Timings mit einem uC erzeugen

{TEXTBOX:video}

{TEXTBOX:cheatsheet}

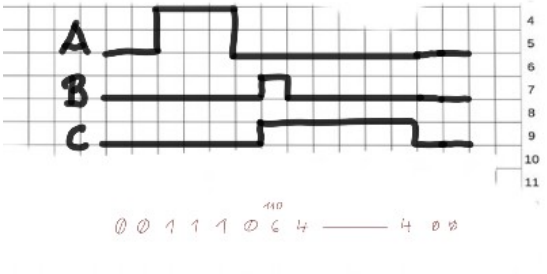
Timings können nach verschiedenen Methoden erzeugt werden, jed nachem, wie genau die Zeiten sein müssen, ob das Hauptprogramm blockiert werden darf und welche Ressourcen zur Verfügung stehen:

- Schleife
  das Timing wird erzeugt, indem eine Warteschleife programmiert wird; diese Zeiten können blockierend oder nicht blockierend realisiert werden. ACHTUNG! **Die Timings ändern sich stark wenn die Optimierung des Compilers aktiviert ist.**
- Timeroverflow Interrupt
  Ein Timer wird so eingestellt, dass seine Überläufe den Takt vorgeben; man kann den Timer auf einen bestimmten Wert setzen (preload), dann läuft er schneller über. Nach einem Overflow wird in der Interrupt Service Routine (ISR) mitgezählt wie oft der Überlauf passiert ist und immer wieder der richtige Preload-Wert gesetzt.
- CTC
  Der Timer läuft im CTC Mode bis zu einem Vergleichswert und wird dann auf 0 rückgesetzt. Man kann durch geeignete Wahl der Zählgeschwindigkeit und des Vergleichswertes das Timing einstellen. In der ISR kann mitgezählt werden, wie oft der Überlauf passiert ist.

Aufgabe

Folgendes Timing soll mit allen Methoden implementiert werden. Die Zeiteinteilung ist z.B: in ms .

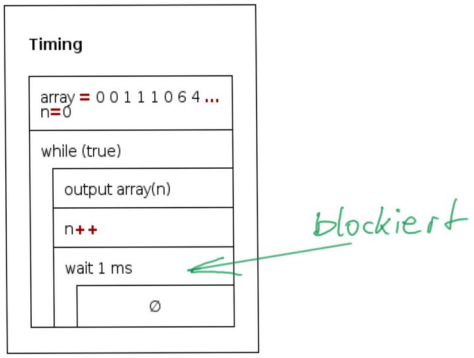
Für die Beispiele wird ein AVR 328p mit f_{OSC}=16MHz (Periode 62,5ns) verwendet.



Man kann dieses Timing als Zählfolge sehen: der Zählerstand ändert sich alle 1ms.

Die Zählfolge ist 0 0 1 1 1 0 6 4 4 4 4 4 0 0 und wiederholt sich dann, die Periode dauert also 14ms. Man muss hier auf die kleinste zu realisierende Zeiteinheit schauen um herauszufinden, welche Zeitauflösung benötigt wird.

Lösung 1

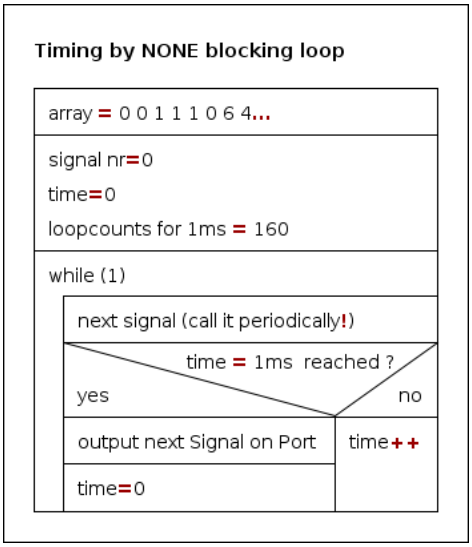


problematisch ist bei der Verwendung von delay_ms, dass die Anwendung wartet, bis die angegebene Zeit vergangen ist.

{TEXTBOX:cheatsheet}

{TEXTBOX:gist}

Lösung 2



Hier blockiert die Schleife nicht, dafür ist die Struktur komplexer. Eine Variable time zählt die Schleifendurchläufe und liefert so die Zeitbasis.

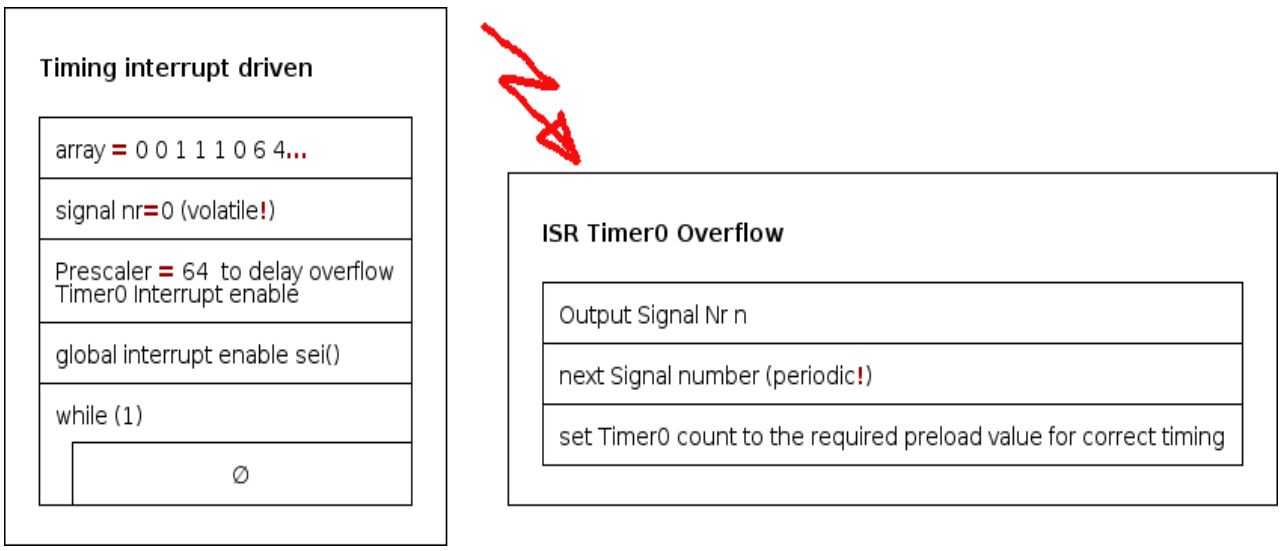
Problematisch: jede Änderung des Codes und die Compiler-Einstellungen ändern das Timing.

{TEXTBOX:cheatsheet}{TEXTBOX:gist2}

Lösung 3

Interruptsteuerung entlastet das Hauptprogramm . Mittels Simulator wird das Timing abgeglichen.

Hier werden Timer Overflow und Compare Match Interrupt verwendet.



{TEXTBOX:cheatsheet}{TEXTBOX:gist3}