Timing-Programmierung

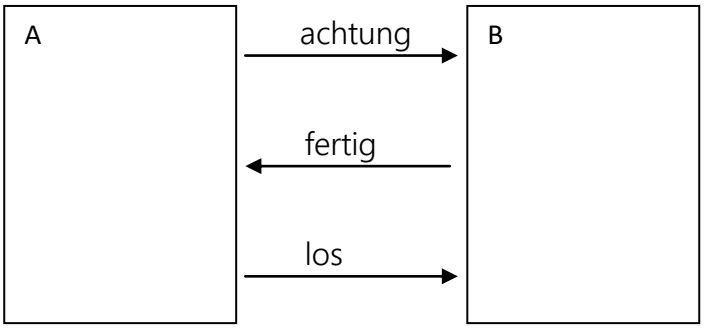
Bei der Programmierung einer Kommunikation zwischen Geräten ist fast immer ein korrekter zeitlicher Ablauf einzuhalten.

Das kann man durch "aktives Warten" (polling) auf Ereignisse machen, oder durch Programmieren eines Zustandsautomaten (state machine).

Am Beispiel erklärt :

Gerät A sagt : "achtung"
Berät B antwortet : "fertig"
Gerät A startet dann Gerät B mit "los"

Ein Blockbild zeigt nochmal deutlich die Signalrichtungen :



Ablauf in Gerät A :

gib achtung aus
warte, bis fertig eintrifft
gib los aus

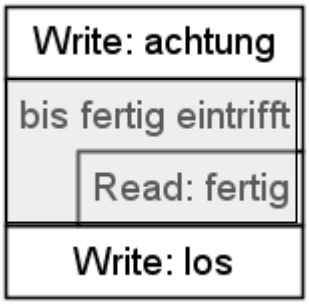
Ablauf in Gerät B :

warte, bis achtung eintrifft
gib fertig aus
warte, bis los eintrifft
(starte mechanik etc.)

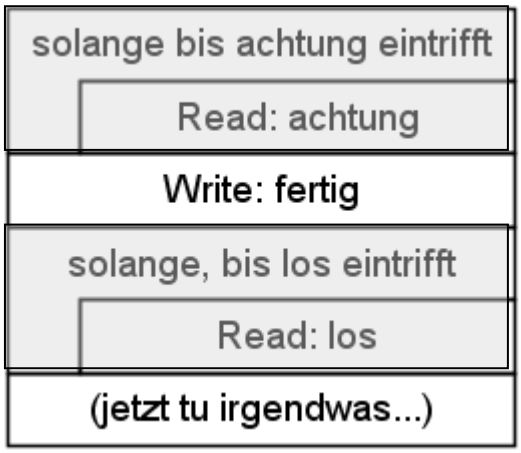
Und nun zwei grundsätzliche Möglichkeiten, das zu realisieren.

Zuerst mit Warteschleifen im Programm (polling) :

Ablauf in Gerät A :



Ablauf in Gerät B :

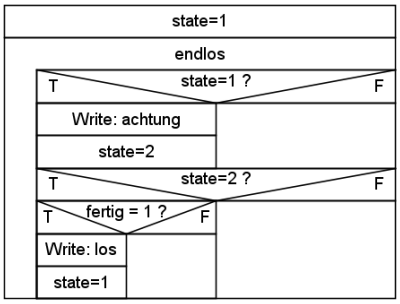


(Das ist nicht ganz korrekt, weil jeweils beim ersten Schleifenlauf noch kein Wert für den Eingang vorhanden ist, den muß man drüber einmal vorher lesen. Das habe ich aber der Übersichtlichkeit wegen weggelassen)

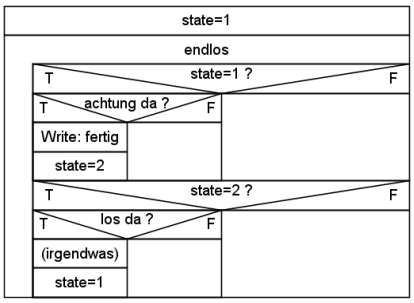
In den markierten Schleifen bleibt das Programm “hängen” und wartet.

Andere Möglichkeit : Programmzyklus und state machine

Ablauf in Gerät A :



Ablauf in Gerät B :



Ob bei dieser minimalen Aufgabe dieser Aufwand nötig ist, soll hier nicht diskutiert werden, das geht viel einfacher, es sollen hier nur die Konzepte dargestellt werden :

Beim Polling wird mitten im Programmlauf angehalten und durch dauerndes Fragen gewartet.

Bei der state machine (Zustandsautomat) bestimmen die Zustände den zeitlichen Ablauf. Das Programm hat keine Warteschleifen, sondern läuft in einer Dauerschleife. Die Zustände werden meist durch Eingänge weitergeschalten.