MODUL 2
Einlesen eines Ports, Interrupt Routinen

Version 1.1 Dipl.-Ing. Dr. Josef Humer 1997

INHALTSVERZEICHNIS
MODUL 2
Einlesen eines Ports, Interrupt Routinen

Inhalt Seite

1 Einlesen eines Ports

Mikrocontroller stellen einige Ports zur Verfügung, über welche Daten ausgegeben und eingelesen werden können. Ports können verwendet werden, um Steuersignale auszugeben oder um Peripheriebausteine, wie z. B. eine LCD Anzeige, anzusprechen. Andererseits können über Portleitungen verschiedenste Geber und Melder, Sensoren und externe Peripheriebausteine wie zusätzliche A/D Umsetzer, Tastaturdekoder, Uhrenbausteine usw. eingelesen werden.

Als Beispiel für einen Peripheriebaustein soll ein Tastaturdekoder behandelt werden. Mit Hilfe dieses Tastaturdekoders wird eine Tastatur mit 20 Tasten, welche auf der Übungsplatine aufgebaut ist, abgefragt. Wenn eine Taste gedrückt wird, erzeugt der Dekoder eine 5 bit breite dual codierte Zahl, welche am 5bit breiten Datenbus des Dekoders ausgegeben wird. Zusätzlich dazu erzeugt der Dekoder ein Steuersignal, mit welchem er anzeigt, daß die Daten am Datenbus gültig sind. Abbildung 1 zeigt, wie die Tastatur über den Tastaturdekoder an den Mikrocontroller angeschlossen ist.

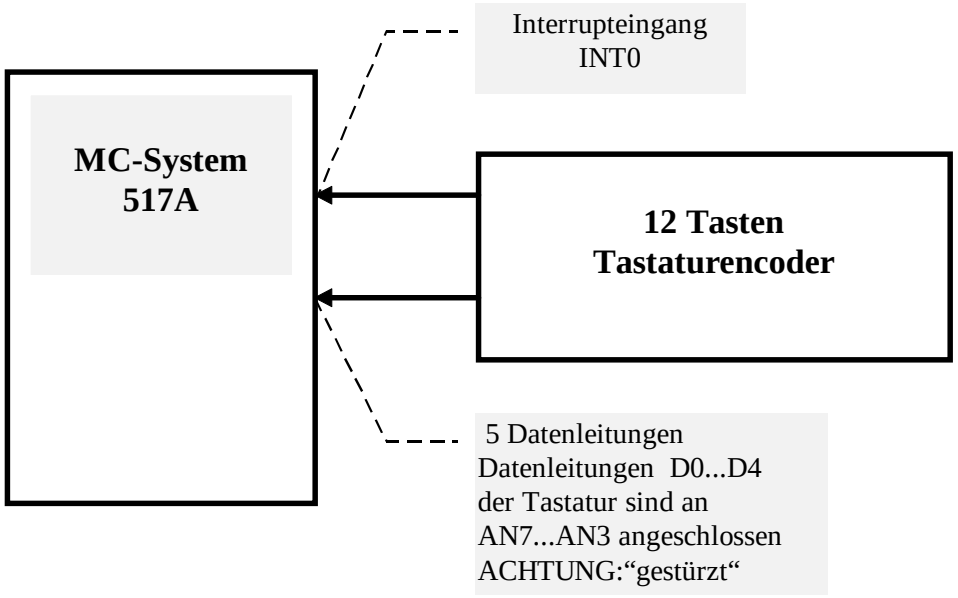


Abb. 1: Anschluß des Tastaturdekoders an den Mikrocontroller

Es gibt nun verschiedene Möglichkeiten, um den Wert des Tastaturdekoders in den Mikrocontroller einzulesen. Zwei Verfahren sollen besprochen werden:

- Das Polling Verfahren
- Die Abfrage mit Hilfe einer Interrupt Service Routine

1.1 Das Polling Verfahren

Bei diesem Verfahren fragt der Mikrocontroller den Port zyklisch ab, ob Daten anliegen. Die zyklische Abfrage kann auf verschiedene Arten realisiert werden. Die einfachste Möglichkeit ist, eine Schleife zu programmieren. Bei jedem Schleifendurchlauf wird der Port 6 abgefragt und daraus der Tastenwert berechnet.
Der Nachteil dieser Methode ist, daß der Controller durch die zyklische Abfrage jedoch ständig beschäftigt ist.

🕮 Übung 5:

Schreiben Sie ein Programm, welches die Tastatur an Port 6 einliest. Wenn eine der oberen zehn Tasten gedrückt wird, soll die entsprechende Zahl im Binärcode auf dem LED-Balken visualisiert werden.
Achtung: Der Tastaturdekoder ist mit den unteren 5 Bits des Ports 7 verbunden. Überlegen Sie, welche Möglichkeiten Sie haben, um den eingelesenen Wert zu korrigieren.

Eine Möglichkeit, den Controller zu entlasten, ist der Einbau einer Zeitschleife. Der Controller arbeitet andere Programme ab, während im Hintergrund einer der internen 16-Bit Zähler läuft. Dieser kann so programmiert werden, daß er mit 1/12 des Controllertaktes getaktet wird. Der Zähler zählt also automatisch hoch. Wenn der Zähler von 0xFFFF auf 0x0000 überläuft, löst er ein Interrupt Signal aus. Durch dieses Signal wird eine Interrupt Service Routine aufgerufen, welche auch dazu verwendet werden kann, die Tastatur einzulesen.
Mit dieser Methode ist es möglich, die Tastatur z.B. alle 0.1 Sekunden abzufragen. In der Zwischenzeit steht der Controller voll für alle anderen Aufgaben zur Verfügung. Die für die Programmierung der Interrupt Routinen nötigen Kenntnisse werden im nächsten Kapitel behandelt.

2 Die Programmierung von Interrupt Routinen

2.1 Allgemeines

In einem Mikrocontroller gibt es verschiedene Baugruppen, die einen Interrupt auslösen können. Dies sind beim 80C517 z. B. die Timer 0 und 1, der A/D Umsetzer, die seriellen Schnittstellen, usw. Außerdem besteht die Möglichkeit, über Portleitungen sieben externe Interrupts auszulösen, welche von Endschaltern, Signalgebern, usw. erzeugt werden können.
Der Mikrocontroller besitzt einige special function register in seinem internen Speicher, über welche die nötigen Einstellungen für die Bedienung der Interrupts durchgeführt werden können. Die einzelnen Bits, welche als Schalter dienen, sind mit Namen versehen und in Abb. 3a und Abb. 3b dargestellt.

2.2 Vektoradressen

Interrupt-Quelle	Request-Flags	Vektoradresse
Ext. Interrupt 0	IE0	0003H
Timer 0-Interrupt	TF0	000BH
Ext. Interrupt 1	IE1	0013H
Timer 1-Interrupt	TF1	001BH
Interrupt der ser. Schnittst. 0	RI0/TI0	0023H
Timer 2-Interrupt	TF2/EXF2	002BH
A/D-Wandler-Interrupt	IADC	0043H
Ext. Interrupt 2	IEX2	004BH
Ext. Interrupt 3	IEX3	0053H
Ext. Interrupt 4	IEX4	005BH
Ext. Interrupt 5	IEX5	0063H
Ext. Interrupt 6	IEX6	006BH
Interrupt der ser. Schnittst. 1	RI1/TI1	0083H
Compare-Int. in CM0 - CM7 (nur 80C517A)	ICMP0 - ICMP7	0093H
Compare-Timer-Interrupt	CTF	009BH
Compare-Int. in COMSET (nur 80C517A)	ICS	00A3H
Compare-Int. in COMCLR (nur 80C517A)	ICR	00ABH

Abb 2: Vektoradressen der einzelnen Interrupts

2.3 Interruptsystem 80C517

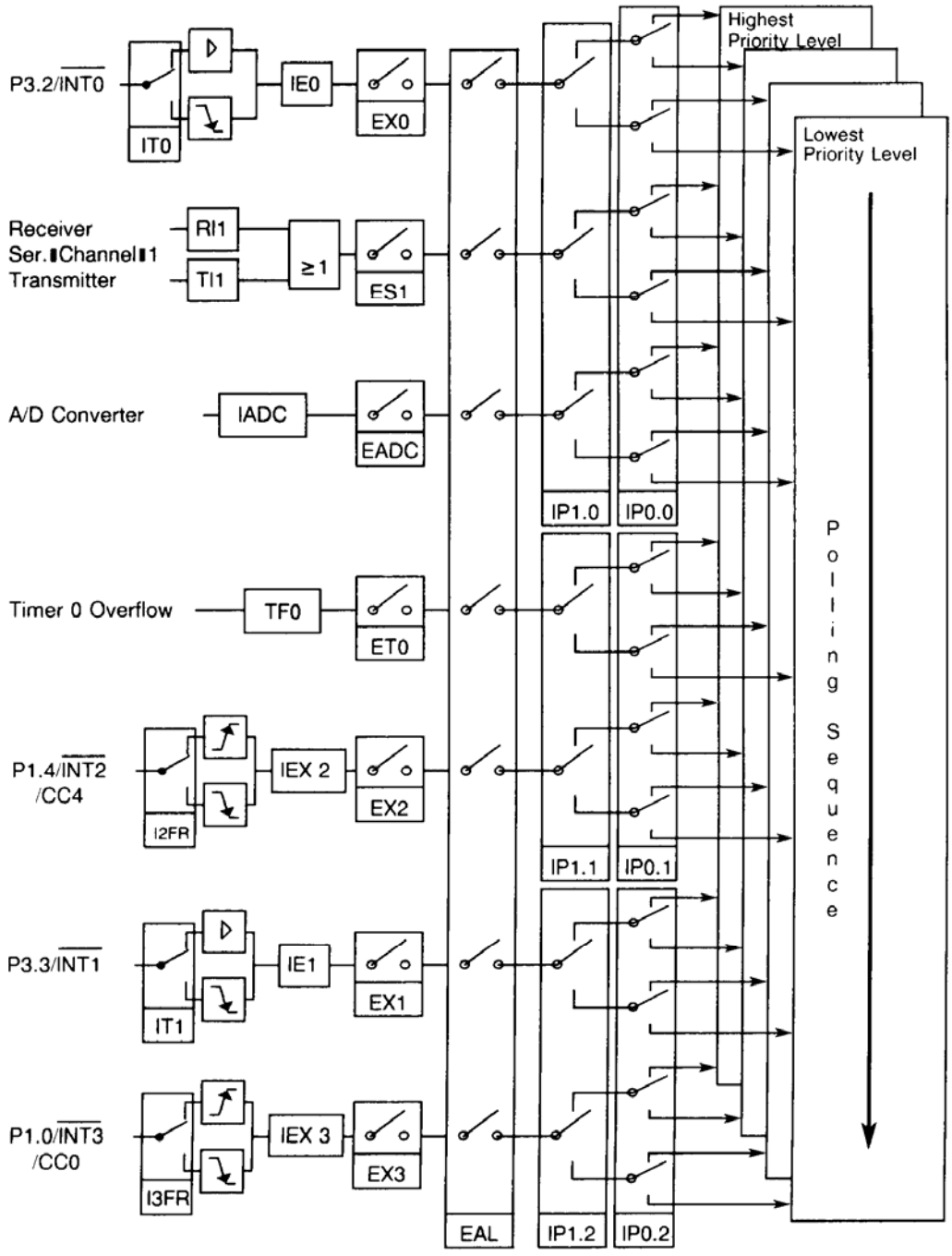


Abb. 3a: Interrupt Struktur des 80C517

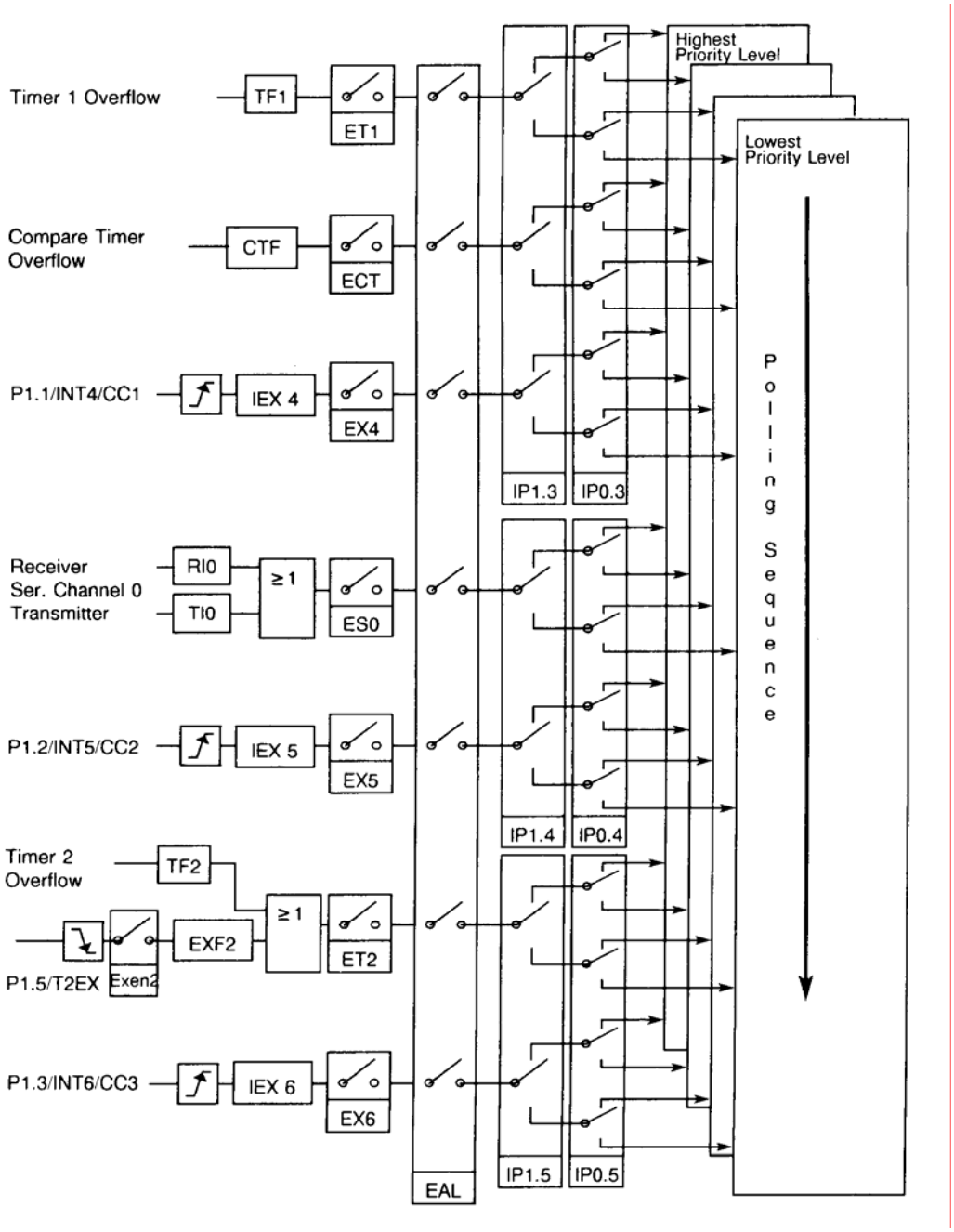


Abb. 2b: Interrupt Struktur des 80C517, Fortsetzung

Grundsätzlich sind zwei Schritte notwendig, um einen Interrupt überhaupt zu ermöglichen. Erstens muß die einzelne Interruptquelle eingeschaltet werden, anschließend müssen alle Interrupts allgemein eingeschaltet werden.

Beispiel: Der Timer 0 soll in einer Zeitschleife laufen und bei jedem Überlauf von 0xFFFF auf 0x0000 einen Interrupt auslösen.

Um den Interrupt zu erlauben, muß das Interrupt Enable Bit ET0 gesetzt werden (Schalter ET0 wird geschlossen). Außerdem muß das allgemeine Interrupt Enable Bit EAL gesetzt werden, das alle Interruptquellen gleichzeitig ein- und ausschaltet.
Die anschließenden Prioritätsregister IPX.X dienen dazu, festzulegen, welcher Interrupt wichtiger ist und zuerst bearbeitet wird, wenn mehrere Interruptsignale gleichzeitig auftreten. Sie werden im Rahmen dieser Übung auf ihrer Default Einstellung belassen.

2.4 Special Function Register für Interrupts

IEN0 (A8H), bitadressierbar

MSB							LSB
EAL	WDT	**ET2**	**ES0**	**ET1**	**EX1**	**ET0**	**EX0**
AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H

Interrupt-Freigaberegister 0 (Interrupt Enable 0). Es enthält Freigabebits mehrerer Interrupts und ein Steuerbit des Watchdog-Timers.
Reset-Wert: 00H

Bitsymbol	Funktion
EAL	Generelles Freigabebit aller Interrupts. Bei EAL = 0 wird kein angeforderter Interrupt bedient. Bei EAL = 1 gilt das individuelle Freigabebit des jeweiligen Interrupts.
ET2	Freigabebit des Timer 2-Interrupts. Es gibt den Interrupt frei (ET2 = 1) bzw. sperrt ihn (ET2 = 0).
ES0	Freigabebit des Interrupts der seriellen Schnittstelle 0. Es gibt den Interrupt frei (ES0 = 1) bzw. sperrt ihn (ES0 = 0).
ET1	Freigabebit des Timer 1-Interrupts. Es gibt den Interrupt frei (ET1 = 1) bzw. sperrt ihn (ET1 = 0).
EX1	Freigabebit des externen Interrupts 1. Es gibt den Interrupt frei (EX1 = 1) bzw. sperrt ihn (EX1 = 0).
ET0	Freigabebit des Timer-0-Interrupts. Es gibt den Interrupt frei (ET0 = 1) bzw. sperrt ihn (ET0 = 0).
EX0	Freigabebit des externen Interrupts 0. Es gibt den Interrupt frei (EX0 = 1) bzw. sperrt ihn (EX0 = 0).

Bild 17-2: Special-Function-Register IEN0 (A8H)

IEN1 (B8H), bitadressierbar

MSB							LSB
EXEN2	SWDT	**EX6**	**EX5**	**EX4**	**EX3**	**EX2**	**EADC**
BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H

Interrupt-Freigaberegister 1 (Interrupt Enable 1). Es enthält Freigabebits mehrerer Interrupts und ein Steuerbit des Watchdog-Timers.
Reset-Wert: 00H

Bitsymbol	Funktion
EXEN2	Freigabebit des extern gesteuerten Nachlade-Interrupts des Timers 2 (External Reload of Timer 2 Enable). Es gibt den Interrupt frei (EXEN2 = 1) bzw. sperrt ihn (EXEN2 = 0). Die Nachladefunktion selbst wird von EXEN2 nicht beeinflußt.
EX6	Freigabebit des externen Interrupts 6 bzw. Compare/Capture-Interrupts 3 am Pin P1.3. Es gibt den Interrupt frei (EX6 = 1) bzw. sperrt ihn (EX6 = 0). Die Compare/Capture-Funktion wird von EX6 nicht beeinflußt.
EX5	Freigabebit des externen Interrupts 5 bzw. Compare/Capture-Interrupts 2 am Pin P1.2. Es gibt den Interrupt frei (EX5 = 1) bzw. sperrt ihn (EX5 = 0). Die Compare/Capture-Funktion wird von EX5 nicht beeinflußt.
EX4	Freigabebit des externen Interrupts 4 bzw. Compare/Capture-Interrupts 1 am Pin P1.1. Es gibt den Interrupt frei (EX4 = 1) bzw. sperrt ihn (EX4 = 0). Die Compare/Capture-Funktion wird von EX4 nicht beeinflußt.
EX3	Freigabebit des externen Interrupts 3 bzw. Compare/Capture-Interrupts 0 am Pin P1.0. Es gibt den Interrupt frei (EX3 = 1) bzw. sperrt ihn (EX3 = 0). Die Compare/Capture-Funktion wird von EX3 nicht beeinflußt.
EX2	Freigabebit des externen Interrupts 2 bzw. Compare/Capture-Interrupts 4 am Pin P1.4. Es gibt den Interrupt frei (EX2 = 1) bzw. sperrt ihn (EX2 = 0). Die Compare/Capture-Funktion wird von EX2 nicht beeinflußt.
EADC	Freigabebit des A/D-Wandler-Interrupts. Es gibt den Interrupt frei (EADC = 1) bzw. sperrt ihn (EADC = 0).

Bild 17-3: Special-Function-Register IEN1 (B8H)

IEN2 (9AH), nicht bitadressierbar

MSB							LSB
-	-	**ECR/-**	**ECS/-**	**ECT**	**ECMP/-**	-	**ES1**

Interrupt-Freigaberegister 2 (Interrupt Enable Register 2). Es enthält mehrere Freigabebits für Interrupts.
Reset-Wert: xxxx 0xx0B (80C517), xx00 00x0B (80C517A)

Bitsymbol	Funktion
ECR (80C517A)	Freigabebit des Compareregisters COMCLR (Set/Reset-Modus). Es gibt den Interrupt frei (ECR = 1) bzw. sperrt ihn (ECR = 0). Im 80C517 ist dieses Bit reserviert.
ECS (80C517A)	Freigabebit des Compareregisters COMSET (Set/Reset-Modus). Es gibt den Interrupt frei (ECS = 1) bzw. sperrt ihn (ECS = 0). Im 80C517 ist dieses Bit reserviert.
ECT	Freigabebit des Compare-Timer-Interrupts. Es gibt den Interrupt frei (ECT = 1) bzw. sperrt ihn (ECT = 0).
ECMP (80C517A)	Freigabebit der Interrupts der Compareregister CM0 bis CM7 im Compare-Modus 1. Es gibt die Interrupts frei (ECMP = 1) bzw. sperrt sie (ECMP = 0). Im 80C517 ist dieses Bit reserviert.
ES1	Freigabebit des Interrupts der seriellen Schnittstelle 1. Es gibt den Interrupt frei (ES1 = 1) bzw. sperrt ihn (ES1 = 0).

Bild 17-4: Special-Function-Register IEN2 (9AH)

TCON (88H), bitadressierbar

MSB							LSB
TF1	TR1	**TF0**	TR0	**IE1**	**IT1**	**IE0**	**IT0**
8FH	8EH	8DH	8CH	8BH	8AH	89H	88H

Steuerregister für Timer 0 und 1 (Timer 0/1 Control Register). Neben Steuerbits für die Timer 0 und 1 enthält es auch Steuerbits für die Interrupts.
Reset-Wert: 00H

Bitsymbol	Funktion
TF1	Interrupt-Request-Flag des Überlaufs von Timer 1. Es wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht.
TF0	Interrupt-Request-Flag des Überlaufs von Timer 0. Es wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht.
IE1	Interrupt-Request-Flag des externen Interrupts 1 (P3.3/INT1#). Es wird gesetzt, wenn eine fallende Flanke am Pin P3.3 entdeckt wird (Voraussetzung: mit IT1 wurde fallende Flanke selektiert). Im Fall der Low-Pegel-Aktivierung bleibt IE1 gesetzt, solange der Pegel anliegt, und kann nicht durch Software gelöscht werden.
IT1	Selektionsbit für Interrupt 1 (Interrupt 1 Type Select Bit). Es legt fest, ob der Interrupt 1 durch eine fallende Flanke (IT1 = 1) oder durch einen Low-Pegel (IT1 = 0) ausgelöst wird.
IE0	Interrupt-Request-Flag des externen Interrupts 0 (P3.2/INT0#). Es wird gesetzt, wenn eine fallende Flanke am Pin P3.2 entdeckt wird (Voraussetzung: mit IT0 wurde fallende Flanke selektiert). Im Fall der Low-Pegel-Aktivierung bleibt IE0 gesetzt, solange der Pegel anliegt, und kann nicht durch Software gelöscht werden.
IT0	Selektionsbit für Interrupt 0 (Interrupt 0 Type Select Bit). Es legt fest, ob der Interrupt 0 durch eine fallende Flanke (IT0 = 1) oder durch einen Low-Pegel (IT0 = 0) ausgelöst wird.

Bild 17-5: Special-Function-Register TCON (88H)

T2CON (C8H), bitadressierbar

MSB							LSB
T2PS	**I3FR**	**I2FR**	T2R1	T2R0	T2CM	T2I1	T2I0
CFH	CEH	CDH	CCH	CBH	CAH	C9H	C8H

Steuerregister des Timers 2 (Timer 2 Control Register). Es enthält neben Bits des Timers 2 auch Bits der Interrupt-Steuerung.
Reset-Wert: 00H

Bitsymbol	Funktion
I3FR	Auswahlbit zur Einstellung der aktiven Flanke des externen Interrupts 3 (Interrupt 3 Falling/Rising Edge). Es legt fest, ob das Request Flag IEX3 gesetzt wird, wenn eine fallende (I3FR = 0) oder eine steigende (I3FR = 1) Flanke am Pin P1.0/INT3# entdeckt wird.
I2FR	Auswahlbit zur Einstellung der aktiven Flanke des externen Interrupts 2 (Interrupt 2 Falling/Rising Edge). Es legt fest, ob das Request Flag IEX2 gesetzt wird, wenn eine fallende (I2FR = 0) oder eine steigende (I2FR = 1) Flanke am Pin P1.4/INT2# entdeckt wird.

Bild 17-6: Special-Function-Register T2CON (C8H)

IRCON/IRCON0 (C0H), bitadressierbar

MSB LSB

EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC
C7H	C6H	C5H	C4H	C3H	C2H	C1H	C0H

Interrupt-Request-Register (Interrupt Request Control Register). Es enthält Flags zur Anforderung von verschiedenen Interrupts.
Reset-Wert: 00H

Bitsymbol	Funktion
EXF2	Interrupt-Request-Flag des externen Timer 2-Nachlademodus (Timer 2 External Reload Flag). Es wird bei einer fallenden Flanke am Pin 1.5/T2EX gesetzt, wenn der externe Reload-Modus für den Timer 2 gewählt ist. Ist der Nachlade-Interrupt des Timers 2 freigegeben (EXEN2 = 1), wird in die Interrupt-Routine verzweigt. EXF2 muß durch Software gelöscht werden.
TF2	Interrupt-Request-Flag des Überlaufs von Timer 2 (Timer 2 Overflow Flag). Bei einem Überlauf des Timers 2 wird es von der Hardware gesetzt. TF2 muß durch Software gelöscht werden.
IEX6	Interrupt-Request-Flag des externen Interrupts 6. Es wird gesetzt, wenn eine steigende Flanke bzw. eine Compare-Flanke am Pin P1.3/INT6/CC3 auftritt. IEX6 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.
IEX5	Interrupt-Request-Flag des externen Interrupts 5. Es wird gesetzt, wenn eine steigende Flanke bzw. eine Compare-Flanke am Pin P1.2/INT5/CC2 auftritt. IEX5 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.
IEX4	Interrupt-Request-Flag des externen Interrupts 4. Es wird gesetzt, wenn eine steigende Flanke bzw. eine Compare-Flanke am Pin P1.1/INT4/CC1 auftritt. IEX4 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.
IEX3	Interrupt-Request-Flag des externen Interrupts 3. Es wird gesetzt, wenn eine steigende/fallende Flanke (abhängig von I3FR in T2CON) bzw. eine Compare-Flanke am Pin P1.0/INT3#/CC0 auftritt. IEX3 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.

Bild 17-7 a): Special-Function-Register IRCON/IRCON0 (C0H)

Bitsymbol	Funktion
IEX2	Interrupt-Request-Flag des externen Interrupts 2. Es wird gesetzt, wenn eine steigende/fallende Flanke (abhängig von I2FR in T2CON) bzw. eine Compare-Flanke am Pin P1.4/INT2#/CC4 auftritt. IEX2 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.
IADC	Interrupt-Request-Flag des A/D-Wandlers. Es wird am Ende einer A/D-Wandlung von der Hardware gesetzt. Es muß durch Software gelöscht werden.

Bild 17-7 b): Special-Function-Register IRCON/IRCON0 (C0H), Fortsetzung

CTCON (E1H), nicht bitadressierbar

MSB							LSB
T2PS1	-	**ICR/-**	**ICS/-**	**CTF**	CLK2	CLK1	CLK0

Steuerregister des Compare-Timers (Compare Timer Control Register). Es enthält neben Steuerbits für den Compare-Timer und den Timer 2 mehrere Interrupt-Request-Flags. Reset-Wert: 0xxx 0000B (80C517), 0x00 0000B (80C517A)

Bitsymbol	Funktion
ICR (80C517A)	Interrupt-Request-Flag des Compare-Interrupts mit COMCLR. Es wird im Set/Reset-Modus bei Übereinstimmung zwischen Timer 2 und COMCLR gesetzt. ICR wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, kann aber auch durch Software gelöscht werden. Dieses Bit ist im 80C517 reserviert.
ICS (80C517A)	Interrupt-Request-Flag des Compare-Interrupts mit COMSET. Es wird im Set/Reset-Modus bei Übereinstimmung zwischen Timer 2 und COMSET gesetzt. ICS wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, kann aber auch durch Software gelöscht werden. Dieses Bit ist im 80C517 reserviert.
CTF	Interrupt-Request-Flag des Compare-Timer-Überlaufs. Es wird bei Überlauf oder Nachladen des Compare-Timers gesetzt. CTF muß durch Software gelöscht werden.

Bild 17-8: Special-Function-Register CTCON (E1H)

Der externe Interrupt 3 verhält sich völlig analog zum externen Interrupt 2. Das Request Flag ist IEX3 (in Register IRCON/IRCON0, Abbildung 17-7). Ebenso besteht die Wahlmöglichkeit, ob der Interrupt-Eingang P1.0/INT3#/CC0 auf positive oder negative Flanken reagiert; dies wird durch Bit I3FR gesteuert. Auch hier reagiert das Request Flag IEX3 auf Compare-Ereignisse im angeschlossenen Compare-Register CC0; dies geschieht bei allen Compare-Modi und unabhängig davon, ob und welcher Flankenwechsel durch diesen Compare am Pin erzeugt wird. Das Interrupt-Request-Flag IEX3 wird beim Sprung in die Interrupt-Routine automatisch zurückgesetzt.

2.5 Prioritätssteuerung

IP0 (A9H), nicht bitadressierbar
IP1 (B9H), nicht bitadressierbar

MSB							LSB
OWDS	WDTS	**IP0.5**	**IP0.4**	**IP0.3**	**IP0.2**	**IP0.1**	**IP0.0**
-	-	**IP1.5**	**IP1.4**	**IP1.3**	**IP1.2**	**IP1.1**	**IP1.0**

Interrupt-Prioritätsregister 0 und 1. Neben den Statusflags für den Oszillator-Watchdog und den Watchdog-Timer enthalten sie Bits zur Einstellung von Interrupt-Prioritäten.
Reset-Wert: IP0 = 00H, IP1 = xx00 0000B

Bitsymbol	Funktion
IP1.x IP0.x	Prioritätsprogrammierung der entsprechenden Interrupt-Gruppe (s. unten)
0 0	setzt Prioritätsstufe 0 (niedrigste Priorität)
0 1	setzt Prioritätsstufe 1
1 0	setzt Prioritätsstufe 2
1 1	setzt Prioritätsstufe 3 (höchste Priorität)

Bitpaar	Interrupt-Gruppe		
IP1.0/IP0.0	Ext. Interrupt 0	Ser. Schnittst. 1	A/D-Wandler
IP1.1/IP0.1	Timer 0	-	Ext. Interrupt 2
IP1.2/IP0.2	Ext. Interrupt 1	Compare mit CMx *)	Ext. Interrupt 3
IP1.3/IP0.3	Timer 1	Compare-Timer	Ext. Interrupt 4
IP1.4/IP0.4	Ser. Schnittst. 0	Comp. mit COMSET *)	Ext. Interrupt 5
IP1.5/IP0.5	Timer 2	Comp. mit COMCLR *)	Ext. Interrupt 6

*) nur im 80C517A

Bild 17-12: Special-Function-Register IP0 (A9H) und IP1 (B9H)

Hoch	->	Niedrig	**Priorität**
IE0	RI1/TI1	IADC	Hoch
TF0	-	IEX2	
IE1	ICMP0 - ICMP7 *)	IEX3	|
TF1	CTF	IEX4	V
RI0/TI0	ICS *)	IEX5	
TF2/EXF2	ICR *)	IEX6	Niedrig

Bild 17-13: Rangfolge innerhalb einer Prioritätsstufe

2.6 Eine Interruptgesteuerte Zeitschleife

Alle Controller der Familie 8051 enthalten mehrere Timer/Zähler Bausteine. Diese Baugruppen können in verschiedenen Betriebsarten betrieben werden. Für die Zeitschleife soll der Timer/Zähler 0 als 16-Bit Zähler verwendet werden (Betriebsart 1).

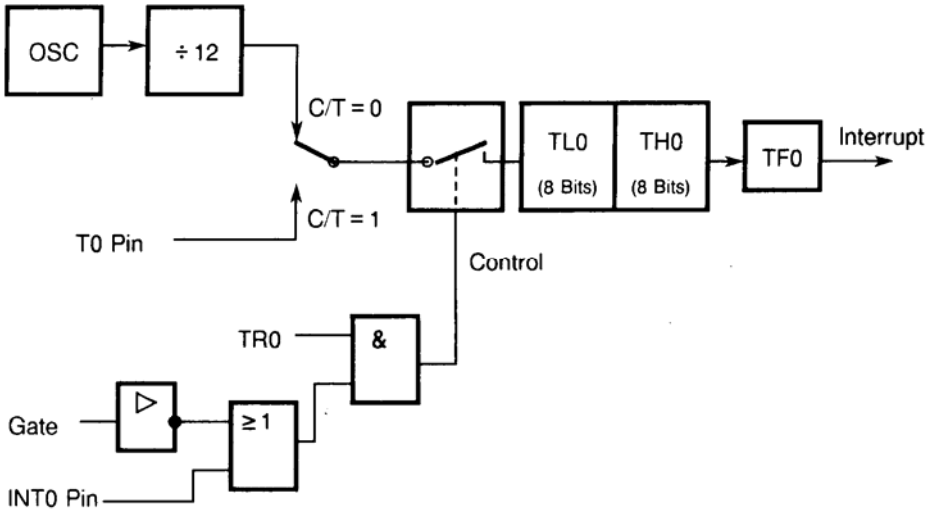


Abb. 3: Timer/Zähler 0, Betriebsart 1: 16-Bit Timer/Zähler

Die Einstellungen werden über das special function register TMOD vergenommen.

TMOD (89H), nicht bitadressierbar

MSB LSB

Timer 1				Timer 0			
Gate	**C/T#**	**M1**	**M0**	**Gate**	**C/T#**	**M1**	**M0**

Modusregister für Timer 0 und 1 (Timer Mode). Es enthält die Bits zur Einstellung der Betriebsarten für die Timer 0 und 1. In den Erläuterungen steht x für 0 oder 1, je nachdem, welcher Timer verwendet wird.
Reset-Wert: 00H

Bitsymbol		Funktion
Gate		Wenn dieses Bit gesetzt ist, läuft der Timer nur dann, wenn er mit TRx freigegeben ist und gleichzeitig der Portpin P3.2/INT0# (Timer 0) bzw. P3.3/INT1# (Timer 1) auf High-Pegel ist.
C/T#		Dieses Bit legt fest, ob Timer- oder Counter-Modus verwendet wird. C/T# = 0 wählt Timer- und C/T# = 1 wählt Counter-Modus.
M0	M1	Arbeitsmodi
0	0	THx dient als 8-bit-Timer; TLx bildet einen 5-bit-Vorteiler.
0	1	THx und TLx bilden einen 16-bit-Timer.
1	0	Auto-Reload-Timer (8 bit). Der Inhalt von THx wird beim Timerüberlauf nach TLx kopiert. THx selbst bleibt unverändert.
1	1	Timer 0: Beide Register THx und TLx arbeiten als eigenständige 8-bit-Timer. TLx wird durch die Steuerbits von Timer 0, THx durch die Steuerbits von Timer 1 eingestellt. Timer 1: Timer 1 stoppt in dieser Betriebsart.

Abb. 4: SFR TMOD

Die richtige Einstellung für die benötigte Betriebsart lautet: **TMOD = 0x01;**
Um eine Zeitschleife mit einer definierten Zeit aufzubauen, geht man folgendermaßen vor:
Der Timer wird mit 1/12 der Quarzfrequenz getaktet. In unserem Fall ist dies 1 MHz. Das Zählregister TL0, welches 8 Bit breit ist, muß von 0 bis 255 zählen, um einmal überzulaufen. Erst dann wird der Inhalt des Zählregister TH0 um Eins erhöht.
D. h., das Register TL0 teilt die Frequenz durch 256. Es ergibt sich eine Zählfrequenz von 3906.25 Hertz. Soll die Tastatur alle 1/100 Sekunden eingelesen werden, müssen die 3906.25 Hertz nochmals durch 39.0625, also ungefähr 39 geteilt werden. Dies erreicht man, wenn das Register TH0 mit dem Wert -39 (= 217) vorgesetzt wird. Durch diese Einstellung wird alle 10 msec ein Interrupt ausgelöst.

Ein vollständiges Interruptprogramm:

```
void INTTIM0 (void) interrupt 1
  {
    taste = P7&0x1F;
    TH0 = -39;
  }
```

2.7 Interruptquellen und -nummern

Interrupt-Quellen	Intr. Nr.	auslösendes Moment	Request -Flag	Rück-setzen	Einsprung-adressen
Externer Interrupt 0	0	neg. Flanke/Pegel	IE0	H	3H
Timer-0-Interrupt	1	Timer-0-Überlauf	TF0	H	BH
Externer Interrupt 1	2	neg. Flanke/Pegel	IE1	H	13H
Timer-1-Interrupt	3	Timer-1-Überlauf	TF1	H	1BH
Interrupt der seri-ellen Schnittstelle	4	Ende der Eingabe und Ausgabe	RI + TI	S	23H
Timer-2-Interrupt	5	Timer-2-Überlauf externer Reload	TF2 + EXF2	S	2BH
A/D-Wandler	8	Ende der Wandlung	IADC	S	43H
Externer Interrupt 2	9	neg./pos. Flanke	IEX2	H	4BH
Externer Interrupt 3/ Capture-0-Eingang/ Compare-0-Ausgang	10	neg./pos. Flanke neg./pos. Flanke neg./pos. Flanke	IEX3	H	53H
Externer Interrupt 4/ Capture-1-Eingang/ Compare-1-Ausgang	11	pos. Flanke pos. Flanke neg./pos. Flanke	IEX4	H	5BH
Externer Interrupt 5/ Capture-2-Eingang/ Compare-2-Ausgang	12	pos. Flanke pos. Flanke neg./pos. Flanke	IEX5	H	63H
Externer Interrupt 6/ Capture-3-Eingang/ Compare-3-Ausgang	13	pos. Flanke pos. Flanke neg./pos. Flanke	IEX6	H	6BH
Rücksetzen durch: H = Hardware; S = Software					

Bild 9-1: Interrupt-Quellen und Einsprungadressen des 8051 - 80515

2.8 Änderung in der startup.a51 - Datei

```
;------------------------------------------------------------------------------
;  This file is part of the C-51 Compiler package
;  Copyright KEIL ELEKTRONIK GmbH 1990
;------------------------------------------------------------------------------
;  STARTUP.A51:  This code is executed after processor reset.
;
;
;
;------------------------------------------------------------------------------
;
;  User-defined Power-On Initialization of Memory
;
;  With the following EQU statements the initialization of memory
;  at processor reset can be defined:
;
;               ; the absolute start-address of IDATA memory is always 0
IDATALEN        EQU     80H     ; the length of IDATA memory in bytes.
;
XDATASTART      EQU     0H      ; the absolute start-address of XDATA memory
XDATALEN        EQU     0H      ; the length of XDATA memory in bytes.
;
PDATASTART      EQU     0H      ; the absolute start-address of PDATA memory
PDATALEN        EQU     0H      ; the length of PDATA memory in bytes.
;
;  Notes:  The IDATA space overlaps physically the DATA and BIT areas of the
;          8051 CPU. At minimum the memory space occupied from the C-51
;          run-time routines must be set to zero.
;------------------------------------------------------------------------------
;
;  Reentrant Stack Initilization
;
;  The following EQU statements define the stack pointer for reentrant
;  functions and initialized it:
;
;  Stack Space for reentrant functions in the SMALL model.
IBPSTACK        EQU     0       ; set to 1 if small reentrant is used.
IBPSTACKTOP     EQU     0FFH+1  ; set top of stack to highest location+1.
;
;  Stack Space for reentrant functions in the LARGE model.
XBPSTACK        EQU     0       ; set to 1 if large reentrant is used.
XBPSTACKTOP     EQU     0FFFFH+1; set top of stack to highest location+1.
;
;  Stack Space for reentrant functions in the COMPACT model.
PBPSTACK        EQU     0       ; set to 1 if compact reentrant is used.
PBPSTACKTOP     EQU     0FFFFH+1; set top of stack to highest location+1.
;
;------------------------------------------------------------------------------
;
;  Page Definition for Using the Compact Model with 64 KByte xdata RAM
;
;  The following EQU statements define the xdata page used for pdata
;  variables. The EQU PPAGE must conform with the PPAGE control used
;  in the linker invocation.
;
PPAGEENABLE     EQU     0       ; set to 1 if pdata object are used.
PPAGE           EQU     0       ; define PPAGE number.
;
;------------------------------------------------------------------------------

      NAME    ?C_STARTUP


?C_C51STARTUP   SEGMENT   CODE
?STACK          SEGMENT   IDATA

      RSEG    ?STACK
      DS      1

      EXTRN CODE (?C_START)
      EXTRN CODE (INTTIM0)
```

```
      EXTRN CODE (EXTINT1)
      PUBLIC  ?C_STARTUP
      CSEG    AT      800BH
      LJMP    INTTIM0
      CSEG    AT      8013H
      LJMP    EXTINT1
      CSEG    AT      2000H
?C_STARTUP:     LJMP    STARTUP1

      RSEG    ?C_C51STARTUP
STARTUP1:

IF IDATALEN <> 0
      MOV     R0,#IDATALEN - 1
      CLR     A
IDATALOOP:      MOV     @R0,A
      DJNZ    R0,IDATALOOP
ENDIF

IF XDATALEN <> 0
      MOV     DPTR,#XDATASTART
      MOV     R7,#LOW (XDATALEN)
  IF (LOW (XDATALEN)) <> 0
      MOV     R6,#(HIGH XDATALEN) +1
  ELSE
      MOV     R6,#HIGH (XDATALEN)
  ENDIF
      CLR     A
XDATALOOP:      MOVX    @DPTR,A
      INC     DPTR
      DJNZ    R7,XDATALOOP
      DJNZ    R6,XDATALOOP
ENDIF

IF PDATALEN <> 0
      MOV     R0,#PDATASTART
      MOV     R7,LOW (PDATALEN)
      CLR     A
PDATALOOP:      MOVX    @R0,A
      INC     R0
      DJNZ    R7,PDATALOOP
ENDIF

IF IBPSTACK <> 0
EXTRN DATA (?C_IBP)

      MOV     ?C_IBP,#LOW IBPSTACKTOP
ENDIF

IF XBPSTACK <> 0
EXTRN DATA (?C_XBP)

      MOV     ?C_XBP,#HIGH XBPSTACKTOP
      MOV     ?C_XBP+1,#LOW XBPSTACKTOP
ENDIF

IF PBPSTACK <> 0
EXTRN DATA (?C_PBP)
      MOV     ?C_PBP,#LOW PBPSTACKTOP
ENDIF

IF PPAGEENABLE <> 0
      MOV     P2,#PPAGE
ENDIF
      MOV     SP,#?STACK-1
      LJMP    ?C_START
      END
```

Die Änderung der startup.a51 - Datei kann unterlassen werden, wenn im Programm ein Interruptvektor angegeben wird! Ab VERSION 4.0!!

#pragma INTVECTOR(0x8000)

🕮 Übung 6:

Schreiben Sie eine interruptgesteuerte Zeitschleife, welche alle 10 msec den Tastaturdekoder an Port 7 einließt. Geben Sie die gedrückte Taste wie gehabt auf dem LED-Balken aus.

2.9 Bearbeitung eines externen Interrupts

Wie in Abb. 1 auf Seite 4 zu sehen ist, ist der Tastaturdekoder auch mit einer externen Interruptleitung verbunden. Eine negative Flanke an diesem Pin löst den externen Interrupt 1 aus.
Diese Steuerleitung kann ebenfalls benützt werden, um ein Programm zu erstellen, welches mittels einer Interrupt Routine die Tastatur einließt. Die dazu notwendigen Einstellungen sind der Interrupt Struktur in Abb. 2a und Abb. 2b zu entnehmen.

🕮 Übung 7:

Lesen Sie die Tastatur mit Hilfe des externen Interrupts 0 ein und geben Sie die Tastennummer auf PORT 4 aus.

2.10 Musterlösung zu Übung 5

```
/*********************************************************/
/*                      UEBUNG5.C                        */
/*********************************************************/

/****   Steueranweisungen an den Compiler       ****/


/****   Angabe der Include Dateien              ****/
#include <reg517a.h>


code char tasten[23]={0,0,2,0,1,0,3,0,8,0,10,0,9,0,11,0,4,0,6,0,5,0,7};


/* Hauptprogramm */
main()
{
        while(1)
        {
        P4=tasten[P7>>3];
        }
}
```

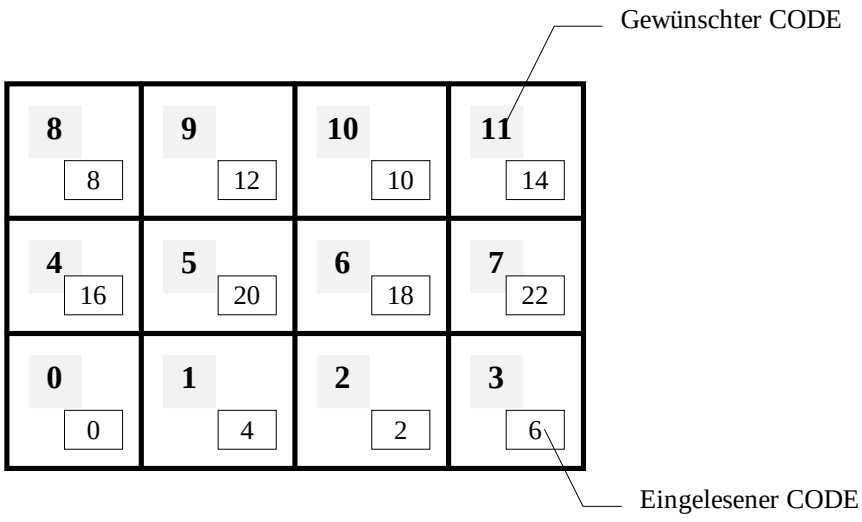
Anmerkungen:

P7>>3 bedeutet, daß alle 8 Datenbits um 3 Stellen nach rechts verschoben werden.

Tasten[23] ist ein ARRAY vom Typ char und dient der Konvertierung der Tasten.

Code bedeutet, daß Nachfolgendes ARRAY nicht im internen RAM, sondern im Programmspeicher abgelegt wird.

Tastenbelegung für obiges Beispiel:



2.11 Musterlösung zu Übung 6

```
/* ******************************************************  */
/*                      UEBUNG6.C                          */
/*      Einlesen der Tastatur ueber einen Timer Interrupt  */
/* ******************************************************  */

/****   Steueranweisungen an den Compiler       ****/
#pragma iv(0x8000)

/****   Angabe der Include Dateien              ****/
#include <reg517.h>

/****   Globale Variablen                       ****/
code char tasten[23]={0,0,2,0,1,0,3,0,8,0,10,0,9,0,11,0,4,0,6,0,5,0,7};
unsigned int taste;

/* Interruptprogramm */
/* wird alle 10.000 Takte ausgelöst, bei 12MHz alle 10ms */

void INTTIM0 (void) interrupt 1
{
   taste = tasten[P7>>3];
   TH0 = -39;
   TL0 = 0;
}

/* Hauptprogramm */
main()
{

/*      Anweisungen zur Behandlung des Interrupts       */

TMOD = 0x01;           /**** Timer 0 Mode 1 16 Bit             ****/
TH0  = -39;            /**** Vorgesetzter Wert, alle 10 ms Int.****/
TL0  = 0;              /**** Vorgesetzter Wert                 ****/
ET0 = 1;              /**** Timer 0 Interrupt disabled        ****/
EAL = 1;              /**** Interrupt allgemein erlaubt       ****/
P4 = 0x00;
TR0 = 1;               /**** Einschalten der Zeitschleife      ****/

while (1)
   {
   P4=taste;
   }

}
```

2.12 Musterlösung zu Übung 7

```
/*******************************************************/
/*                      UEBUNG7.C                      */
/*      Einlesen der Tastatur ueber ext Interrupt 0    */
/*******************************************************/

/****   Steueranweisungen an den Compiler       ****/
#pragma iv(0x8000)

/****   Angabe der Include Dateien              ****/
#include <reg517.h>

/****   Globale Variablen                       ****/
code char tasten[23]={0,0,2,0,1,0,3,0,8,0,10,0,9,0,11,0,4,0,6,0,5,0,7};
unsigned int taste;
bit gedrueckt;

void TAST (void) interrupt 0
{
   taste = tasten[P7>>3];
   gedrueckt=1;
}


/* HAUPTPROGRAMM */

main()
{

/*      Anweisungen zur Behandlung des Interrupts    */

EX0 = 1;        /**** Externer Int0               ****/
EAL = 1;        /**** Interrupt allgemein erlaubt ****/
P4 = 0x00;

while (1)
   {
   if(gedrueckt)
      {
      P4=taste;
      gedrueckt=0;
      }
   }
}
```

Erweiterung der Übung 7 durch die Ausgabe an die serielle Schnittstelle mit printf!

Die Initialisierung der seriellen Schnittstelle kann hier unterlassen werden, da der Monitor bereits dies Aufgabe durchgeführt hat.

```
/*******************************************************/
/*                      UEBUNG7a.C                     */
/*      Einlesen der Tastatur ueber einen Interrupt    */
/*******************************************************/

/****   Steueranweisungen an den Compiler       ****/
#pragma iv(0x8000)

/****   Angabe der Include Dateien              ****/
#include <reg517.h>
#include <stdio.h>

/****   Globale Variablen                       ****/
code char tasten[23]={0,0,2,0,1,0,3,0,8,0,10,0,9,0,11,0,4,0,6,0,5,0,7};
char taste;
bit gedrueckt;

void TAST (void) interrupt 0
{
   taste = tasten[P7>>3];
   gedrueckt=1;
}

main()
{

/*      Anweisungen zur Behandlung des Interrupts    */

EX0 = 1;        /**** Externer Int0               ****/
EAL = 1;        /**** Interrupt allgemein erlaubt ****/
IT0 = 1;        /* neg. Flanke */
P4 = 0x00;

while (1)
   {
   if(gedrueckt)
     {
     P4=taste;
     printf("Taste %2bd gedrückt\n",taste);
     gedrueckt=0;
     } /* endif */
   } /* endwhile */
}
```

Versuch:

Steuern Sie den externen Interrupt0 nicht mit der Flanke, sondern mit dem Pegel!

Untersuchen Sie den Unterschied!