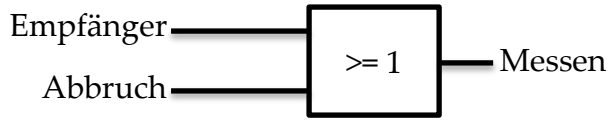
Lösungen zu 16-17 T1 A2

2.1.1 In einem 40kHz Takt dauert ein Takt

$$\frac{1}{40k}s = \frac{1}{40\,000}s = \frac{1\,000\,000}{40\,000}\mu s = \frac{100}{4}\mu s = 25\mu s$$

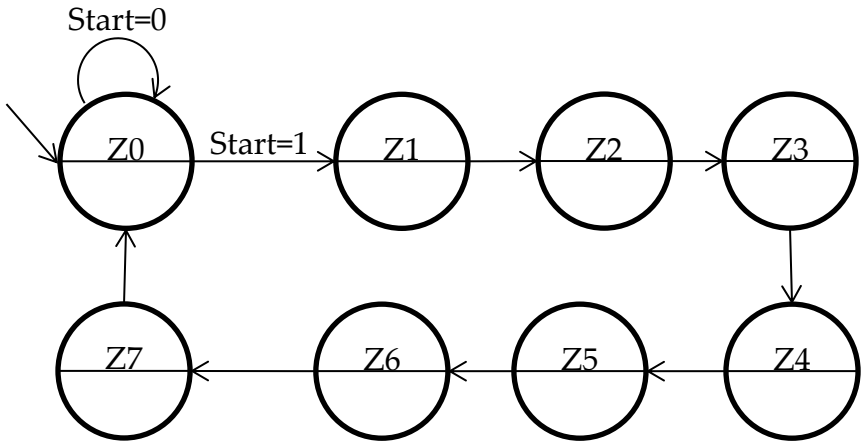
2.1.2 Ein Oder: Abbruch oder Empfänger:



2.1.3
Mit Schaltnetz nach dem Schaltwerk ist es eine **minimale Codierung**.
Ohne Schaltnetz ist es eine **optimale Codierung.**

Vorteile:
minimal: Wenig FF, einfacherer Entwurf vom Schaltnetz
optimal: kein SN nötig, das SW liefert die Codes passend für die Ausgabe

2.1.4
Im Impulsdiagramm sieht man, dass es 8 Takte vom Anfang bis zum Zyklusende sind ➔ 8 Zustände zu Beginn warten wir auf das Start Signal



2.1.5
Für das Schaltwerk werden bei minimaler Codierung 3 FF benötigt, da es 8 Zustände sind und 2^3=8 ist.

2.1.6 **Optimale** Zustandscodierung heißt, dass wir die Ausgangssignale direkt in dem Code (FF) haben müssen. Die Ausgänge von SW1 sind *Senden (B)* und *Ende (E)* und haben folgende Werte:

Name	**B**	**E**
Z0	0	0
Z1	1	0
Z2	0	0
Z3	1	0
Z4	0	0
Z5	1	0
Z6	0	0
Z7	0	1

Da der Code 00 viermal und 10 dreimal vorkommt brauchen wir noch 2 FF um die 4 gleichen Zustände zu unterscheiden. (Q_3 und Q_2 können andere Werte haben, wichtig ist nur, dass kein Code doppelt vorkommt)

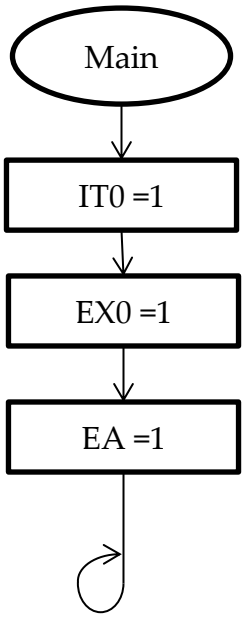
	optimaler Code			
Name	**Q_3**	**Q_2**	**B**	**E**
Z0	0	0	0	0
Z1	0	0	1	0
Z2	0	1	0	0
Z3	0	1	1	0
Z4	1	0	0	0
Z5	1	0	1	0
Z6	1	1	0	0
Z7	0	0	0	1

2.1.7 Da die Codes schon in 2.1.6 da sind kann man auch eine nicht codierte Zustandsübergangstabelle machen. Man muss aber das Signal *Start* beachten.

Start	**Z_n**	**Z_{n+1}**
0	Z0	Z0
1	Z0	Z1
X	Z1	Z2
X	Z2	Z3
X	Z3	Z4
X	Z4	Z5
X	Z5	Z6
X	Z6	Z7
X	Z7	Z0

2.2.1 Als Code und PAP

```
main:
     setb IT0
     setb EX0
     setb EA
     jmp $
```



2.2.2 Vorteile von Interrupts:

- Die Erkennung von Interrupts läuft nebenher, daher kann im Hauptprogramm anderer Code ausgeführt werden
- Interrupts vereinfachen den Code
- Interrupts prüfen jeden MZ ob ein Ereignis vorliegt

2.2.3 Da in 59µs Schritten gezählt wird, eignet sich ein 8-Bit Timer.

```
timerinit:  mov TMod, #0010b  ; Timer 0 als 8-Bit-Auto-Reload-Timer
                              ; (mov TMod, #2 geht auch, da 10b = 2)
            mov TL0, #197     ; Startwert für den ersten Durchlauf 2^8-59
                              ; so dass er nach 59µs einen Überlauf macht
            mov TH0, #197     ; … und auch als Reload-Wert
            setb ET0          ; Enable Timer 0 Interrupt
            setb EA           ; Enable All
            setb TR0          ; Timer starten
            ret               ; weil es ein Unterprogramm ist
```

2.2.4

```
org 03h
            clr EX0         ; Externen Interrupt deaktivieren
            mov distanz, #0
            call timerinit
            reti
```

2.2.5

```
org 0Bh
            inc distanz
            jnb P3.2, weiterMessen
            clr TR0             ; Timer stoppen
            mov P1, distanz     ; Distanz ausgeben
            setb EX0            ; Externer Interrupt 0 wieder an
weiterMessen:
            reti
```