Lösung.
Zu 1.

```
      jmp init

org 03
      nop
      nop
;     setb tr0    ;Timer start
      reti        ;zurück ins Hauptprg
org 0bh
      nop
      nop
      reti

org 13h
      nop
      nop
;     clr  tr0    ;Timer stop
      reti        ;zurück ins Hauptprg
org 1bh
      nop
      nop
      reti
init:
      setb ex0    ;Freigabe ext. Int
      setb ex1
      setb et0    ;Freigabe Timer Int
      setb et1
      ;setb it0   ;Flankensteuerung
      ;setb it1
      setb ea           ;globale I-Freigabe

loop:

      jmp loop
```

zu2. Zunächst wird die Routine für den Ext-I0 ausgeführt, dann ins HP zurückgesprungen bevor der Ext-I1-Routine ausgeführt wird

zu 3+4. Das Req-Bit wird stets gesetzt sobald der Eingang P3.2/P3.3 auf 0 gesetzt wird unabhängig von der Freigabe.

Zu5+9. Hier gilt das in 2. bereits gesagte. Man sollte bei der Reihenfolge der Initialisierung aufpassen und EA grundsätzlich erst am Ende setzten. Im folgenden Beispiel würde, falls P3.3 beim Start des Programms 0 ist, ein Pegelgesteuerter Ext-Int1 ausgeführt, bevor die Flankensteuerung aktiviert wird.

```
      setb ex1    ;Freigabe ext. Int
      nop
      nop
      nop
      setb ex0
      setb et0    ;Freigabe Timer Int
      setb et1
      setb it0    ;Flankensteuerung
      setb it1
```

zu 6. Die Reihenfolge der Ausführung entspricht der bei den I-Vektoren: Ext0-T0-Ext1-T1.

Zu 7. Die Ext1-I-Routine wird vollständig ausgeührt, zurück zum HP gesprungen, wo mindestend ein Befehl ausgeführt wird. Dies ist die Zeit, die benötigt wird, um den PC zu beschreiben. Danach erst wird in die Ext0-I-Routine gesprungen.

Zu 8. Die Routine für den Ext0-I beginnt an der Adresse 03h und die für den Timer0 an der Adresse 0Bh. Das entspricht einer Differenz von 8. Die Ext0-I-Routine darf also nicht länger als 8 Byte sein. Bei längeren Routinen verwendet man einen jmp-Befehl und führt das Programm so an anderer Stelle fort.

```
Org 03
      Jmp Ext0_R
.
.
Ext0_R:
      nop
      nop
      reti
```