MSP430 – Intelligenter Stromsparer Teil 3

Im vorangegangenen Teil der Serie über die MSP430-Reihe hatten wir damit begonnen, den MSP430-Befehlssatz anhand eines Beispieles – einer Fahrstuhlsteuerung – zu beschreiben. Jetzt wollen wir das Projekt vervollständigen. Für die (experimentelle) praktische Realisierung des Projekts stellen wir, neben dem Emulation Tool von TI, ein Demoboard vor, das neben der Spannungsversorgung auch alle wichtigen Elemente für die Ein- und Ausgabe enthält.

Allgemeines

Nachdem wir bisher mit dem Lauflicht und der Ansteuerung einer 7-Segment-Anzeige mittels des MSP430-Mikrocontrollers vor allem die Ausgabeseite des Projekts betrachtet haben, folgt nun die Vervollständigung des Fahrstuhl-Steuerprogramms mit der Eingabe über Taster sowie der Generierung eines akustischen Signals.

Die kleine Artikelserie über die MSP430-Reihe wird schließlich mit der Vorstellung des passenden „Emulation Toolkits" von TI und einem dazu passenden Demoboard abgeschlossen. Wollen wir uns also zunächst der Fortführung des Fahrstuhl-Steuerprogramms zuwenden.

Eingänge

Mikrocontroller arbeiten nach dem EVA-Prinzip. Dies bedeutet, dass sich die Ausführung des Mikrocontroller-Programms in die drei Phasen Eingabe, Verarbeitung und Ausgabe gliedert. Verarbeitung sowie Ausgabe haben wir im bisherigen Verlauf der Artikelserie bereits ausführlich diskutiert. Jetzt geht es an die Realisierung der Eingabefunktionen.

Alle digitalen Ein-/Ausgänge (Digital I/Os) sind unabhängig voneinander als Eingang oder Ausgang verwendbar. Die einzelnen Bits des Control-Registers P5DIR definieren, welcher Pin Eingang und welcher Ausgang ist. In Abbildung 8 erkennt man, dass eine „0" beispielsweise an der Bitposition 7 den entsprechenden Pin (P5.7) als Eingang definiert. Eine „1" hingegen hat zur Folge, dass der entsprechende Pin als Ausgang dient (z. B. Bitposition 0 => P5.0).

Am einfachsten kann man das Ereignis eines Tastendruckes mit Hilfe von Interrupts handhaben. Das heißt, sobald eine Taste gedrückt ist, wird aufgrund der dadurch generierten Flanke am Eingangspin die Programmabarbeitung unterbrochen, die momentane Position im Programm zwischengespeichert und dann an eine speziell für dieses Ereignis definierte Adresse (Sprungmarke) gesprungen. Die Sprungmarke wird in der Interrupt-Vektortabelle definiert, die die unterschiedlichsten Quellen der Unterbrechung (des Interrupts) enthält. In unserem Beispielprogramm (Programm-Listing 3) sehen Sie, dass dort in der Tabelle bei der Adresse FFE8h die Sprungmarke für alle Port-P1-Pins angegeben ist. In der Interrupt-Serviceroutine ISR_PortP1 wird dann durch Testen des PIFG-Control-Registers ermittelt, welcher Pin diesen Interrupt generiert hat. In der Interrupt-Serviceroutine muss dabei unbedingt auch dieses Flag (P1IFG) wieder zurückgesetzt werden (BIC.b #02h,&P1IFG), um einen erneuten Tastendruck erkennen zu können.

Zu beachten ist auch, dass im Hauptprogramm die Interrupts der einzelnen peripheren Module aktiviert werden müssen (z.B. MOV.b #01Eh,&P1IE). Dies reicht jedoch noch nicht aus, denn mittels des

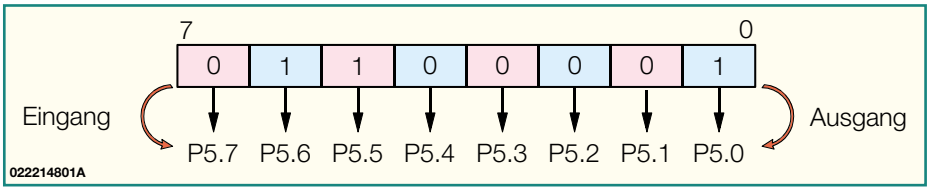


Bild 8: Port Direction Control Register P5DIR

Programm Listing 3: Eingabe über Taster

```
;*** Programm Beispiel: Tastatur
#include "msp430x14x.h"
;--- Hauptprogramm
    RSEG CODE
Start    MOV    #0A00h,SP        ; Initialisiere Stack Pointer
         MOV    #05A80h,&WDTCTL  ; stoppen des Watchdog
         MOV.b  #0E1h,&P1DIR     ; P1.1 bis P1.4 sind Eingaenge
         MOV.b  #01Eh,&P1IE      ; Interrupts P1.1 bis P1.4 erlauben
         MOV.b  #0h,&P5OUT
         MOV.b  #0FFh,&P5DIR     ; P5.0 bis P5.7 sind Ausgaenge
         EINT                   ; Interrupts erlauben
Schleife JMP    Schleife         ; Endlosschleife
;
;--- Unterprogramm
warte    MOV    #5000h,R5        ; Warteschleife
warte1   DEC    R5
         JNZ    warte1
         RET
;
;--- Interrupt Service Routinen
ISR_PortP1
          MOV.b &P1IFG,&P5OUT
          CALL  #warte
          BIT.b #02h,&P1IFG     ; P1.1 hat Interrupt generiert?
          JC    ISR_P11
          BIT.b #04h,&P1IFG     ; P1.2 hat Interrupt generiert?
          JC    ISR_P12
          BIT.b #08h,&P1IFG     ; P1.3 hat Interrupt generiert?
          JC    ISR_P13
          BIT.b #10h,&P1IFG     ; P1.4 hat Interrupt generiert?
          JC    ISR_P14
          RETI
ISR_P11   BIC.b #02h,&P1IFG     ; loesche P1.1 Interrupt Flag
          JMP   ISR_PortP1
ISR_P12   BIC.b #04h,&P1IFG     ; loesche P1.2 Interrupt Flag
          JMP   ISR_PortP1
ISR_P13   BIC.b #08h,&P1IFG     ; loesche P1.3 Interrupt Flag
          JMP   ISR_PortP1
ISR_P14   BIC.b #10h,&P1IFG     ; loesche P1.4 Interrupt Flag
          JMP   ISR_PortP1

ISR_Dummy RETI

;--- Interrupt Vektor Tabelle
    RSEG INTVEC
     DW  ISR_Dummy    ; FFE0h: nicht benutzt
     DW  ISR_Dummy    ; FFE2h: Port P2
     DW  ISR_Dummy    ; FFE4h: USART1 senden
     DW  ISR_Dummy    ; FFE6h: USART1 empfangen
     DW  ISR_PortP1   ; FFE8h: Port P1
     DW  ISR_Dummy    ; FFEAh: Timer_A3 (CCIFG1, CCIFG2, TAIFG)
     DW  ISR_Dummy    ; FFECh: Timer_A3 (CCIFG0)
     DW  ISR_Dummy    ; FFEEh: ADC
     DW  ISR_Dummy    ; FFF0h: USART1 senden
     DW  ISR_Dummy    ; FFF2h: USART0 empfangen
     DW  ISR_Dummy    ; FFF4h: Watchdog Timer
     DW  ISR_Dummy    ; FFF6h: Comparator_A
     DW  ISR_Dummy    ; FFF8h: Timer_B7 (BCCIFG1-BCCIFG6, TBIFG)
     DW  ISR_Dummy    ; FFFAh: Timer_B7 (BCCIFG0)
     DW  ISR_Dummy    ; FFFCh: NMI, Oszillator Fehler, Flash
     DW  Start        ; FFFEh: externer Reset, Watchdog, Power-Up

     end
```

„General Interrupt Enable“-Bits (GIE), das sich im Statusregister SR befindet, können, unabhängig von den einzelnen Moduleinstellungen, die Interrupts dieser Module durch Zurücksetzen des GIE-Bits abgeschaltet werden. Konkret heißt dies nun, dass sowohl das GIE-Bit als auch die Interrupt-Enable-Bits innerhalb der jeweiligen Peripheriemodule zu setzen sind, um einen Interrupt aktivieren zu können.

Es piepst

Auf der noch später beschriebenen Experimentierplatine befindet sich auch ein Summer. Dieser Summer könnte im Prinzip mit dem MSP430 einfach über das schnelle Umschalten eines digitalen Ausgangs dazu gebracht werden, einen Piepton zu erzeugen. Hier soll aber ein anderer Weg gezeigt werden. Die Stärke von Mikrocontrollern sind die vorhandenen, integrierten Peripheriemodule. Der MSP430 hat beispielsweise ein mächtiges Timer-Modul – den Timer_A. Dieser ist mittels Software konfigurierbar und kann unabhängig von der CPU Taktsignale oder Pulse generieren. Dies soll nun Verwendung finden, um den erwähnten Summer anzusteuern.

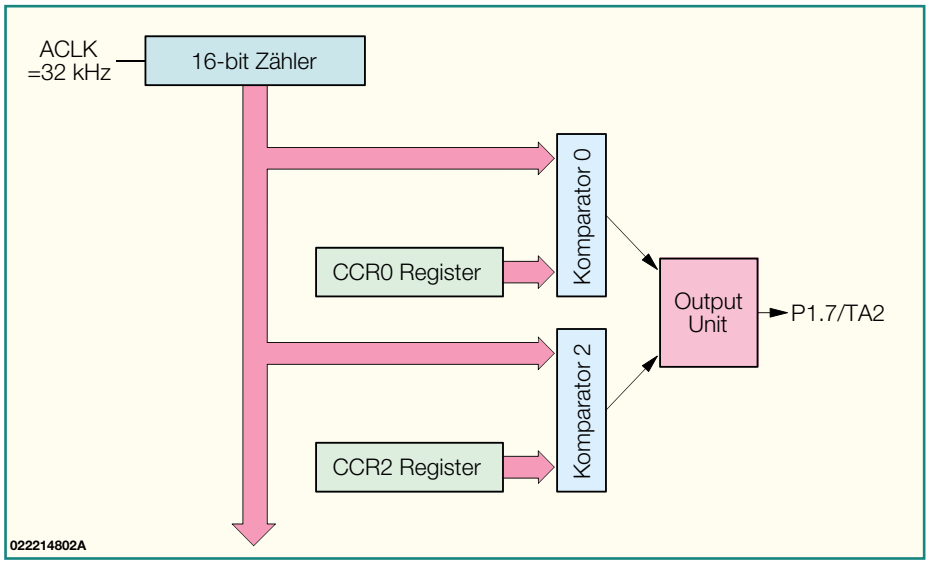


Bild 9: Timer_A-Module eines MSP430-Mikrocontrollers

Der Timer_A wird in unserem Fall vom 32-kHz-Oszillator des MSP430 getaktet. Es ist dabei in der praktischen Umsetzung auf der Experimentierplatine wichtig, dass auf dem Sockelboard des „Flash Emulation Tools" auch der 32-kHz-Quarz bestückt wurde. Beachten Sie bitte, dass dieser auf einem neuen Tool nicht bestückt ist!

Der 32-kHz-Takt wird, was wohl am Naheliegendsten ist, mit dem Zähler durch 11 geteilt. Dies erfolgt, indem von 0 bis 0 Ah gezählt und danach ein Ausgangspin getoggelt wird. Das Toggeln kann hierbei die im Timer_A-Modul bzw. in dessen Capture-/Compare-Modul enthaltene „Output Unit" übernehmen.

Im Bild 9 ist schematisch gezeigt, wie mittels der beiden Komparatoren der beiden Capture-/Compare-Blöcke die Inhalte der CCR0- und CCR2-Register mit dem momentanen Zählerstand verglichen werden. Sobald Zählerstand und Register identisch sind, ändert sich der Ausgang des Komparators auf „1". Die „Output Unit" wertet nun die Komparatorausgangssignale aus. Hierbei wird bei dem gewählten „PWM reset/set" Mode der Pin P1.7/TA2 zurückgesetzt, wenn der Ausgang von Komparator 2 den Zustand „1" annimmt. Ist der Ausgang von Komparator 0 dagegen „1", so wird das Signal am Pin P1.7/TA2 gesetzt.

Beachten Sie, dass fast alle Pins des MSP430 im Multiplex-Betrieb angesteuert werden. Das heißt, jeder digitale „I/O" kann auch eine „Modul-Funktion" übernehmen. In dem gerade beschriebenen Timer_A-Beispiel haben wir diese „Modul-Funktion" verwendet. Um den im Multiplex gesteuerten Pin als Modul-Pin zu verwenden, muss dies im P1SEL- Register auch definiert werden. Die Umschaltung in unserem Beispiel mit dem Summer geschieht einfach mittels des Befehles MOV.b #80h,&P1SEL. Nicht vergessen werden sollte auch die Definition, ob es sich um einen Eingang oder Ausgang handelt. Dies wird wieder im Register P1DIR festgelegt.

Im Programm-Listing 4 ist das lauffähige Programm für die Ansteuerung des Summers gezeigt.

Fahrstuhlsteuerung

Nachdem nun alle Einzelprogramme der Fahrstuhlsteuerung entwickelt sind, müssen diese einzelnen Teile abschließend zu einem Gesamtprogramm für die Fahrstuhlsteuerung zusammengefügt werden. Auf dessen Abdruck verzichten wir hier aus Platzgründen, es ist unter [1] ebenso zum Download zu finden wie die einzelnen Beispielprojekte.

Damit wollen wir an die experimentelle praktische Realisierung des Steuerungsprojekts gehen.

Experimentieren einfach gemacht

Die Firma Texas Instruments bietet ein „Emulation Toolkit" an, das aus einer Leiterplatte mit einem IC-Sockel, zwei Mikrocontrollern vom Typ MSP 430F149, einem JTAG-Programmieradapter sowie einer Daten-CD besteht. Da die Mikrocontroller aufgrund des QFP-Gehäuses und des geringen Abstandes zwischen den Pins nicht ganz einfach für Testaufbauten einsetzbar sind, ist das Emulation-Tool sehr hilfreich. Hier kann der Mikrocontroller einfach in den IC-Sockel eingesetzt werden und alle Pins stehen an einer Buchsenleiste gut erreichbar zur Verfügung.

Das Emulation-Tool ist bereits fertig vormontiert, lediglich die Buchsenleisten sind noch einzulöten. Diese werden von der Unterseite der Leiterplatte her bestückt und auf der Oberseite verlötet. Der 2 x 7-polige Wannenstecker auf dem Emulation-Tool dient der Verbindung mit dem JTAG-Programmieradapter, der wiederum die Programmierung über den PC ermöglicht.

Die Schaltbilder des Emulation-Tools und des Programmieradapters befinden sich auf der dem Kit beiliegenden CD und geben Einblick in dessen Funktionsweise.

Auf dem Emulation-Tool sind keine Oszillatoren vorhanden, da der Controller bereits über interne RC-Oszillatoren verfügt. Soll z. B. eine Anwendung mit einer Echtzeituhr realisiert werden, ist der Einsatz eines externen Quarzes, der auf einer festen Frequenz schwingt, erforderlich. Dazu befindet sich im Bausatz des weiter unten beschriebenen Demoboards ein 32,768-kHz-Quarz, der auf dem Emulation-Tool bestückt werden kann. Die entsprechenden Lötpads, auf die Quarze in verschiedenen Bauformen und die zugehörigen Kondensatoren bestückt werden können, befinden sich bereits auf der Platine des Emulation-Tools.

Der 32,768-kHz-Uhrenquarz wird an Pin 8 (XIN) und Pin 9 (XOUT) angeschlossen, die Pads befinden sich direkt neben dem Testsockel. Hilfreich ist hierbei die aufgedruckte Nummerierung der Pins auf dem Emulation-Tool. Für den angegebenen Uhrenquarz braucht man keine Kondensatoren zu bestücken, da diese bereits im Controller integriert sind.

Um das Emulation-Tool auch praktisch einsetzen zu können, haben wir ein Demoboard entwickelt, das alle wichtigen Elemente enthält, um Programmier-Experimente ausführen zu können.

Das MSP430-Demoboard

Das MSP430-Demoboard ist als Erweiterung für das Emulation-Tool gedacht und eignet sich besonders gut für Schulungszwecke, da auf dem Board bereits die Spannungsversorgung für das Emulation-Tool und einige Zusatzkomponenten für die Ein- und Ausgabe, so z. B. Taster, Summer und LEDs untergebracht sind. Das in dieser Artikelserie vorgestellte Programm zur Fahrstuhlsteuerung ist auf die Konfiguration des Demoboards abgestimmt und nutzt dessen Taster und Anzeigen.

Das Schaltbild des Demoboards ist in Abbildung 10 dargestellt.

Die Spannungsversorgung erfolgt über die Klinkenbuchse BU 1. Hier wird eine Gleichspannung im Bereich von 8 bis 15 V eingespeist.

Die Diode D 1, eine 1N4001, schützt die Schaltung vor versehentlicher Verpolung der Betriebsspannung. Der Spannungsregler IC 1 vom Typ LM317 stabilisiert die Versorgungsspannung auf eine konstante Höhe von 3,1 V für das Emulation-Tool. Die Widerstände R 1 und R 2 dienen dabei

```
;*** Programm Beispiel: Summer
#include "msp430x14x.h"
;--- Hauptprogramm
        RSEG CODE
Start    MOV   #0A00h,SP        ; Initialisiere Stack Pointer
         MOV   #05A80h,&WDTCTL  ; stoppen des Watchdog
         MOV   #0110h,&TACTL    ; Timer_A konfigurieren:
                                ;    - Takt= ACLK (=32kHz)
                                ;    - up to CCR0 Mode
         MOV   #20E0h,&CCTL2    ; Capture/Compare Block 2:
                                ;    - Output Unit: PWM reset/set
mode
         MOV   #000Ah,&CCR0     ; zaehle bis 0Ah
         MOV   #0005h,&CCR2     ; Duty Cycle=50%
         MOV.b #80h,&P1DIR      ; P1.7 ist Ausgang
         MOV.b #80h,&P1SEL      ; Pin P1.7 wird in Module Funktion
benutzt
Schleife JMP   Schleife
;
;--- Interrupt Vektor
        ASEG  0FFE0h+RESET_VECTOR
        DW  Start     ; Reset Interrupt Vektor

        end
```

Programm Listing 4: Beispiel für Frequenzgenerierung für Summer

zur Einstellung der Ausgangsspannung. Die Leuchtdiode D 11 zeigt die vorhandene Betriebspannung an.

Die Portpins des Mikrocontrollers sind über das Emulation-Tool auf die Stiftleisten des Demoboards geführt. An die Portpins P1.1 bis P1.4 sind die Taster TA1 bis TA4 angeschlossen, die zu Eingabezwecken nutzbar sind. Die Portpins P5.0 bis P5.7 sind über Vorwiderstände mit den Leuchtdioden D3 bis D10 verbunden und können für die Ausgabe optischer Signale zum Einsatz kommen. Um auch eine Ziffer, in unserem Programmbeispiel das aktuelle Stockwerk, anzeigen zu können, sind die einzelnen Segmente der 7-Segment-Anzeige DI 1 mit den Ports P 4.0 bis P 4.7 verbunden. Zur Ausgabe von Tonsignalen schließlich ist der Sound-Transducer PZ1 nutzbar, der über den Transistor T1 angesteuert wird.

Die freien Portpins des Mikrocontrollers sind auf die seitlichen Pads der Lochrasterfläche geführt und dort für weitere Applikationen verfügbar. Die zugehörige Portbezeichnung ist neben den Pads aufgedruckt.

Nachbau

Die Schaltung des MPS430-Demoboards ist auf einer doppelseitigen Leiterplatte mit den Abmessungen 140 x 185 mm untergebracht.

Die Bestückung erfolgt in konventioneller Technik anhand des Bestückungsdrucks und der Stückliste.

Sie beginnt mit den Widerständen, Dioden (polrichtige Bestückung beachten, Ringaufdruck=Katode), Kondensatoren (außer Elkos), gefolgt vom Jumper JP 1 und den Stiftleisten ST 1 bis ST 4. Bei deren Bestückung ist darauf zu achten, dass sie gerade eingesetzt werden, damit das Emulation Tool später genau auf die Stiftleisten passt.

Nun folgt die Bestückung von T 1 (Einbaulage laut Layout), DI 1 (Lage des Dezimalpunkts laut Bestückungsdruck beachten), der Elkos (polrichtig! Minuspol am Gehäuse markiert), von PZ 1 (Polung beachten) sowie der Klinkenbuchse BU 1.

Der Spannungsregler IC 1 wird jetzt

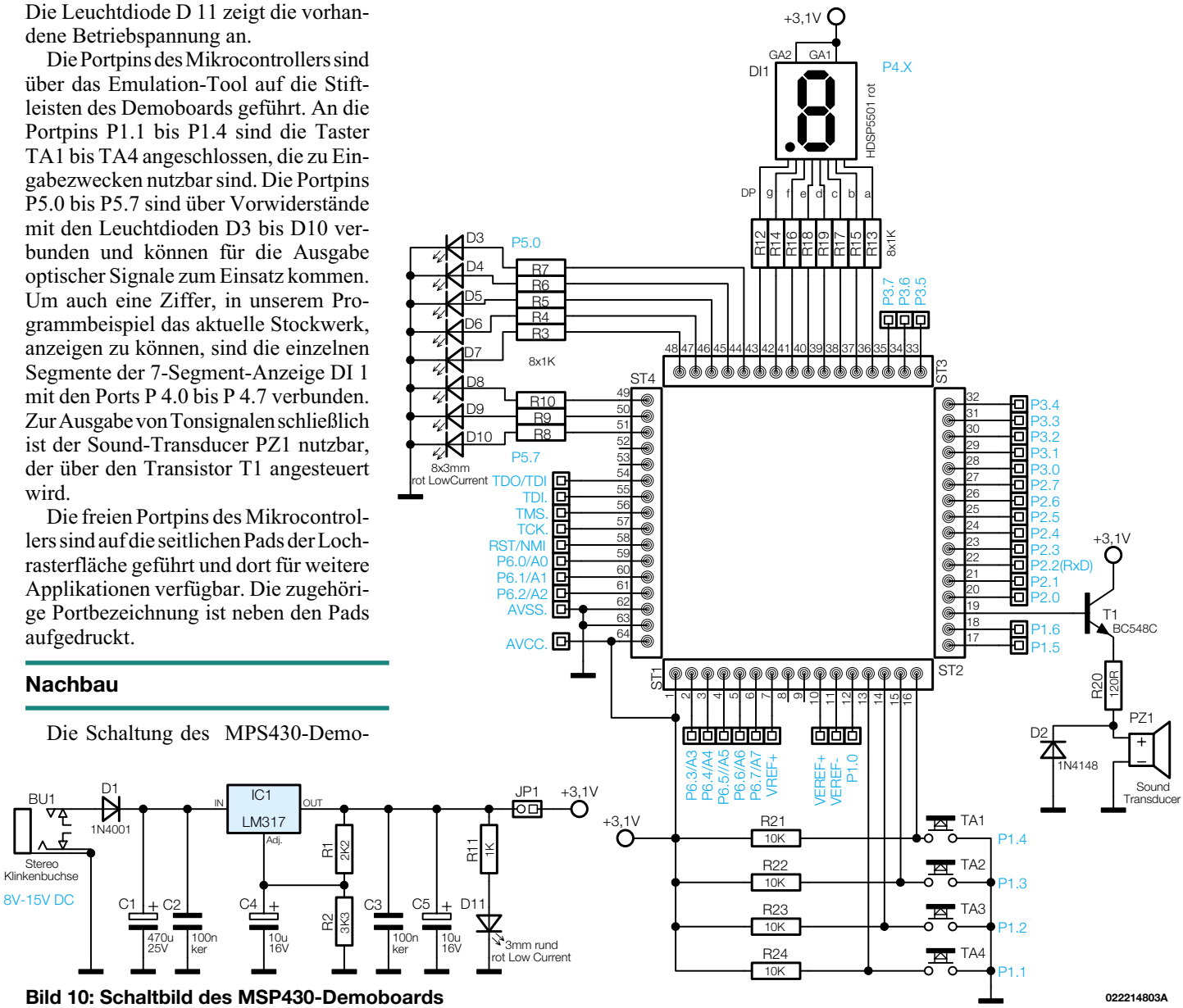
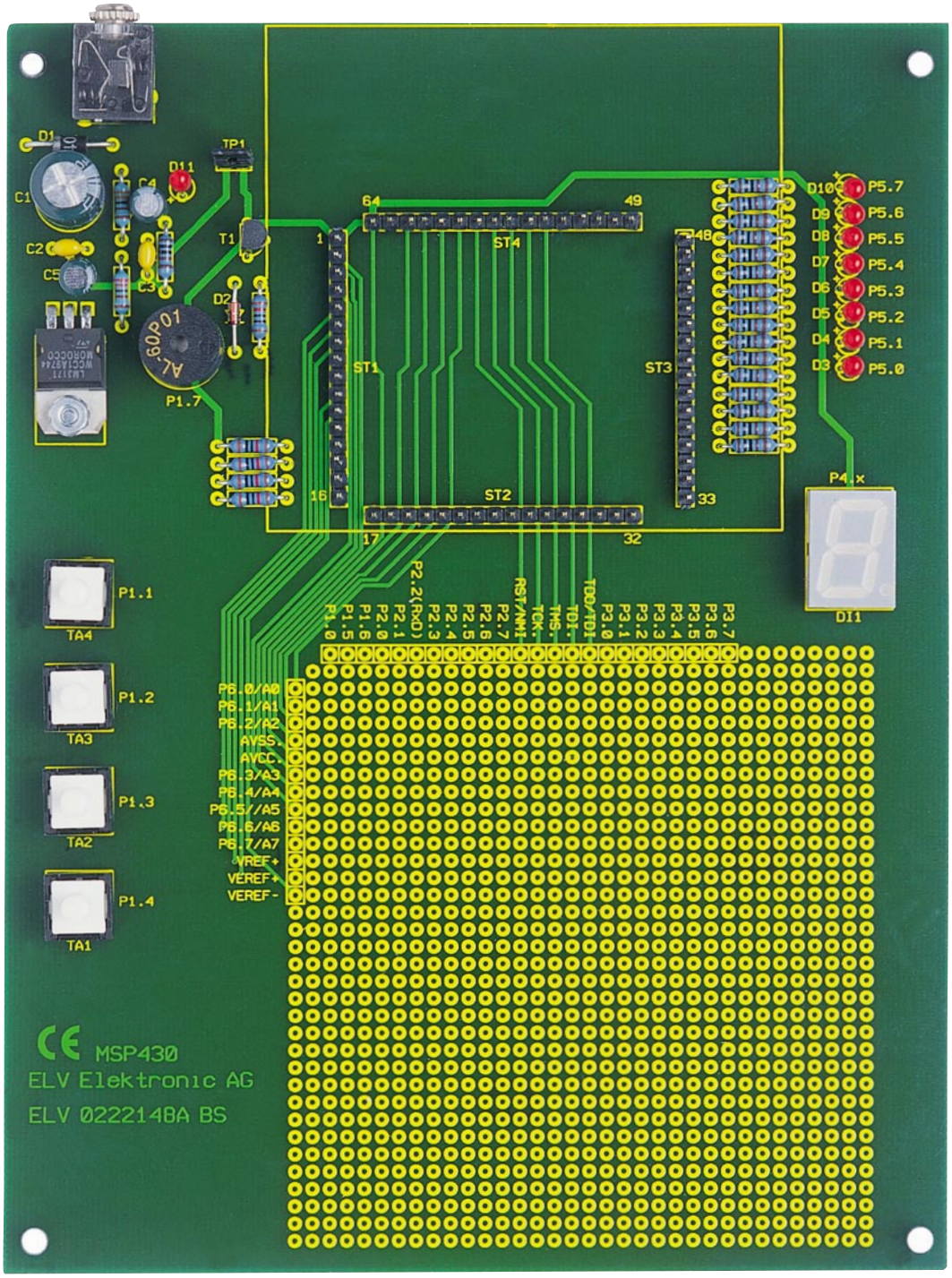


Bild 10: Schaltbild des MSP430-Demoboards

Ansicht der fertig bestückten Platine des MSP430-Demoboards



Stückliste: MSP430-Demoboard

Widerstände:

120Ω	R20
1kΩ	R3-R19
2,2kΩ	R1
3,3kΩ	R2
10kΩ	R21, R22, R23, R24

Kondensatoren:

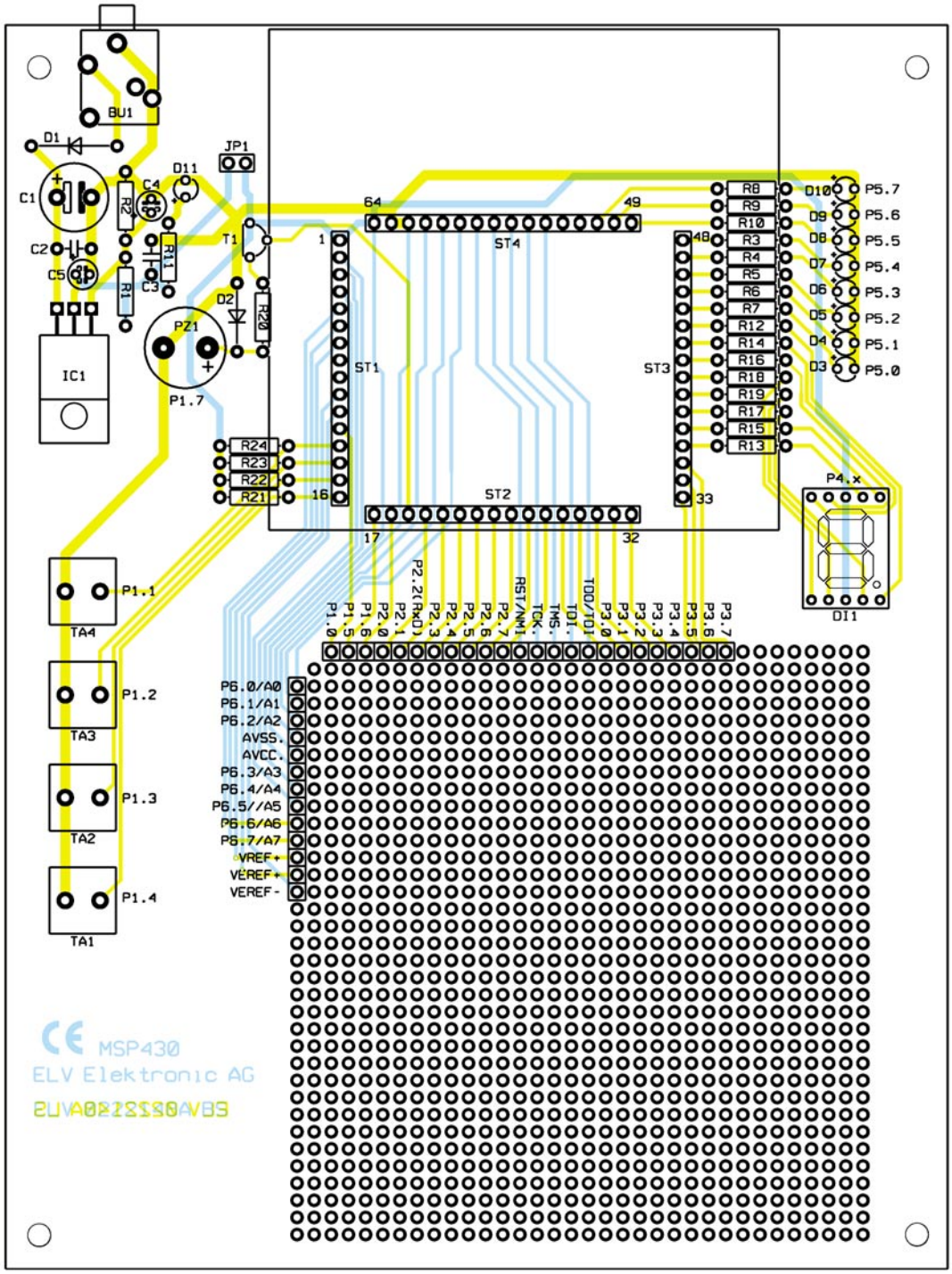
100nF/ker	C2, C3
10µF/25V	C4, C5
470µF/25V	C1

Halbleiter:

LM317	IC1
BC548C	T1
1N4001	D1
1N4148	D2
LED, 3 mm, rot, low current	D3-D11
7-Segment-Anzeige, HDSP-5501, rot	DI1

Sonstiges:

Klinkenbuchse, 3,5 mm, print, stereo	BU1
ELV-Print-Taster, 1 x ein, 15 mm	TA1-TA4
Stiftleisten, 1 x 16-polig	ST1-ST4
Stiftleiste, 1 x 2-polig	JP1
Sound-Transducer, ST2	PZ1
1 Jumper	
1 Zylinderkopfschraube, M3 x 8 mm	
1 Mutter, M3	
1 Fächerscheibe, M3	
4 Gummi-Gehäusefüße, zylindrisch	



Bestückungsdruck des MSP430-Demoboards

liegend montiert (siehe auch Platinenfoto) und vor dem Verlöten der Anschlüsse mit einer M3x8-mm-Zylinderkopfschraube, einer Fächerscheibe und einer M3-Mutter verschraubt.

Nach dem Bestücken der Taster sind abschließend die LEDs mit einem Abstand von 5 mm, gemessen zwischen Leiterplattenoberfläche bis zur Unterseite des LED-Gehäusekörpers, zu bestücken (ebenfalls polrichtig, langer Anschluss ist die Anode).

Damit das Demoboard auf eine Unterlage gelegt werden kann, ohne diese zu verkratzen bzw. Kurzschlüsse zu erzeugen, sind an den Ecken der Leiterplattenunterseite vier selbstklebende Gummifüße zu montieren.

Jetzt ist das Demoboard komplett aufgebaut und das Emulation Tool kann in der im Bestückungsplan eingezeichneten Lage auf die Steckerleisten des Demoboards aufgesteckt werden.

Einsatzhinweise

Der Jumper JP1 auf dem Demoboard trennt die Spannungsversorgung des Mikrocontrollers auf und ermöglicht über die beiden Pins die Messung der Stromaufnahme – im Normalfall muss der Jumper aufgesteckt sein.

Das Demoboard ist mit einer freien Lochraster-Fläche ausgestattet, auf der man zusätzliche Schaltungskomponenten bzw. eigene Applikationen unterbringen kann. Die freien Ports des Mikrocontrollers sind dazu auf jeweils beschriftete Lötpads geführt.

Die Programmierung des Mikrocontrollers erfolgt über den mit dem Emulation Toolkit gelieferten JTAG-Programmieradapter nach der dem Kit beiliegenden Anleitung.

Damit steht ein komplettes Programmier- und Applikationswerkzeug zur Verfügung, das die Erschließung der Anwendungsmöglichkeiten der vielseitigen MSP430-Mikrocontroller einfach macht.