Ansteuerung einer (Full-Color)-RGB-Leuchtdiode

(mit PIC-Mikrocontroller)

Autor: Buchgeher Stefan

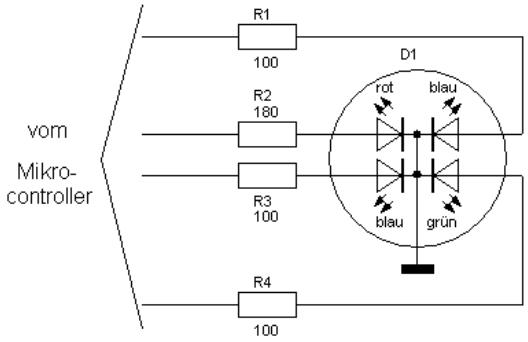
Letzte Bearbeitung: 29. März 2004

Inhaltsverzeichnis

1. Grundlegendes zur (Full-Color)-RGB-Leuchtdiode

noch nicht ausgeführt!

2. Hardware



Die Hardware zur Ansteuerung ist sehr einfach. Neben der Full-Color-RGB-Leuchtdiode (D1) sind nur mehr die 4 Strombegrenzungswiderstände R1 bis R4 notwendig.

Treiberstufen (z.B. mit einem Transistor) sind hier nicht notwendig, da ein PIC-Ausgang einen Strom von 25 mA liefern kann (bei High- und bei Low-Pegel). Jede der vier Teilleuchtdioden benötigt nur einen Strom von 20 mA.

3. Software

3.1. Benötigte Register und Portdefinitionen

Register:
Für die softwaremäßige Ansteuerung werden neben einigen PIC internen Register (SFR, **S**pezielle **F**unktions-**R**egister) noch folgende Register benötigt:

- RGB_ZAEHLER: Zählregister für die Puls-Weiten-Modulation
- RGB_WERT_ROT: Beinhaltet den Rot-Anteil
- RGB_WERT_GRUEN: Beinhaltet den Grün-Anteil
- RGB_WERT_BLAU: Beinhaltet den Blau-Anteil
- TEMP1: Hilfsregister

Portdefinition:
Im Allgemeinen werden bei jeder Anwendung die vier einzelnen Leuchtdioden der RGB-LED an einem beliebigen Port angeschlossen. Damit dies in der Software nur an einer Stelle berücksichtigt werden muss befindet sich in der Software eine Portdefinition. Diese besteht aus den folgenden Parametern:

- RGB_LED_ROT: Dieser Parameter gibt den Portpin für die rote Leuchtdiode an
- RGB_LED_GRÜN: Dieser Parameter gibt den Portpin für die grüne Leuchtdiode an
- RGB_LED_BLAU1: Dieser Parameter gibt den Portpin für die erste blaue Leuchtdiode an
- RGB_LED_BLAU2: Dieser Parameter gibt den Portpin für die zweite blaue

Eine mögliche Portdefinition ist:

```
RGB_LED_GRUEN           equ   0
RGB_LED_BLAU1           equ   1
RGB_LED_ROT             equ   6
RGB_LED_BLAU2           equ   7
```

3.2. Initialisierung des Mikrocontroller (Unterprogramm „INIT“)

Dieses Unterprogramm dient zur Initialisierung des Mikrocontrollers. Die Portpins an denen die einzelnen Leuchtdioden angeschlossen sind müssen als Ausgang definiert werden, und anschließend gelöscht werden.

Weiter sollten das Zählregister RGB_ZAEHLER und die Register für den roten, grünen und blauen Anteil initialisiert (also gelöscht) werden.

Der folgende Programmausschnitt zeigt eine mögliche Initialisierungsroutine. Die schwarz hervorgehobenen Stellen sind für die Ansteuerung der RGB-Leuchtdiode notwendig.

```
INIT        clrf  TMR0              ;Timer0 auf 0 voreinstellen

            clrf  PORTA
            movlw 0x07              ;Alle Komparatoreingaenge
            movwf CMCON             ;  auf digital I/O umschalten

            bsf   STAT,RP0          ;Registerseite 1
            movlw b'00000000'       ;interner Takt, Vorteiler = 2 an TMR0
            movwf OPTREG

            movlw b'00111100'       ;Port A definieren: Bits 0,1,6,7 als
                                    ;  Eingang
            movwf PORTA             ;                   Bits 2,3,4,5 als
                                    ;  Ausgang

            bcf   STAT,RP0          ;Registerseite 0

            movf  PORTA,w           ;Port A (Zustand der Tasten) einlesen
            movwf TASTENALT         ;  und im Register TASTENALT sichern

            clrf  PORTA             ;Port A loeschen

            movlw KONSTISR4MS       ;Zaehlregister fuer 4ms-Zeitbasis mit der
            movwf ZAEHLERISR4MS     ;  dazugehoerigen Konstanten laden

            clrf  RGB_ZAEHLER       ;Zaehlregister initialisieren (mit 0)
            clrf  RGB_WERT_ROT      ;Rot-Anteil initialisieren (mit 0)
            clrf  RGB_WERT_GRUEN    ;Gruen-Anteil initialisieren (mit 0)
            clrf  RGB_WERT_BLAU     ;Blau-Anteil initialisieren (mit 0)

            return
```

Anmerkung:
Hier bei diesem Beispiel befinden sich die vier Leuchtdioden der RGB-LED am Port A an den Pins 0 (grüne Leuchtdiode), 6 (rote Leuchtdiode) und 1 bzw. 7 (die beiden blauen Leuchtdioden)

3.3. Unterprogramm für die Ansteuerung der RGB-Leuchtdiode (mittels PWM)

Das Unterprogramm PWM_RGB_LED ist für die Ansteuerung der RGB-Leuchtdiode die wichtigste Komponente.

Diese Routine muss zyklisch (z.B. alle 512 µs) aufgerufen werden.

Aufgabe:
PWM für die RGB-Leuchtdiode erzeugen

Vorgehensweise:
- Hilfsregister (TEMP1) löschen
- PWM für den roten Anteil erzeugen: Dazu das Register RGB_WERT_ROT mit dem Zählregister RGB_ZAEHLER vergleichen. Ist RGB_WERT_ROT größer als das Zählregister (RGB_ZAEHLER), so muss der Ausgang für den Rotanteil gesetzt werden. Im Hilfsregister (TEMP1) daher das Flag für den roten Anteil (Flag RGB_LED_ROT) setzen.
- PWM für den grünen und blauen Anteil erzeugen (Dieser Vorgang erfolgt analog wie der für den roten Anteil)
- Hilfsregister (TEMP1) am Port A ausgeben
- Zählregister (RGB_ZAEHLER) für den nächsten Aufruf um 8 erhöhen

Hier das Unterprogramm:
```
PWM_RGB_LED clrf  TEMP1             ;Hilfsregister loeschen

            movf  RGB_WERT_ROT,w    ;PWM fuer den Rotanteil
            subwf RGB_ZAEHLER,w
            btfss STAT,C
            bsf   TEMP1,RGB_LED_ROT

            movf  RGB_WERT_GRUEN,w  ;PWM fuer den Gruenanteil
            subwf RGB_ZAEHLER,w
            btfss STAT,C
            bsf   TEMP1,RGB_LED_GRUEN

            movf  RGB_WERT_BLAU,w   ;PWM fuer den Blauanteil
            subwf RGB_ZAEHLER,
            btfsc STAT,C
            goto  PWM_RGB_WEITER1
            bsf   TEMP1,RGB_LED_BLAU1
            bsf   TEMP1,RGB_LED_BLAU2

PWM_RGB_WEITER1
            movf  TEMP1,w           ;Hilfsregister TEMP1 am
            movwf PORTA             ;  Port A ausgeben

            movf  RGB_ZAEHLER,w     ;Zaehlregister fuer den naechsten
            addlw .8                ;  Aufruf um 8 erhoehen
            movwf RGB_ZAEHLER

            return
```

Anmerkungen:
- Das temporäre Register TEMP1 wird hier nur zum Zwischenspeichern benötigt. Es kann daher auch in anderen Unterprogrammen verwendet werden.

- Das Zählregister RGB_ZAEHLER wird bei jedem Aufruf um 8 erhöht. Dadurch wird die Auflösung der Puls-Weiten-Modulation von 8 auf 5 Bit verkleinert. Die PWM kann nur mehr von 0 bis 31 eingestellt werden. Daraus ergibt sich aber eine PWM-Frequenz von ca. 61 Hz. (Bei 256 Schritten würde diese Frequenz 7,6Hz (!) betragen.)

4. Demonstrationsbeispiel

Das folgende Beispiel dient nur zur Demonstration. Es zeigt eine mögliche Einbindung des oben beschriebenen Unterprogramms.

4.1. Hardware

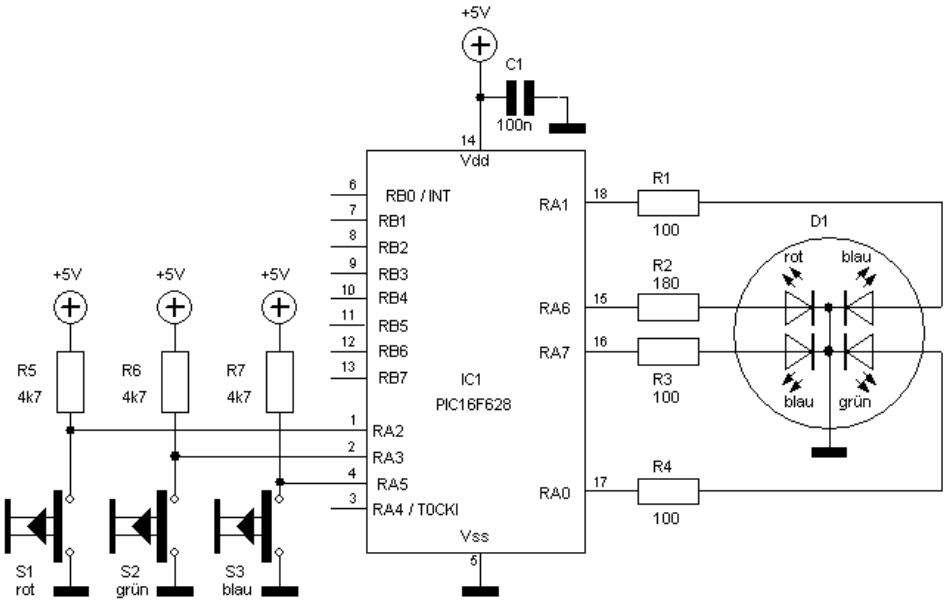


Bild 2: Schaltung zur Demonstration der Full-Color-RGB-Leuchtdiode

Bei diesem Demonstrationsbeispiel steuert ein Mikrocontroller vom Typ PIC16F628 über die Strombegrenzungswiderstäne R1 bis R4 die Full-Color-RGB-Leuchtdiode (D1) an.

Mit den Tastern S1 bis S3 kann die auszugebende Farbe bestimmt werden. Durch Drücken der Taste S1 wird der Rot-Anteil um den Wert 16 erhöht. Durch Drücken der Taste S2 der Grün-Anteil und der Blau-Anteil mit der Taste S3. Die Widerstände R4 bis R7 dienen hier als Pull-Up-Widerstände. Bei einem nicht gedrückten Taster (z.B. S1) liegt durch diesen Pull-Up-Widerstand (R5) am Portpin RA2 ein High-Pegel. Dieser wird durch drücken der Taste S1 auf low gezogen. Bei den Tastern S2 und S3 gilt natürlich dasselbe wie für S1.

Bei genauer Betrachtung der Hardware fällt auf, dass keine Oszillator und keine Reset-Schaltung vorhanden sind. Diese ist hier auch nicht notwendig, da diese der Mikrocontroller intern erzeugt. Der intern erzeugte Takt ist zwar nicht so frequenzstabil wie mit einem externen Quarz, reicht aber für diese Anwendung völlig aus.

Der Kondensator C1 dient zur Entkoppelung der Betriebsspannung für den Mikrocontroller (IC1). Für diesen Koppelkondensator sollte ein Keramiktyp verwendet werden. Dieser muss möglichst nahe beim Mikrocontroller angebracht werden.

Der Port B wird hier nicht verwendet und bleibt daher unbeschaltet.

4.2. Software

```
;*************************************************************************************************
;** Demonstrationsbeispiel zur Ansteuerung einer (Full-Color)-RGB-Leuchtdiode                   **
;**                                                                                             **
;** Entwickler: Buchgeher Stefan                                                                **
;** Entwicklungsbeginn der Software: 23. Maerz 2004                                             **
;** Funktionsfaehig seit: 23. Maerz 2004                                                        **
;** Letzte Bearbeitung: 29. Maerz 2004                                                          **
;*************************************************************************************************

               List p=PIC16F628


;******************************** Register (in Registerseite 0) ******************************
TMR0                    equ     1               ;Timer0-Register
STAT                    equ     3               ;Statusregister
PORTA                   equ     5               ;PortA-Register
INTCON                  equ     0B              ;Interrupt-Register
CMCON                   equ     1F              ;Komparator-Register


;******************************** Register (in Registerseite 1) ******************************
OPTREG                  equ     1               ;OPTION-Register
TRISA                   equ     5               ;Richtungsregister PortA


;******************************** Eigene Register (in Registerbank 0) ************************
ISR_STAT_TEMP           equ     20              ;Zwischenspeicher des Statusregister der ISR
ISR_w_TEMP              equ     21              ;Zwischenspeicher des Arbeitsregister der ISR
FLAGSISRHP              equ     22              ;beinhaltet Botschaftsflags ISR -> HP
ZAEHLERISR4MS           equ     23              ;Zaehlregister fuer die 4-ms-Zeitbasis
RGB_ZAEHLER             equ     24              ;Zaehlregister fuer die Puls-Weiten-Modulation
RGB_WERT_ROT            equ     25              ;Beinhaltet den Rot-Anteil
RGB_WERT_GRUEN          equ     26              ;Beinhaltet den Gruen-Anteil
RGB_WERT_BLAU           equ     27              ;Beinhaltet den Blau-Anteil
TASTENALT               equ     28              ;Register zur Ermittlung einer gedrueckten
                                                ;  Taste

TEMP1                   equ     5A              ;allgemeines Hilfsregister 1


;******************************** Bits in Registern der Registerbank 0 ***********************
;Register STAT
C                       equ     0               ;Carrybit im Statuswort-Register
RP0                     equ     5               ;Seitenauswahlbit im Statuswort-Register

;Register INTCON
T0IF                    equ     2               ;TMR0-Interruptflag im INTCON-Register


;******************************** Bits in den eigenen Registern der Registerbank 0 ************
;Register FLAGSISRHP
FLAG512USEK             equ     0               ;Botschaftsflag fuer 512us-Zeitbasis
FLAG4MSEK               equ     1               ;Botschaftsflag fuer 4ms-Zeitbasis


;******************************** Portbelegung ***********************************************
;Port A
RGB_LED_GRUEN           equ     0
RGB_LED_BLAU1           equ     1
RGB_LED_ROT             equ     6
RGB_LED_BLAU2           equ     7

TASTE_ROT               equ     2
TASTE_GRUEN             equ     3
TASTE_BLAU              equ     5
```

```
;******************************** Konstanten ************************************************
KONSTISR4MS           equ     .8              ;Konstante fuer 4ms-Zeitbasis


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


;******************************** Konfigurations-Bits ***************************************
_BODEN_ON             equ     h'3FFF'
_BODEN_OFF            equ     h'3FBF'

_CP_                  equ     h'03FF'
_CP_75                equ     h'17FF'
_CP_50                equ     h'2BFF'
_CP_OFF               equ     h'3FFF'

_DATA_CP_ON           equ     h'3EFF'
_DATA_CP_OFF          equ     h'3FFF'

_PWRTE_OFF            equ     h'3FFF'
_PWRTE_ON             equ     h'3FF7'

_WDT_ON               equ     h'3FFF'
_WDT_OFF              equ     h'3FFB'

_LVP_ON               equ     h'3FFF'
_LVP_OFF              equ     h'3F7F'

_MCLRE_ON             equ     h'3FFF'
_MCLRE_OFF            equ     h'3FDF'

_ER_OSC_CLKOUT        equ     h'3FFF'
_ER_OSC_NOCLKOUT      equ     h'3FFE'

_INTRC_OSC_CLKOUT     equ     h'3FFD'
_INTRC_OSC_NOCLKOUT   equ     h'3FFC'

_EXTCLK_OSC           equ     h'3FEF'
_LP_OSC               equ     h'3FEC'
_XT_OSC               equ     h'3FED'
_HS_OSC               equ     h'3FEE'

        __config      _MCLRE_OFF  &  _PWRTE_OFF  &  _WDT_OFF  &  _INTRC_OSC_NOCLKOUT  &  _BODEN_OFF  &
_LVP_OFF


                      ORG     0x000
                      goto    Beginn
                      ORG     0x004
                      goto    ISR


;******************************** Tabellen **************************************************
;keine Tabellen verwendet


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

;*****************************************************************************************
;** Interrupt Service Routine:                                                          **
;**                                                                                     **
;** Aufruf:                                                                             **
;**     alle 512 us                                                                     **
;**                                                                                     **
;** Aufgaben:                                                                           **
;**     + w-Register (=Arbeitsregister) und Status-Register zwischenspeichern (PUSH)    **
;**     + Zeitbasis fuer 512us und 4ms erzeugen                                         **
;**     + Das Timer-Interrupt-Flag T0IF wieder loeschen                                 **
;**     + w-Register (=Arbeitsregister) und Statusregister wiederherstellen (POP).      **
;*****************************************************************************************
ISR
PUSH            movwf   ISR_w_TEMP              ;w-Register retten
                swapf   STAT,w                  ;Statusregister
                movwf   ISR_STAT_TEMP           ;  retten
```

```
                ;Beginn der eigentlichen ISR-Routine
                bsf     FLAGSISRHP,FLAG512USEK;Botschaftsflag (512us-Zeitbasis) setzen

                decfsz  ZAEHLERISR4MS,f
                goto    ISRWEITER1

                bsf     FLAGSISRHP,FLAG4MSEK  ;Botschaftsflag (4ms-Zeitbasis) setzen
                movlw   KONSTISR4MS           ;Zaehlregister (ZAEHLERISR4MS) mit der
                movwf   ZAEHLERISR4MS         ;  Konstante (KONSTISR4MS) nachladen

ISRWEITER1
                ;Ende der eigentlichen ISR-Routine
ISRFERTIG       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:                                                    **
;**     + Komparatoreingaenge auf digitale I/O-Pins umschalten                        **
;**     + TMR0-ISR soll alle 512us aufgerufen werden, daher Vorteiler mit 2 laden (Bei einem  **
;**       internen 4-MHz-Takt)                                                        **
;**     + Ports: Port A: Bits 2,3,5 ..... Ausgaenge                                   **
;**                      Bits 0,1,6,7 ... Eingaenge                                   **
;**                      Bit 4 .......... wird hier nicht verwendet                   **
;**              Port B: wird hier nicht verwendet                                    **
;**     + Zustand der Tasten (am Port A) einlesen und im Register TASTENALT sichern   **
;**     + Port A loeschen                                                             **
;**     + Zaehlregister fuer die 4ms-Zeitbasis (ZAEHLERISR4MS) mit der Konstante KONSTISR4MS  **
;**       laden.                                                                      **
;**     + Zaehlregister fuer PWM (RGB-ZAEHLER) mit 0 initialisieren                   **
;**     + Rot, Gruen und Blauanteile mit 0 initialisieren                             **
;***************************************************************************************
INIT            clrf    TMR0                  ;Timer0 auf 0 voreinstellen

                clrf    PORTA
                movlw   0x07                  ;Alle Komparatoreingaenge
                movwf   CMCON                 ;  auf digital I/O umschalten

                bsf     STAT,RP0              ;Registerseite 1
                movlw   b'00000000'           ;interner Takt, Vorteiler = 2 an TMR0
                movwf   OPTREG

                movlw   b'00111100'           ;Port A definieren: Bits 0,1,6,7 als Eingang
                movwf   PORTA                 ;                   Bits 2,3,4,5 als Ausgang

                bcf     STAT,RP0              ;Registerseite 0

                movf    PORTA,w               ;Port A (Zustand der Tasten) einlesen
                movwf   TASTENALT             ;  und im Register TASTENALT sichern

                clrf    PORTA                 ;Port A loeschen

                movlw   KONSTISR4MS           ;Zaehlregister fuer 4ms-Zeitbasis mit der
                movwf   ZAEHLERISR4MS         ;  dazugehoerigen Konstanten laden

                clrf    RGB_ZAEHLER           ;Zaehlregister initialisieren (mit 0)
                clrf    RGB_WERT_ROT          ;Rot-Anteil initialisieren (mit 0)
                clrf    RGB_WERT_GRUEN        ;Gruen-Anteil initialisieren (mit 0)
                clrf    RGB_WERT_BLAU         ;Blau-Anteil initialisieren (mit 0)

                return


;***************************************************************************************
;** PWM_RGB_LED                                                                       **
;**                                                                                   **
;** Aufgabe:                                                                          **
;**     PWM fuer die RGB-Leuchtdiode erzeugen:                                        **
;**                                                                                   **
;** Vorgehensweise:                                                                   **
;**     + Hilfsregister (TEMP1) loeschen                                              **
```

```
;**     + PWM fuer den roten Anteil erzeugen: Dazu das Register RGB_WERT_ROT mit dem Zaehl-   **
;**       Register RGB_ZAEHLER vergleichen. Ist RGB_WERT_ROT groesser als das Zaehlregister   **
;**       (RGB_ZAEHLER) so muss der Ausgang fuer den Rotanteil gesetzt werden. Im Hilfs-      **
;**       register (TEMP1) daher das Flag fuer den roten Anteil (Flag RGB_LED_ROT) setzen.    **
;**     + PWM fuer den gruenen und blauen Anteil erzeugen (Dieser Vorgang erfolgt analog wie  **
;**       der fuer den roten Anteil)                                                          **
;**     + Hilfsregister (TEMP1) am Port A ausgeben                                          **
;**     + Zaehlregister (RGB_ZAEHLER) fuer den naechsten Aufruf um 8 erhoehen               **
;**                                                                                         **
;** Anmerkungen:                                                                            **
;**     + Das temporaere Register TEMP1 wird hier nur zum Zwischenspeichern benoetigt. Es   **
;**       kann daher auch in anderen Unterprogrammen verwendet werden.                      **
;**     + Das Zaehlregister RGB_ZAEHLER wird bei jedem Aufruf um 8 erhoeht. Dadurch wird die  **
;**       Aufloesung der Puls-Weiten-Modulation von 8 auf 5 Bit verkleinert. Die PWM kann    **
;**       nur mehr von 0 bis 31 eingestellt werden. Daraus ergibt sich aber eine PWM-Frequenz **
;**       von ca. 61 Hz. (Bei 256 Schritten wuerde diese Frequenz 7,6Hz (!) betragen.)      **
;*****************************************************************************************
PWM_RGB_LED     clrf    TEMP1                           ;Hilfsregister loeschen

                movf    RGB_WERT_ROT,w                  ;PWM fuer den Rotanteil
                subwf   RGB_ZAEHLER,w                   ;  (siehe obige Beschreibung)
                btfss   STAT,C
                bsf     TEMP1,RGB_LED_ROT

                movf    RGB_WERT_GRUEN,w                ;PWM fuer den Gruenanteil
                subwf   RGB_ZAEHLER,w                   ;  (siehe obige Beschreibung)
                btfss   STAT,C
                bsf     TEMP1,RGB_LED_GRUEN

                movf    RGB_WERT_BLAU,w                 ;PWM fuer den Blauanteil
                subwf   RGB_ZAEHLER,w                   ;  (siehe obige Beschreibung)
                btfsc   STAT,C
                goto    PWM_RGB_WEITER1
                bsf     TEMP1,RGB_LED_BLAU1
                bsf     TEMP1,RGB_LED_BLAU2

PWM_RGB_WEITER1
                movf    TEMP1,w                         ;Hilfsregister TEMP1 am
                movwf   PORTA                           ;  Port A ausgeben

                movf    RGB_ZAEHLER,w                   ;Zaehlregister fuer den naechsten
                addlw   .8                              ;  Aufruf um 8 erhoehen
                movwf   RGB_ZAEHLER

                return


;*****************************************************************************************
;** TASTENROUTINE                                                                         **
;** Diese Routine wird alle 4ms vom Hauptprogramm aufgerufen                              **
;**                                                                                       **
;** Aufgabe: Ermittelt welche Taste gedrueckt wurde und fuehrt die dazugehoerigen Anweisungen**
;**     aus.                                                                              **
;**                                                                                       **
;** Allgemeines: Die Tasten sind am Port A mit je einem Pull-Up-Widerstand angeschlossen.    **
;**     Im Ruhezustand (Taste nicht gedrueckt) liegt an dem Portpin ein High-Pegel. Durch  **
;**     Druecken der Taste wird dieser Portpin auf Masse gelegt (Low-Pegel). Dieses Verhalten **
;**     wird auch als Lowaktiv bezeichnet.                                                **
;**                                                                                       **
;** Vorgehensweise:                                                                       **
;**     + Port A einlesen und im Register TEMP1 sichern.                                  **
;**     + Das Register TASTENALT gibt beim Aufruf den Zustand der Tasten an, welcher beim  **
;**       vorhergehenden Aufruf dieses Unterprogramms am Port A herrschte. (Also den Zustand  **
;**       am Port A vor 4ms)                                                              **
;**     + Mit Hilfe der beiden Register TEMP1 und TASTENALT laesst sich somit ermitteln ob  **
;**       eine bestimmte Taste zwischen dem vorhergehenden Aufruf dieses Unterprogramms und  **
;**       den gerade bearbeitenden Aufruf gedrueckt wurde. Mit diesem Verfahren wird auch   **
;**       gleichzeitig das so genannte Tastenprellen ueberbrueckt, da dieses Tastenprellen  **
;**       in der Regel weit weniger als 4ms dauert.                                       **
;**       Die Befehlsfolge    comf    TEMP1,w                                             **
;**                           andwf   TASTENALT,f                                         **
;**       prueft also, ob eine Taste gedrueckt wurde. Dazu muessen die folgenden Bedingungen  **
;**       gelten:                                                                         **
;**                           TEMP1,x         0 (Low)                                     **
;**                           TASTENALT,x     1 (High)                                    **
;**       Das Ergebnis (welche Taste gedrueckt wurde) wird im Register TASTENALT gesichert.  **
;**       Das Register TEMP1 bleibt dabei unveraendert.                                   **
;**     + je nachdem welche Bits (Flags) nun im Register TASTENALT gesetzt sind, die dazuge-  **
;**       hoerigen Anweisungen ausfuehren (Hier: Je nach gedrueckter Taste die Anteile fuer  **
```

```
;**       Rot, Gruen oder Blau um 16d erhoehen)                                          **
;**     + Registerinhalt von TEMP1 in das Register TASTENALT kopieren                    **
;**     + Unterprogramm verlassen                                                        **
;**                                                                                      **
;** Achtung: Tasten sind LOWAKTIV!                                                       **
;**                                                                                      **
;** Anmerkung: Das temporaere Register TEMP1 wird hier nur zum Zwischenspeichern benoetigt.  **
;**       Es kann daher auch in anderen Unterprogrammen verwendet werden.                **
;******************************************************************************************
TASTENROUTINE  movf   PORTA,w              ;Port A einlesen,
               movwf  TEMP1                ;  und in Register TEMP1 kopieren
               comf   TEMP1,w
               andwf  TASTENALT,f

               btfsc  TASTENALT,TASTE_ROT
               goto   TASTEROT             ;Taste TASTE_ROT wurde gedrueckt

               btfsc  TASTENALT,TASTE_GRUEN
               goto   TASTEGRUEN           ;Taste TASTE_GRUEN wurde gedrueckt

               btfsc  TASTENALT,TASTE_BLAU
               goto   TASTEBLAU            ;Taste TASTE_BLAU wurde gedrueckt
               goto   TASTENWEITER1

TASTEROT       movf   RGB_WERT_ROT,w        ;Rot-Anteil um 16d erhoehen
               addlw  .16
               movwf  RGB_WERT_ROT
               goto   TASTENWEITER1

TASTEGRUEN     movf   RGB_WERT_GRUEN,w      ;Gruen-Anteil um 16d erhoehen
               addlw  .16
               movwf  RGB_WERT_GRUEN
               goto   TASTENWEITER1

TASTEBLAU      movf   RGB_WERT_BLAU,w       ;Blau-Anteil um 16d erhoehen
               addlw  .16
               movwf  RGB_WERT_BLAU
               goto   TASTENWEITER1

TASTENWEITER1  movf   TEMP1,w               ;TEMP1 in Register TASTENALT kopieren
               movwf  TASTENALT
               return


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

;******************************************************************************************
;** Aufgaben des Hauptprogramms:                                                         **
;**     + Controller initialisieren (Unterprogramm INIT)                                 **
;**     + Interrupt freigeben                                                            **
;**     + Taetigkeiten/Unterprogramme, die alle 4ms durchgefuehrt werden muessen:        **
;**           + Unterprogramm PWM_RGB_LED aufrufen                                       **
;**           + Unterprogramm TASTENROTINE aufrufen                                      **
;******************************************************************************************
Beginn         call   INIT                 ;Controller initialisieren

               movlw  b'10100000'          ;Timer0 freigeben durch Setzen von
               movwf  INTCON               ;  GIE und T0IE im Register INTCON

HPSCHLEIFE     btfsc  FLAGSISRHP,FLAG512USEK
               goto   HPWEITER1
               btfsc  FLAGSISRHP,FLAG4MSEK
               goto   HPWEITER2
               goto   HPSCHLEIFE

HPWEITER1      ;Taetigkeiten, die alle 512 us durchgefuehrt werden muessen
               call   PWM_RGB_LED
               bcf    FLAGSISRHP,FLAG512USEK;Anforderungsflag fuer die 512us-Aktivitaeten
                                           ;  zuruecksetzen
               goto   HPSCHLEIFE

HPWEITER2      ;Taetigkeiten, die alle 4 ms durchgefuehrt werden muessen
               call   TASTENROUTINE
               bcf    FLAGSISRHP,FLAG4MSEK  ;Anforderungsflag fuer die 4ms-Aktivitaeten
                                           ;  zuruecksetzen
               goto   HPSCHLEIFE

               end
```

4.3. Anmerkungen zur Software

Die Software besteht im Wesentlichen aus einem kurzen Hauptprogramm, einer kurzen Interrupt-Service-Routine (kurz ISR), das im Abschnitt 3.3. besprochene Unterprogramm PWM_RGB_LED und einem Unterprogramm welches zyklisch die Tasten einliest.

Die ISR wird alle 512µs aufgerufen und besitzt „nur“ die Aufgabe zwei Zeitbasen zu erzeugen, indem sie zwei Botschaftsflags für das Hauptprogramm erzeugt. Eine Zeitbasis alle 512µs und die zweite Zeitbasis ca. alle 4ms. Da die ISR alle 512µs aufgerufen wird ist die 512-µs-Zeitbasis besonders einfach. Es muss nur bei jedem Aufruf ein Flag gesetzt werden, und zwar das Flag FLAG512USEK im Register FLAGSISRHP. Die Generierung für die 4-ms-Zeitbasis ist dagegen schon etwas „umfangreicher“. Damit eine Zeit von ca. 4 ms entsteht, muss die ISR 8-mal aufgerufen werden (8 x 512µs = 4096µs = 4ms). Bei jedem ISR-Aufruf muss also ein Zählregister um 1 vermindert werden. Besitzt es danach den Wert 0, so sind 4ms vergangen. Nun wird das Botschaftsflag FLAG4MSEK im Register FLAGSISRHP gesetzt, und das Zählregister muss mit dem Wert 8 neu geladen werden. Der Wert 8 wird hier durch die Konstante KONSTISR4MS ersetzt.

Die ISR wird, wie schon mehrmals erwähnt, alle 512µs aufgerufen Diese 512µs ergeben sich folgendermaßen: TMR0 wird mit dem Wert 0 geladen – es dauert also 256 Taktzyklen bis das Register wieder den Wert 0 besitzt, der Vorteiler besitzt den Wert 2 Der Taktzyklus ergibt sich aus dem internen Takt von 4MHz. Dieser ist bei der PIC-Familie wie folgt definiert:

$$\frac{4}{f_{Quarz}}$$

Daraus ergibt sich folgender Zusammenhang:

$$ISRAUFRUF \quad [\mu s] = \frac{4 \bullet 256 \bullet 2}{f_{Quarz}\ [MHz]} = \frac{4 \bullet 256 \bullet 2}{4} = 512$$

Also ein ISR-Aufruf alle 512µs.

Das Hauptprogramm befindet sich nach der Initialisierung (Unterprogramm INIT) und der Interruptfreigabe in einer Endlosschleife. Diese Schleife besitzt die Aufgabe ständig die so genannten Botschaftsflags abzufragen. Ist eines dieser Botschaftsflags gesetzt, so muss vom Hauptprogramm eine bestimmte Aufgabe ausgeführt werden. Diese Aufgaben sind in Form von Unterprogrammen vorhanden. Hier bei der Ansteuerung einer RGB-Leuchtdiode werden diese beiden Botschaftsflags in der Timer-ISR gesetzt. Und zwar wird alle 512µs ein Flag gesetzt. Das zweite Flag von der ISR wird alle 4ms gesetzt. (Botschaftsflags können aber auch in anderen Unterprogrammen gesetzt werden, was aber hier nicht der Fall ist.)

Hier Zusammenfassend die Tätigkeiten in der Endlosschleife, welche durch die Botschaftsflags ausgelöst werden:

- Das eigentliche Unterprogramm zur Ansteuerung der RGB-Leuchtioden wird alle 512µs aufgerufen (Unterprogramm PWM_RGB_LED)
- Alle 4ms die Tasten S1 bis S3 abfragen. Dazu dient das Unterprogramm TASTENROUTINE

Das Unterprogramm INIT dient zur Initialisierung des Controllers. Hier werden unter anderem 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. Siehe auch Abschnitt 3.2 (Initialisierung)

Das Unterprogramme PWM_RGB_LED ist für die eigentliche Ansteuerung der RGB-Leuchtdiode zuständig. Siehe Abschnitt 3.3

Das Unterprogramm TASTENROUTINE wird zyklisch alle 4ms aufgerufen und ermittelt welche Taste gedrückt wird. Erkennt dieses Unterprogramm, dass eine Taste gedrückt wurde, so erhöht es den entsprechenden Farbanteil. Wie dieses Unterprogramm einen Tastendruck erkennt ist im Listing ausreichend beschrieben (siehe Listing, Abschnitt 4.2)

5. Quellen