Aufgabe 1

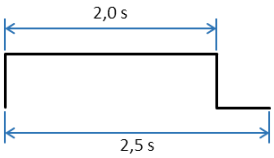
Lernziele: Machen Sie sich mit der Entwicklungsumgebung vertraut. Sie besteht einem Eval-Board mit einem MSP430FR5729-Mikrocontroller, einem Download-/Debugger-Adapter und dem Code Composer Studio von Texas Instruments. In dieser Aufgabe sollen Sie lernen

a) wie die Grundstruktur einer Applikation in C aufgebaut ist,
b) wie man GPIOs des MSP430 als Eingänge/Ausgänge konfiguriert,
c) wie man eine Leuchtdiode mit Timer-Interrupts blinken lässt,
d) wie man Taster mit Timer-Interrupts entprellt.

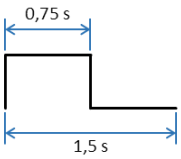
Aufgabenstellung

Das Eval-Board ist u.a. mit zwei Leuchtdioden LED1 und LED2 sowie mit zwei Tastern (Buttons) BTN1 und BTN2 bestückt. In eingebetteten Systemen benutzt man Leuchtdioden häufig dazu, Warnungen oder interne Fehlzustände durch verschiedene Blinkmuster anzuzeigen. In unserem System soll die Leuchtdiode LED2 zu solchen Zwecken verwendet werden. Mit Hilfe dieser Leuchtdiode soll es möglich sein, ein von sechs möglichen Blinkmustern anzeigen zu lassen. Die unteren Impulsdiagramme stellen zeitliche Anhängigkeiten in Leucht- und Dunkelphasen der einzelnen Blinkmuster dar. Die Blinkmuster sind in dieser Darstellung von (1) bis (6) durchnummeriert und mit passenden Namen versehen. Leuchtphasen sind in den Impulsdiagrammen mit dem High-Pegel dargestellt, Dunkelphasen mit dem Low-Pegel. Beispielsweise hat das Blinkmuster (5) „2 x Blinken“ eine Periode von insgesamt 3,5 Sekunden und besteht aus zwei Paaren von Dunkel-/Leuchtphasen. Die Dunkel- und Leuchtphasen dauern jeweils 0,5 Sekunden.

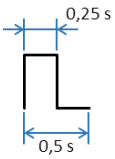
(1) unterbrochenes Licht



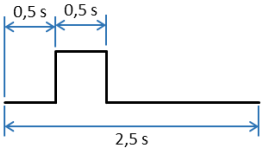
(2) langsames Gleichtaktlicht



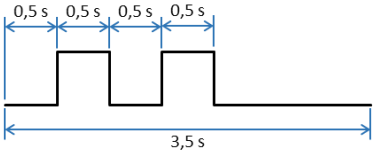
(3) schnelles Gleichtaktlicht



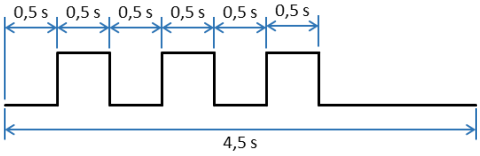
(4) 1 x Blinken



(5) 2 x Blinken



(6) 3 x Blinken



Das Abspielen eines selektierten Blinkmusters wird mit einem Timer gesteuert und muss vollständig innerhalb einer Interrupt-Service-Routine erfolgen. Der Austausch von Events zwischen der Interrupt-Service-Routine und der Funktion main() ist nicht erforderlich. Bei der Lösung dieser Aufgabe ist der Timer A0 des MSP430 zusammen mit dem Capture/Compare-Register TA0CCR0 zu benutzen. In den bereitgestellten Quelltextdaten zu dieser Aufgabe (TA0.h und TA0.c) ist der Timer A0 teilweise vorkonfiguriert. Das Feld ID (Input divider) im Steuerregister TA0CTL sowie das Feld TAIDEX (Input divider expansion) im Steuerregister TA0EX0 sind in Anlehnung an Beispiele aus der Vorlesung zu berechnen und mit geeigneten Werten zu initialisieren. Die Werte für die High- und Low-Phasen der einzelnen Blinkmuster, die bei der Abspielung ins TA0CCR0 geladen werden, sind ebenfalls in Anlehnung an die Vorlesung zu bestimmen.

Aufgrund der mechanischen Konstruktion verursachen Taster (Buttons) häufig Störimpulse beim Niederdrücken oder beim Loslassen. Deshalb ist die Abfrage solcher Taster durch GPIO-Interrupts (Auslösung durch eine steigende oder fallende Signalflanke am GPIO) äußerst problematisch und auch fehleranfällig. Eine bewährte Methode, den Zustand von Tastern zu erfassen, ist eine zeitgesteuerte Hysterese-Funktion. Man bezeichnet den Vorgang auch als „das Entprellen von Tasten". Im Rahmen dieser Aufgabe soll eine solche Funktion innerhalb der Interrupt-Service-Routine implementiert werden und die beiden Taster BTN1 und BTN2 entprellen.

Durch das erste Niederdrücken des Tasters BTN1 wird die Leuchtdiode LED1 eingeschaltet. Durch das erneue Niederdrücken des Tasters BTN1 wird die Leuchtdiode LED1 ausgeschaltet. Der Ein-/Ausschaltvorgang (sog. Toggeln) soll beliebig häufig wiederholt werden können.

Zu Demonstrationszwecken steht ein Modulo-6-Zähler in der Funktion main() zur Verfügung, mit dem man Blinkmuster selektieren kann. Der Klick auf den Taster BTN2 führt dazu, dass der Modulo-6-Zähler um Eins inkrementiert wird, wodurch das nächste Blinkmuster ausgewählt wird. Die Umschaltung eines Blinkmusters soll nicht zu einem beliebigen Zeitpunkt erfolgen, sondern nur dann, wenn die Periode eines Blinkmusters vollständig abgelaufen ist. Das Inkrementieren des Modulo-6-Zählers soll jederzeit möglich sein und unabhängig von dem Umschaltzeitpunkt eines Blinkmusters erfolgen. Wenn beispielsweise das Blinkmuster (6) „3 x Blinken" ausgeführt wird, und während dieser Zeit der Taster BTN2 fünf Mal betätigt wird, dann wird als nächstes Blinkmuster das Blinkmuster (5) „2 x Blinken" selektiert.

Die Ausführung der Funktion zur Tastenentprellung wird mit einem Timer gesteuert und muss innerhalb einer Interrupt-Service-Routine erfolgen. Bei der Lösung dieser Aufgabe ist der Timer A1 des MSP430 zusammen mit dem Capture/Compare-Register CCR0 zu benutzen. In den bereitgestellten Quelltextdaten zu dieser Aufgabe (TA1.h und TA1.c) ist der Timer A1 schon teilweise vorkonfiguriert. Das Feld ID (Input divider) im Steuerregister TA1CTL sowie das Feld TAIDEX (Input divider expansion) im Steuerregister TA1EX0 sind in Anlehnung an Beispiele aus der Vorlesung zu berechnen und mit geeigneten Werten zu initialisieren. Auch derjenige Wert, der ins TA0CCR0 geladen wird, ist ebenfalls in Anlehnung an die Vorlesung zu bestimmen.

Bei der Implementierung der Funktion zur Tastenentprellung ist darauf zu achten, dass sie mindestens zwei Zustände enthalten muss. Während der Ausführung der Interrupt-Service-Routine darf nur ein einzelner Taster mit der Funktion zur Tastenentprellung „behandelt" werden. Nachdem das Niederdrücken eines Tasters durch die Hysterese-Funktion erkannt ist, wird ein Event von der Interrupt-Service-Routine an die Funktion main() gesendet. Die Reaktion des Systems auf ein solches Event ist in den bereitgestellten Quelltextdaten bereits implementiert.

1. Hinweis: Überlegen Sie sich geeignete Datenstrukturen für die Blinkmuster und für die Entprellung von Tastern, die leicht erweiterbar und modifizierbar sind. Beachten Sie, dass Pointer-Operationen effizienter sind als die Indizierung von Tabellen.

2. Hinweis: Daten und Funktionen in den bereitgestellten Quelltextdateien dürfen zwar erweitert aber nicht gelöscht oder auskommentiert werden. Vor Modifikationen der Funktion Event_wait() ist abzusehen. Das Löschen und ggf. das Setzen des globalen Interrupt-Flags ist außerhalb der Funktion Event_wait() nicht zulässig.

3. Hinweis: Interrupt-Service-Routinen dürfen keine Schleifen enthalten, die mit for, while, goto oder anderen Anweisungen die Ausführung einer Interrupt-Service-Routine „in die Länge ziehen" und somit andere Aktivitäten im Mikrocontroller blockieren. Auch das „Ausrollen" einer Schleife, in dem man die Anweisungen aus dem Schleifenrumpf kopiert und hintereinanderschreibt, ist nicht zulässig.