.