

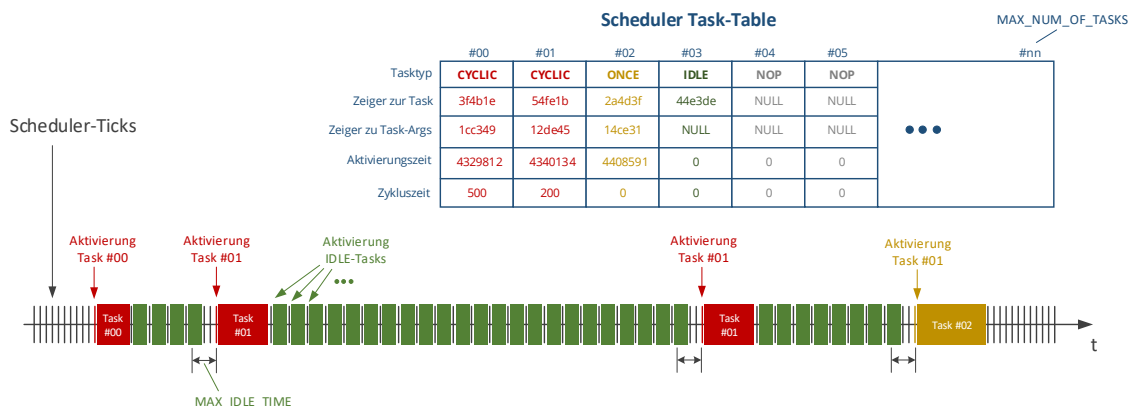
# Übung 2-2 – Echtzeitbetriebssysteme

## Nichtpreemptiver Scheduler

(a.)

Schreiben Sie für den Aurix TC375 Lite Kit einen einfachen **nichtpreemptiven Scheduler** (**Scheduler.h**, **Scheduler.cpp** – siehe Template-Dateien mit den folgenden Anforderungen:

- Der Scheduler besitzt einen Puffer mit einer maximalen Anzahl handelbarer Tasks (z.B. 10)
- Jede Task besitzt einen **Task Control Block**, bestehend aus (vgl. Bild)
  - Zeiger zur Taskfunktion
  - (Optional: Zeiger auf mögliche Argumente der Taskfunktion)
  - Zykluszeit
  - Aktivierungszeitpunkt
  - Tasktyp (**NOP**, **IDLE**, **ONCE**, **CYCLIC**)



- Der Scheduler ruft in Abhängigkeit des Typs und der Zyklus- und Aktivierungszeit die aktiven Tasks im Puffer entsprechend auf
- Verwenden Sie als Zeitbasis die Funktion **millis()**. Sie liefert einfach einen aktuellen Zähler zurück (Zeit in ms seit Reset), der durch einen Interrupt aller 1024  $\mu$ s inkrementiert wird

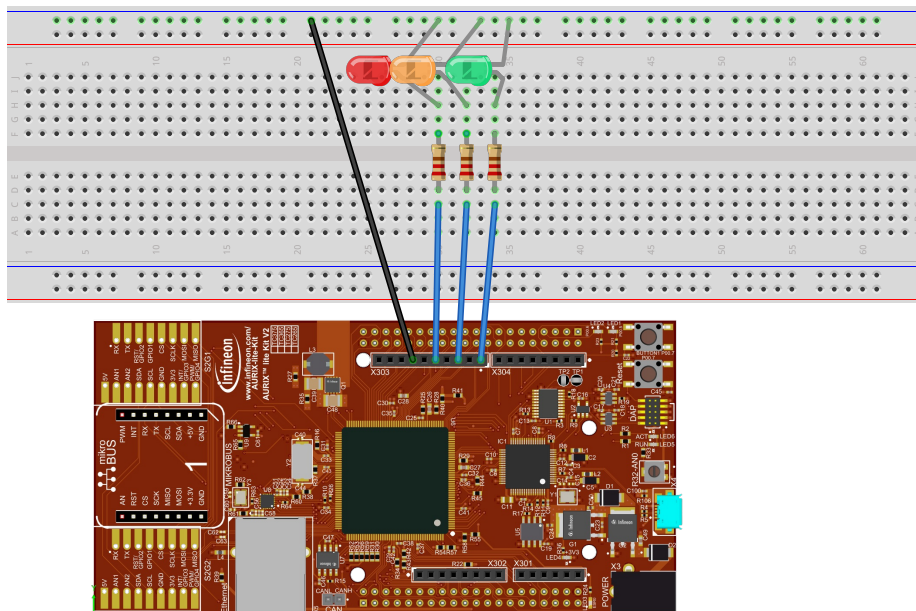
- Liegt keine Aktivierung einer regulären Task unmittelbar bevor, werden die möglichen IDLE-Tasks aufgerufen
- **Öffentliche Schnittstellen** (Funktionen) zur Anwendung:
  - Initialisieren des Schedulers
  - Ausführen des Schedulers
  - Hinzufügen einer Task
  - Löschen einer Task

(b.)

Schreiben Sie ein einfaches Anwendungsprogramm für den Aurix TC375 Lite Kit (siehe Template in OPAL), das drei periodische Tasks unterschiedlicher Zykluszeiten definiert.

- **Task100ms**
- **Task500ms**
- **Task1000ms**

Um die Aufruffrequenzen der Task zu prüfen, soll mit deren Durchlaufen **je eine eigene LED ein- bzw. ausgeschaltet** werden, wie in der folgenden Abbildung aufgebaut.



fritzing