Mikrocontroller-Grundlagen Teil 16

Im sechzehnten Teil dieser Artikelserie zeigen wir anhand ausführlicher Beispiele die praktische Programmierung und Funktion der Interrupts der MCS51-Familie.

6.4 Praktische Interrupt-Programmierung

In den folgenden Kapiteln wollen wir anhand praktischer Beispiele die Interrupt-Möglichkeiten der MCS51-Familie näher beschreiben, um eine Grundlage für weitere Entwicklungsschritte zu schaffen.

Für die korrekte Funktion der folgenden Beispielprogramme ist es erforderlich, daß an dem Port P1 der Mikrocontroller-Grundlagenplatine die 8-Bit-LED-Ausgabeeinheit angeschlossen ist, während an dem Port P3 die 8-Bit-Schaltereinheit adaptiert wird.

Abbildung 129 zeigt den allgemeinen Aufbau einer Interrupt-Service-Routine, deren erster Sprungbefehl, wie in Kapitel 6.3 beschrieben, an der zu der Interrupt-Quelle gehörenden Einsprungadresse beginnen muß. Die eigentliche Interrupt-Verarbeitungs-Routine, die sich in einem beliebigen Bereich des Code-Speichers des Mikrocontrollers befinden kann, sichert zunächst den Inhalt des Akkumulators und des Programmstatuswortes auf dem Stack.

Üblicherweise beginnt nach der Anwahl des benötigten Registersatzes die eigentliche Interrupt-Bearbeitung. In den zu diesem Grundlagenartikel gehörenden Beispielprogrammen werden die Programmunterbrechungsroutinen von unterschiedlichen Programmteilen, die über die beiden 10fach-Drehschalter einzustellen sind, benutzt.

In der betreffenden Interrupt-Service-Routine ist nun der richtige Programmteil auszuführen, wozu zunächst der Inhalt der Variablen „SCH_NR", die beim Start des Mikrocontrollers mit der Programmnummer geladen wird, in den Akkumulator zu kopieren ist. Der darauffolgende CJNE-Befehl vergleicht nun den Inhalt des Akkumulators mit der Nummer der Interrupt-Service-Routine und springt zur nächsten, wenn diese nicht übereinstimmt.

Ist nun der benötigte Programmteil gefunden, arbeitet der Mikrocontroller den sich dort befindenden Programmcode ab, an dessen Abschluß ein Sprung zum Ende der Interrupt-Routine steht. Dort werden der Inhalt des Programmstatuswortes und des Akkumulators wiederhergestellt und anschließend die Programmunterbrechung durch Ausführung des RETI-Befehls beendet.

6.4.1 Externer Interrupt

Abbildung 130 zeigt das Testprogramm 43 mit der dazugehörenden Interrupt-Service-Routine, die mit jeder fallenden Flanke am INT0-Anschlußpin des Mikrocontrollers eine Programmunterbrechung veranlaßt, in welcher der Inhalt des Ports P1 jeweils um 1 inkrementiert wird.

Nachdem das Programm den Inhalt des Ports P1 zunächst auf 0 initialisiert, erfolgt die Vorbereitung der Interrupt-Freigabe. Durch Setzen des direkt adressierbaren Bits IT0 erfolgt eine Interrupt-Auslösung nur nach einer fallenden Flanke an dem INT0-Anschlußpin des Mikroprozessors, während die Zulassung des externen Interrupts 0 durch das Setzen des EX0-Flags erfolgt.

Die Priorität des Interrupts wird in diesem Beispiel auf niedriger Ebene festgelegt, wobei bei der Benutzung nur eines Interrupts in einem Anwenderprogramm die Prioritätsvergabe keine Rolle spielt, da die Ausführung der Programmunterbrechung jederzeit möglich ist.

Den Abschluß der Initialisierung bildet das Setzen des allgemeinen Interrupt-Freigabebits IEA, wonach die Auslösung der Programmunterbrechung freigegeben ist. Das Hauptprogramm könnte jetzt unabhängig von einer Interrupt-Anforderung beliebige andere Aufgaben wahrnehmen. In dem vorliegenden Beispiel sind keine weiteren Aktivitäten des Hauptprogrammes erforderlich, weshalb sich am Ende des Programmes eine Endlosschleife befindet.

Mit Auslösung eines Interrupts sichert der Mikrocontroller den aktuellen Programmzählerinhalt auf dem Stack und setzt die Programmabarbeitung ab der Adresse 0003H (externer Interrupt 0-Vektor) fort,

```
            ; Adresse des Interruptvektors
            LJMP INTROUT    ;

INTROUT:    ; Beginn der Interrupt-Verarbeitungsroutine
            PUSH ACC        ; Inhalt des Akkus retten
            PUSH PSW        ; Programm-Status-Wort sichern
            MOV A, SCH_NR   ; Dreh-Schalter-Nummer laden
            CJNE A,#43, INT_43; Springe, wenn nicht aktuelles Programm
            ; Hier beginnt die Verarbeitung der Programm-Nr.43
            • • •
            SJMP INT_END ; Ende der Routine
INT_43:     CJNE A,#44, INT_44; Springe, wenn nicht aktuelles Programm
            ; Hier beginnt die Verarbeitung der Programm-Nr.44
            • • •
            SJMP INT_END    ; Ende der Routine
INT_44:     • • •

INT_END:    POP PSW         ; Programm-Status-Wort wieder herstellen
            POP ACC         ; Original-Inhalt des Akkus wieder herstellen
            RETI            ; zurück zum Hauptprogramm
```

Bild 129: Allgemeiner Aufbau einer Interrupt-Service-Routine

```
05B1            PRGM43:                     ; Ext INT 0 Interrupt
05B1  759000            MOV   P1, #000H     ; Alle LEDs aus
05B4  D288              SETB  IT0           ; Interrupt-Auslösung mit der fallenden Flanke
                                            ; an INT0
05B6  D2A8              SETB  EX0           ; Externen Interrupt 0 zulassen
05B8  C2B8              CLR   PX0           ; Externen Interrupt 0 auf niedriger Prioritäts-
                                            ; Ebene
05BA  D2AF              SETB  EA            ; Allgemeine Interruptfreigabe
05BC  A1BC              AJMP  $             ; Endlosschleife
                              Externer Interrupt 0
002E            EXTINT0:                    ; Interruptroutine für den externen Interrupt 0
002E  C0E0              PUSH  ACC           ; Inhalt des Akkus retten
0030  C0D0              PUSH  PSW           ; Programm-Status-Wort sichern
0032  E520              MOV   A, SCH_NR     ; Dreh-Schalter Nummer laden

0034  B42B04            CJNE  A,#43,
                        EX0_43              ; Springe, wenn nicht Programm 43
                                            ; Hier beginnt die eigentliche Interrupt-
                                              Verarbeitung
0037  0590              INC   P1            ; Inhalt des Ports P1 um 1 erhöhen
0039  8000              SJMP  EXT1END       ; Ende der Routine

003B            EX0_43:                     ; Nächste Routine
003B  D0D0      EXT1END: POP  PSW           ; Programm-Status-Wort wieder herstellen
003D  D0E0              POP   ACC           ; Original-Inhalt des Akkus wieder herstellen
003F  32                RETI                ; zurück zum Hauptprogramm
```

Bild 130: Testprogramm 43 mit der dazugehörenden Interrupt-Service-Routine

wo ein Sprung zur eigentlichen Interrupt-Service-Routine EXTINT0 steht.

Nach dem Sichern des Akku- und Programmstatuswort-Inhaltes und der Programmnummernabfrage beginnt die eigentliche Interrupt-Verarbeitung.

In dem vorliegenden Beispiel wird lediglich der Inhalt des Ports P1 um 1 erhöht. Am Ende der Interrupt-Verarbeitung werden die Inhalte des Programmstatuswortes und des Akkumulators in umgekehrter Reihenfolge vom Stack zurückgelesen.

Den Abschluß der Interrupt-Service-Routine bildet die Ausführung des RETI-Befehls, der zum einen dafür sorgt, daß das Hauptprogramm an die zuvor verlassene Stelle zurückkehrt und zum anderen das IPP-Flip-Flop zurücksetzt.

Liegt nach dem Einschalten und Start des Prozessors der Pegel des Schalters S 2 der Schalterplatine auf „low", so wertet dies der Prozessor bereits als eine fallende Flanke, woraufhin bereits der erste Interrupt ausgelöst wird, welcher den Port-Inhalt von P3 auf 1 setzt.

Mit jeder über den Schalter S2 verursachten High-Low-Flanke erhöht die aufgerufene Interrupt-Service-Routine den Inhalt des P1-Zählers um 1.

6.4.2 Timer-Interrupt, 16-Bit-Zähler

Abbildung 131 zeigt das Testprogramm 44, das die Timer-Interrupt-Programmierung näher beleuchtet. Im Hauptprogramm wird zunächst die Timer-Initialisierung und anschließend die Interrupt-Freigabe vorgenommen.

Der Mode 1 des Timers 0 steuert die zu einem 16-Bit-Zähler zusammengeschalteten 8-Bit-Zähler TL0 und TH0 mit einem durch 12 geteilten Oszillatortakt an. Bei unbeeinflußtem Zähler würde somit nach jedem 786432. (12 • 256 • 256) Oszillatortakt das Timer-Flag TF0 gesetzt werden.

Nach der Freigabe der Timer-Ansteuerung durch das Steuerbit TR0 erfolgt die Timer0-Interrupt-Freigabe, das Festlegen der niedrigen Prioritätsebene für den Timer-Interrupt und die allgemeine Interrupt-Freigabe. Auch in diesem Hauptprogramm sind nach der Initialisierung keine weiteren Aktionen erforderlich, was zu einer Endlosschleife führt.

In der Interrupt-Service-Routine TI0_INT erfolgt nach dem Sichern der Akku- und PSW-Inhalte das Neuladen der beiden 8-Bit-Zähler TL0 und TH0. Bei freilaufendem Zählerinhalt würde der Timer0-Interrupt ca. 14mal pro Sekunde (11,0592 MHz geteilt durch 12 • 256 • 256) ausgelöst werden, da der 16-Bit-Zähler beim Überlauf automatisch wieder mit dem Zählerstand „0" beginnt.

In dem vorliegenden Beispiel wird der Zählerinhalt mit einem neuen Zahlenwert geladen. Dieser setzt den Zählerstand auf 7936 (1F00H) vor. Damit sind bis zur nächsten Interrupt-Auslösung nicht mehr 65565, sondern nur noch 57600 (65536 - 7936) Maschinentakte erforderlich, wodurch der Zähler im Idealfall 16 Interrupts pro Sekunde generiert.

Durch die Interrupt-Reaktionszeit von 1 bis 4 Maschinenzyklen und die Abarbeitung der ersten 5 Interrupt-Befehle (ca. 13 Maschinenzyklen) muß zu dem Interrupt-Zeitabstand von 62,5 ms (1 geteilt durch 16 INT/s) noch die Reaktionszeit hinzugezählt werden.

Da in diesem Beispiel die unteren 8 Bit des Zählers TL0 mit dem Zahlenwert 000H geladen werden, könnte in diesem speziellen Beispiel das Neuladen des niederwertigen 8-Bit-Zählers ersatzlos entfallen, was dann die Wiederholgenauigkeit wesentlich erhöht.

Bild 131: DasTestprogramm 44 generiert über einen 16-Bit-Timer ca. 16 Interrupts pro Sekunde

```
05BE            PRGM44:                     ; Timer 0 Interrupt 16-Bit-Zähler
                                            ; Dieses Unterprogramm erzeugt ca. 16
                                            ; Interrupts pro Sekunde
05BE  758901            MOV   TMOD, #00000001B; Timer0: Gate aus,
                                            ; Timer Mode 1 (16 Bit Zähler)  Timer 1
                                            ; keine Funktion
05C1  D28C              SETB  TR0           ; Timer 0 starten
                                            ; Interrupt-Freigaben
05C3  D2A9              SETB  ET0           ; Timer 0 Interrupt freigeben
05C5  C2B9              CLR   PT0           ; Timer 0 Interrupt auf niedriger Prioritäts-Ebene
05C7  D2AF              SETB  EA            ; Allgemeine Interruptfreigabe
05C9  80FE              SJMP  $             ; Endlosschleife
                              Timer0-Interrupt
0040            TI0_INT:                    ; Interruptroutine für den Timer0-Interrupt
0040  C0E0              PUSH  ACC           ; Inhalt des Akkus retten
0042  C0D0              PUSH  PSW           ; Programm-Status-Wort sichern
0044  E520              MOV   A, SCH_NR     ; Dreh-Schalter Nummer laden

0046  B42C0A            CJNE  A,#44, TI0_44 ; Springe, wenn nicht Programm 44
                                            ; Hier beginnt die eigentliche Interrupt-
                                            ; Verarbeitung
0049  758A00            MOV   TL0, #000H    ; LSB des Zähler 0 laden
004C  758C1F            MOV   TH0, #100H -0E1H; MSB des Zähler 0 laden
004F  0590              INC   P1            ; Inhalt des Ports P1 um 1 erhöhen
0051  8027              SJMP  TI0_END       ; Ende der Routine
0053            TI0_44:                     ; Nächste Routine

007A  D0D0      TI0_END: POP  PSW           ; Programm-Status-Wort wieder herstellen
007C  D0E0              POP   ACC           ; Original-Inhalt des Akkus wieder herstellen
007E  32                RETI                ; zurück zum Hauptprogramm
```

```
05CB            PRGM45:                         ; Timer 0 Interrupt 8-Bit-Zähler
                                                ; Dieses Unterprogramm erzeugt 4096
                                                ; Interrupts pro Sekunde
05CB 758902             MOV   TMOD, #00000010B  ; Timer0: Gate aus,
                                                ; Timer Mode 2 (8-Bit-Auto-Reload)
                                                ; Timer 1 keine Funktion
05CE 758C1F             MOV   TH0,#0100H -225   ; 4096 Interrupts pro Sekunde
05D1 D28C               SETB  TR0               ; Timer 0 starten
                                                ; Initialisierung des Software-Vorteilers
05D3 75D008             MOV   PSW, #008H        ; Registerbank 1 selektieren
05D6 7F00               MOV   R7, #000H         ; Softwarezähler initialisieren
05D8 75D000             MOV   PSW, #000H        ; Registerbank 0 selektieren Interrupt-Freigaben
05DB D2A9               SETB  ET0               ; Timer0-Interrupt freigeben
05DDC2B9                CLR   PT0               ; Timer0-Interrupt auf niedriger Prioritäts-Ebene
05DF D2AF               SETB  EA                ; Allgemeine Interruptfreigabe
05E1 80FE               SJMP  $                 ; Endlosschleife
                                  Timer0-Interrupt
0040            TI0_INT:                        ; Interruptroutine für den Timer 0 Interrupt
0040 C0E0               PUSH  ACC               ; Inhalt des Akkus retten
0042 C0D0               PUSH  PSW               ; Programm-Status-Wort sichern
0044 E520               MOV   A, SCH_NR         ; Dreh-Schalter Nummer laden

0053 B42D09             CJNE  A,#45, TI0_45     ; Springe, wenn nicht Programm 45
                                                ; Hier beginnt die eigentliche Interrupt-
                                                ; Verarbeitung
0056 75D008             MOV   PSW, #008H        ; Registerbank 1 selektieren
0059 DF02               DJNZ  R7, TI0_1E        ; R7 = R7 -1; Springe, wenn R7 ungleich 0
005B 1590               DEC   P1                ; Inhalt des Ports P1 um 1 vermindern
005D 801B       TI0_1E: SJMP  TI0_END           ; Ende der Routine
005F            TI0_45:                         ; Nächste Routine

007A D0D0 TI0_END: POP  PSW                     ; Programm-Status-Wort wieder herstellen
007C D0E0               POP   ACC               ; Original-Inhalt des Akkus wieder herstellen
007E 32                 RETI                    ; zurück zum Hauptprogramm
```

Bild 132: Das Testprogramm 45 generiert 4096 Interrupts pro Sekunde und zählt durch einen Software-Vorteiler den Inhalt des Ports P1 genau 16mal pro Sekunde abwärts.

Der Vorteil des Timer-Mode 1 liegt darin, daß sich durch den recht großen Zähler (16 Bit) relativ große zeitliche Abstände zwischen 2 Interrupts realisieren lassen. Nachteilig wirkt sich allerdings die höhere Ungenauigkeit durch das Neuladen des Zählers aus. In der Interrupt-Routine wird der Inhalt des Ports P1 jeweils um 1 erhöht.

6.4.3 Interrupt-Timer, 8-Bit-Zähler

Abbildung 132 zeigt das Testprogramm 45, welches 4096 Interrupts pro Sekunde generiert und durch einen Software-Vorteiler den Inhalt des Ports P1 genau 16mal pro Sekunde abwärts zählt. Der Timer 0 wird dazu in der Auto-Reload-Betriebsart 2 initialisiert, die nach dem Überlauf des niederwertigen Zählers automatisch den Inhalt des höherwertigen Zählers (TH0) übernimmt. Durch die Vorbesetzung des Zählers auf den Wert 31 (0100H - 225) werden genau 4096 (11,0592 MHz geteilt durch 12 • 225) Interrupts pro Sekunde ausgelöst.

Nach dem Setzen der Timerfunktion erfolgt die Anwahl der Registerbank 1 durch das Neuladen des Programmstatuswortes. In dieser durch die Interrupt-Routine verwendeten Registerbank wird zunächst der in der Interrupt-Routine benötigte Vorteiler initialisiert und anschließend auf die Registerbank 0 zurückgeschaltet.

Den Abschluß der Initialisierung bildet das Freigeben des Timer0-Interrupts, das Setzen der Prioritätsebene sowie die allgemeine Interrupt-Freigabe, wonach das Programm in eine Endlosschleife läuft.

Die Interrupt-Service-Routine TI0_INT wird 4096mal pro Sekunde durch den 8-Bit-Timer aufgerufen. Nachdem in der Routine die Registerbank 1 selektiert worden ist, wird durch den nachfolgenden DJNZ-Befehl der Inhalt des als Vorteiler verwendeten Registers R 7 um 1 dekrementiert. Hat das Register den Wert 0 erreicht, wird nicht zu dem angegebenen Ziel gesprungen, sondern der folgende DEC-P1-Befehl ausgeführt, der den Inhalt des Ports P1 um 1 herunterzählt, was genau 16mal (4096 : 256) pro Sekunde erfolgt.

Der Vorteil bei Verwendung der Betriebsart 2 liegt darin, daß durch die Auto-Reload-Funktion ein sehr genauer Teilungsfaktor zu erzielen ist. Nachteilig ist die häufigere Interrupt-Anforderung an den Prozessor, welches zu einer höheren Gesamtbelastung führt.

6.4.4. Zwei Timer-Interrupts

Abbildung 133 zeigt das Testprogramm 46, welches die beiden Timer 0 und 1 unabhängig voneinander verwendet. Beide Zähler werden in diesem Beispielprogramm im Mode 2 betrieben, wobei der Timer 0 mit einem Reload-Wert von 31 (0100H - 225) und Timer 1 mit einem Reload-Wert von 30 (0100H - 226) geladen werden. Damit generiert der Timer 0 4096 Interrupts pro Sekunde, während der Timer 1 durch den kleineren Reload-Wert nur ca. 4078 Interrupts pro Sekunde erzeugt.

Der Timer0-Interrupt soll die unteren 4 Bit (D 0 bis D 3) des Ports P1 hochzählen, während der Timer-Interrupt 1 die Bits D 4 bis D 7 ansteuert.

Es folgt die Initialisierung der Register R 6 und R 7 in den Registerbänken 1 und 2, die von den beiden Interrupt-Serivce-Routinen für den Timer0- bzw. Timer1-Interrupt verwendet werden. Es folgt die Interrupt-Freigabe und die Zuweisung der Interrupt-Prioritäten sowie die allgemeine Interrupt-Freigabe.

Für die beiden Timer-Inerrupts wurden gleiche Interrupt-Prioritäten gewählt, damit zu keinem Zeitpunkt während eines laufenden Interrupts der zweite den ersten unterbrechen kann. Da beide Routinen auf den gleichen Port P1 zugreifen, würde der quasi gleichzeitige Zugriff zu Funktionsstörungen führen.

Auch in diesem Programmbeispiel sind im Hauptprogramm keine weiteren Aktionen notwendig, weshalb auch dieses in einer Endlosschleife mündet.

Die Timer0-Interrupt-Routine TI0_INT selektiert nach der Ablage des Akkumulator- und PSW-Inhaltes auf dem Stack die Registerbank 1. Das Register 7 wird hierbei als Vorteiler durch 256 verwendet, während das Register 6 eine Teilung durch 16 (10H) vornimmt. Mit den nachfolgenden Befehlen wird der Inhalt der niederwertigen 4 Bit des Ports P1 um 1 erhöht.

```
05E3          PRGM46:                          ; Timer 0 und 1 Interrupt
                                               ; Dieses Unterprogramm erzeugt 4096
                                               ; Interrupts pro Sekunde
05E3 758922             MOV  TMOD, #00100010B ; Timer0 und 1: Gate aus,
                                               ; Timer, Mode 2 (8-Bit-Auto-Reload)
05E6 758C1F             MOV  TH0,#0100H -225  ; Timer 0: 4096 Interrupts pro Sekunde
05E9 D28C               SETB TR0              ; Timer 0 starten
05EB 758D1E             MOV  TH1,#0100H -226  ; Timer 1: ca. 4078 Interrupts pro Sekunde
05EE D28E               SETB TR1              ; Timer 1 starten
                                               ; Initialisierung der Software-Vorteiler
05F0 759000             MOV  P1, #000H        ; Alle LEDs aus
05F3 75D008             MOV  PSW, #008H       ; Registerbank 1 selektieren
05F6 7E10               MOV  R6, #010H        ; Vorteiler initialisieren
05F8 7F00               MOV  R7, #000H        ; Vorteiler initialisieren
05FA 75D010             MOV  PSW, #010H       ; Registerbank 2 selektieren
05FD 7E10               MOV  R6, #010H        ; Vorteiler initialisieren
05FF 7F00               MOV  R7, #000H        ; Vorteiler initialisieren
0601 75D000             MOV  PSW, #000H       ; Registerbank 0 selektieren Interrupt-Freigaben
0604 D2A9               SETB ET0              ; Timer0-Interrupt freigeben
0606 C2B9               CLR  PT0              ; Timer0-Interrupt auf niedriger Prioritäts-Ebene
0608 D2AB               SETB ET1              ; Timer1-Interrupt freigeben
060A C2BB               CLR  PT1              ; Timer1-Interrupt auf niedriger Prioritäts-Ebene
060C D2AF               SETB EA               ; Allgemeine Interruptfreigabe
060E 80FE               SJMP $                ; Endlosschleife
                                  Timer0-Interrupt
0040          TI0_INT:                         ; Interruptroutine für den Timer0-Interrupt
0040 C0E0               PUSH ACC              ; Inhalt des Akkus retten
0042 C0D0               PUSH PSW              ; Programm Status Wort sichern
0044 E520               MOV  A, SCH_NR        ; Dreh-Schalter Nummer laden

005F B42E18             CJNE A,#46, TI0_46    ; Springe, wenn nicht Programm 46
                                               ; Hier beginnt die eigentliche Interrupt-Verarbeitung
0062 75D008             MOV  PSW, #008H       ; Registerbank 1 selektieren
0065 DF11               DJNZ R7, TI0_46E      ; DEC und Springe, wenn R7 nicht 0
0067 DE0F               DJNZ R6, TI0_46E      ; DEC und Springe, wenn R6 nicht 0
0069 7E10               MOV  R6, #010H        ; Vorteiler initialisieren
006B E590               MOV  A, P1            ; bisherigen Inhalt von P1 lesen
006D 54F0               ANL  A, #11110000B    ; Bit 0..3 löschen
006F F8                 MOV  R0, A            ; A -> R0 Wert zwischenspeichern
0070 E590               MOV  A, P1            ; Inhalt von P1 lesen
0072 04                 INC  A                ; A = A +1
0073 540F               ANL  A, #00001111B    ; Bit 4..7 ausmaskieren
0075 48                 ORL  A, R0            ; Bit 4..7 mit Inhalt von R0 füllen
0076 F590               MOV  P1, A            ; P1 aktualisieren
0078 8000     TI0_46E:  SJMP TI0_END          ; Ende der Routine
007A          TI0_46:                          ; Nächste Routine

007A D0D0     TI0_END:  POP  PSW              ; Programm-Status-Wort wieder herstellen
007C D0E0               POP  ACC              ; Original-Inhalt des Akkus wieder herstellen
007E 32                 RETI                  ; zurück zum Hauptprogramm
                                  Timer1-Interrupt
007F          TI1_INT:                         ; Interruptroutine für den Timer1-Interrupt
007F C0E0               PUSH ACC              ; Inhalt des Akkus retten
0081 C0D0               PUSH PSW              ; Programm-Status-Wort sichern
0083 E520               MOV  A, SCH_NR        ; Dreh-Schalter-Nummer laden

0085 B42E1A             CJNE A,#46, TI1_46    ; Springe, wenn nicht Programm 46
                                               ; Hier beginnt die eigentliche Interrupt-
                                               ; Verarbeitung
0088 75D010             MOV  PSW, #010H       ; Registerbank 2 selektieren
008B DF13               DJNZ R7, TI1_46E      ; DEC und Springe, wenn R7 nicht 0
008D DE11               DJNZ R6, TI1_46E      ; DEC und Springe, wenn R6 nicht 0
008F 7E10               MOV  R6, #010H        ; Vorteiler initialisieren
0091 E590               MOV  A, P1            ; bisherigen Inhalt von P1 lesen
0093 540F               ANL  A, #00001111B    ; Bit 4..7 ausmaskieren
0095 F8                 MOV  R0, A            ; A -> R0 Wert zwischenspeichern
0096 E590               MOV  A, P1            ; Inhalt von P1 lesen
0098 C4                 SWAP A                ; Bit 0..3 mit Bit 4..7 tauschen
0099 04                 INC  A                ; A = A +1
009A C4                 SWAP A                ; Bit 0..3 mit Bit 4..7 tauschen
009B 54F0               ANL  A, #11110000B    ; Bit 0..3 löschen
009D 48                 ORL  A, R0            ; Bit 0..3 mit Inhalt von R0 füllen
009E F590               MOV  P1, A            ; P1 aktualisieren
00A0 80D8     TI1_46E:  SJMP TI0_END          ; Ende der Routine
00A2          TI1_46:                          ; Nächste Routine

00A2 D0D0     TI1_END:  POP PSW               ; Programm-Status-Wort wieder herstellen
00A4 D0E0               POP ACC               ; Original Inhalt des Akkus wieder herstellen
00A6 32                 RETI                  ; zurück zum Hauptprogramm
```

Bild 133: Das Testprogramm 46 verwendet unabhängig voneinander die Timer 0 und 1

Die Timer-Interrupt-Routine TI1_INT ist ähnlich der Timer0-Interrupt-Routine aufgebaut. Diese unterscheidet sich lediglich dadurch, daß hier die Registerbank 2 verwendet und der Zählerstand durch die 4 Leuchtdioden (D 4 bis D 7) von P1 angezeigt wird.

Nach dem Start des Programmes sind die Zählerstände, die durch die Leuchtdioden D 0 bis D 3 und D 4 bis D 7 angezeigt werden, scheinbar gleich. Wartet man allerdings einen Durchlauf ab (ca. 16 Sekunden), läßt sich bereits feststellen, daß die Timer0-Interrupt-Routine etwas häufiger aufgerufen wird, was sich dadurch bemerkbar macht, daß der durch die Leuchtdioden D 0 bis D 3 angezeigte Zählerstand geringfügig schneller hochzählt.

6.4.5 Serieller Interrupt

Abbildung 134 zeigt das Testprogramm 47, welches die über die serielle Schnittstelle empfangenen Daten interruptgesteuert einliest und auf den Port P1 ausgibt.

Die im zweiten Teil dieser Artikelserie vorgestellte Mikrocontroller-Grundschaltung erlaubt es, mit Hilfe des RS232C/V24-Treibers eine Verbindung zwischen der Schaltung und einem PC herzustellen. Dazu sind der auf der Platine befindliche Steckverbinder JP1 mit Hilfe eines Jumpers zu schließen und der PC mit einem 9poligen Sub-D-Verlängerungskabel (1 : 1) über seine serielle Schnittstelle anzuschließen. Die Schalterplatine ist für dieses Testprogramm zu entfernen.

Auf dem PC ist nach dem Laden des Betriebssystems ein Terminalprogramm zu starten, dessen Übertragungsparameter mit 9600 Baud, 8 Datenbits und einem Stoppbit einzustellen sind, um so die über die Tastatur eingegebenen Daten zu übertragen und deren ASCII-Code über die an P1 angeschlossenen Leuchtdioden anzuzeigen.

In dem vorliegenden Testprogramm wird zunächst der Port 1 gelöscht, um nach dem Start des Programms alle LEDs auszuschalten. Anschließend erfolgt die Initialisierung des Timers 1, dessen Teilungsfaktor so berechnet ist, daß eine Übertragungsrate von 9600 Baud an der seriellen Schnittstelle realisiert wird. Weitere Übertragungsparameter sind mit 8-Bit-Nutzdaten und einem Stoppbit festgelegt. Die folgenden Initialisierungen sind weitestgehend identisch mit den vorhergehenden Beispielprogrammen.

Nach dem Empfang eines seriellen Datenwortes löst der Mikrocontroller einen Interrupt aus, in dessen Routine zunächst,

```
0610            PRGM47:                         ; Serielle Schnittstelle Mode 1 8-Bit-Dateneingabe
0610 759000             MOV   P1, #000H         ; Alle LEDs aus
0613 758920             MOV   TMOD, #00100000B; Timer1: Gate aus,
                                                ; Timer Mode 2 (8-Bit-Auto-Reload)
                                                ; Timer 0 keine Funktion
0616 758DFD             MOV   TH1,#100H-3       ; Nachladewert 3 = 11,0592 MHz/12/16/2/
                                                ; 9600 Baud
0619 759850             MOV   SCON,#50H         ; Mode 1, (8Bit Daten) Receive Enable
061C 758700             MOV   PCON, #00H        ; SMOD =0, Teiler /2
061F D28E               SETB  TR1               ; Timer starten (TCON)
0621 D2AC               SETB  ES                ; Seriell-Interrupt freigeben
0623 C2BC               CLR   PS                ; Seriell-Interrupt auf niedriger Prioritäts-Ebene
0625 D2AF               SETB  EA                ; Allgemeine Interruptfreigabe
0627 80FE               SJMP  $                 ; Endlosschleife
                                  Serieller Interrupt
00A7            SER_INT:                        ; Interruptroutine für den seriellen Interrupt
00A7 C0E0               PUSH  ACC               ; Inhalt des Akkus retten
00A9 C0D0               PUSH  PSW               ; Programm-Status-Wort sichern
00AB E520               MOV   A, SCH_NR         ; Dreh-Schalter Nummer laden

00AD B42F12             CJNE  A,#47, SER_47     ; Springe, wenn nicht Programm 47
                                                ; Hier beginnt die eigentliche Interrupt-
                                                ; Verarbeitung
00B0 75D008             MOV   PSW, #008H        ; Registerbank 1 selektieren
                                                ; Zunächt wird das Empfangs- und anschl.
                                                ; das Sendeflag geprüft
00B3 309805             JNB   RI, SER_47A       ; Springe, wenn das Empfangsbit nicht gesetzt ist
00B6 C298               CLR   RI                ; Lösche Empfangsflag
00B8 859990             MOV   P1, SBUF          ; Seriell-Buffer lesen und den Inhalt auf P1 ausgeben
00BB            SER_47A:                        ; Prüfung des Sendeflags
00BB 309902             JNB   TI, SER_47E       ; Springe, wenn das Sendebit nicht gesetzt ist
00BE C299               CLR   TI                ; Lösche Sendeflag
                                                ; hier ggf. Sende-Behandlungsrouine
00C0 8018       SER_47E: SJMP SER_END           ; Ende der Routine
00C2            SER_47:                         ; Nächste Routine

00DA D0D0       SER_END: POP  PSW               ; Programm-Status-Wort wieder herstellen
00DC D0E0               POP   ACC               ; Original-Inhalt des Akkus wieder herstellen
00DE 32                 RETI                    ; zurück zum Hauptprogramm
```

Bild 134: Das Testprogramm 47 liest die über die serielle Schnittstelle empfangenen Daten interruptgesteuert ein und gibt sie auf den Port P 1 aus.

wie bereits beschrieben, die Sicherung der Register sowie das Umschalten auf die benötigte Registerbank erfolgt. Da die Programmunterbrechung sowohl nach dem Senden eines seriellen Datenwortes als auch nach dem Empfang ausgelöst wird, ist in der Interrupt-Routine zu prüfen, welche Interruptquelle die Programmunterbrechung ausgelöst hat.

In unserem Beispiel wird nach der Detektierung eines Empfangsinterrupts das RI-Flag gelöscht, der serielle Datenbuffer ausgelesen und dessen Inhalt auf den Port P1 ausgegeben. Anschließend erfolgt die Überprüfung des Sende-Interrupt-Bits TI, welches gegebenenfalls gelöscht wird. Den Abschluß der Interrupt-Service-Routine bildet, wie bei den anderen Beispielprogrammen auch, das Zurücksichern der Register und die anschließende Ausführung des Befehls RETI.

6.4.6 Serieller Interrupt mit Daten senden

Abbildung 135 zeigt die Weiterentwicklung des in Abbildung 134 beschriebenen Testprogrammes, dessen Initialisierung bzw. Hauptprogramm übernommen wurde. Die Interrupt-Verarbeitungs-Routine unterscheidet sich lediglich darin, daß das empfangene Datum aus dem Empfangsbuffer ausgelesen und sofort wieder in den Sendebuffer zurückgeschrieben wird, welches das über das angeschlossene Terminalprogramm übertragene Zeichen zurücksendet.

Am Ende der Sende-Datenübertragung löst die serielle Schnittstelle des Mikrocontrollers einen Sende-Interrupt aus, der üblicherweise für die Übergabe des nächsten Zeichens an den Sendebuffer genutzt wird. Das vorliegende Beispiel gibt das empfangene bzw. gesendete Datenwort zur Kontrolle an den Port P1 aus.

Im siebzehnten Teil der Mikrocontroller-Grundlagenserie beschreiben wir zunächst ausführlich den I^2C-Bus, gefolgt von mehreren Beispielen, die die Ansteuerung zeigen. ELV

```
0629            PRGM48:                         ; Serielle Schnittstelle Mode 1 8Bit Datenein- und
                                                ; ausgabe
0629 759000             MOV   P1, #000H         ; Alle LEDs aus
062C 758920             MOV   TMOD, #00100000B; Timer1: Gate aus,
                                                ; Timer Mode 2 (8 Bit Auto-Reload)  Timer 0 keine
                                                ; Funktion
062F 758DFD             MOV   TH1,#100H-3       ; Nachladewert 3 = 11,0592 MHz/12/16/2/9600 Baud
0632 759850             MOV   SCON,#50H         ; Mode 1, (8Bit Daten) Receive Enable
0635 758700             MOV   PCON, #00H        ; SMOD =0, Teiler /2
0638 D28E               SETB  TR1               ; Timer starten (TCON)
063A D2AC               SETB  ES                ; Seriell-Interrupt freigeben
063C C2BC               CLR   PS                ; Seriell-Interrupt auf niedriger Prioritäts-Ebene
063E D2AF               SETB  EA                ; Allgemeine Interruptfreigabe
0640 80FE               SJMP  $                 ; Endlosschleife
                                  Serieller Interrupt
00A7            SER_INT:                        ; Interruptroutine für den seriellen Interrupt
00A7 C0E0               PUSH  ACC               ; Inhalt des Akkus retten
00A9 C0D0               PUSH  PSW               ; Programm-Status-Wort sichern
00AB E520               MOV   A, SCH_NR         ; Dreh-Schalter Nummer laden

00C2 B43015             CJNE  A,#48, SER_48     ; Springe, wenn nicht Programm 48
                                                ; Hier beginnt die eigentliche Interrupt-Verarbeitung
00C5 75D008             MOV   PSW, #008H        ; Registerbank 1 selektieren
                                                ; Zunächt wird das Empfangs- und anschl.
                                                ; das Sendeflag geprüft
00C8 309806             JNB   RI, SER_48A       ; Springe, wenn das Empfangsbit nicht gesetzt ist
00CB C298               CLR   RI                ; Lösche Empfangsflag
00CD AF99               MOV   R7, SBUF          ; Seriell-Buffer lesen und in R7 zwischenspeichern
00CF 8F99               MOV   SBUF, R7          ; Ausgabe über die serielle Schnittstelle (Echo)
00D1            SER_48A:                        ; Prüfung des Sendeflags
00D1 309904             JNB   TI, SER_48E       ; Springe, wenn das Sendebit nicht gesetzt ist
00D4 C299               CLR   TI                ; Lösche Sendeflag
00D6 8F90               MOV   P1, R7            ; Inhalt des gesendeten Bytes auf P1 ausgeben
00D8 8000       SER_48E: SJMP SER_END           ; Ende der Routine
00DA            SER_48:                         ; Nächste Routine

00DA D0D0       SER_END: POP PSW                ; Programm-Status-Wort wieder herstellen
00DC D0E0               POP ACC                 ; Original-Inhalt des Akkus wieder herstellen
00DE 32                 RETI                    ; zurück zum Hauptprogramm
```

Bild 135 zeigt die Weiterentwicklung des in Bild 134 beschriebenen Testprogramms, dessen Initialisierung bzw. Hauptprogramm übernommen wurde.