# Echtzeit-Betriebssysteme

---

Im Wesentlichen gibt es drei Familien von Echtzeit-Schedulern.

- non preemptive realtime multitasking (NPRM)
- preemptive realtime multitasking (PRM)
- event-driven realtime multitasking (EDRM)

Wie in der Automatisierungstechnik ist das wesentlichste Kriterium der Determinismus. Ein deterministisches System kann in seinem Zeitverhalten berechnet werden. Ob das System schnell genug für eine Anwendung ist, kann ich ausrechnen und damit garantieren.

## Non-preemptive realtime multitasking

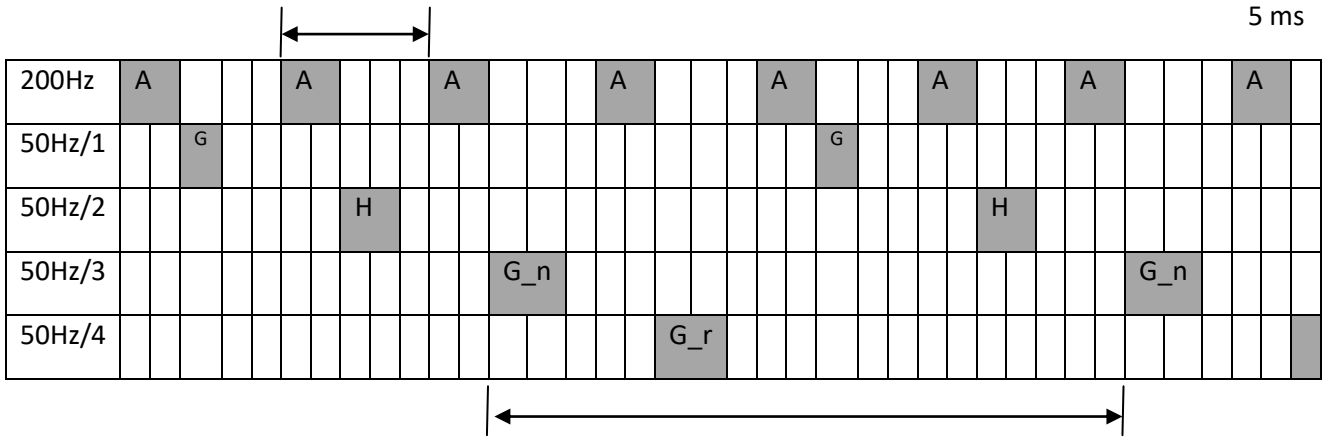
Ein sehr simples Verfahren, in dem die Tasks nicht unterbrochen werden, sondern so geschrieben sind, daß sie nur kurze Rechenintervalle brauchen. Beispiele sind Regler, Meßwert-erfassung und ähnliches in Prozessrechnern oder kleinen Microcontrollern.

Beispiel : der NPRM-Scheduler des tsm-Copters (C++)

( https://www.youtube.com/watch?v=QGgmUf9xNQ8)

```
while(1)
{
                    if (trigger_200Hz ==1)
                    {
                      trigger_200Hz = 0;
                      Flightcontrol.Achsregler();
                      switch (trigger_50Hz)
                      {
                                case 1:
                                {
                                          Flightcontrol.Höhenregler();
                                          trigger_50Hz = 2;
                                }
                                case 2:
                                {
                                          Flightcontrol.Gierregler();
                                          trigger_50Hz = 3;
                                }
                                case 3:
                                {
                                          Flightcontrol.GPS_nick();
                                          trigger_50Hz = 4;
                                }
                                case 4:
                                {
                                          Flightcontrol.GPS_roll();
                                          trigger_50Hz = 1;
                                }
                      }//Ende des 50Hz-Zyklus
                    } //Ende des 200Hz-Zyklus
}//Ende der main-loop
```

Das Flag trigger_200Hz wird von einem Interrupttimer alle 5ms = 1 gesetzt. Damit kommen die Tasks (z.b. Flightcontrol.Achsregle ) alle 20ms dran. Wichtig ist natürlich, daß deren Rechenzeit garantiert unter 5ms liegt Das zeitliche Verhalten läßt sich grafisch darstellen :




A = Achslageregler

G = Gierregler

H = Höhenregler

G_n = GPS-Stabilisierung der Nickachse

G_r = GPS-Stabilisierung der Rollachse

## preemptive realtime multitasking

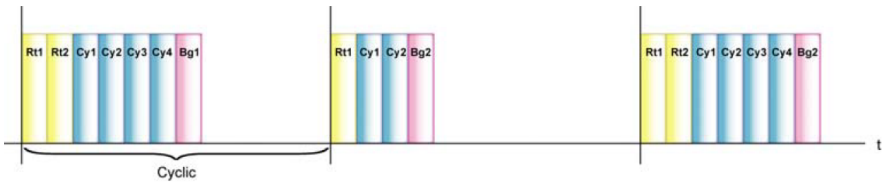
Deutlich anspruchsvoller als beim simplen NPRM braucht es hier einen Scheduler und einen Dispatcher.

Der Entwickler der Software muß das Verhalten seiner Programme kennen. Er legt Prioritäten fest und plant das Scheduling. Als Beispiel hier die Bedienung eines solchen Systems anhand einer modernen SPS von Sigmatec :

In the following graphic, the priority of the different tasks is clearly shown. Real time tasks have the highest priority. Cyclic tasks are processed next. With the remaining time, the operating system calls the background tasks.

Short description of the graphic:

| | | |
|---|---|---|
| Rt | => | Real time work |
| Cy | => | Cyclic work |
| Bg | => | Background |

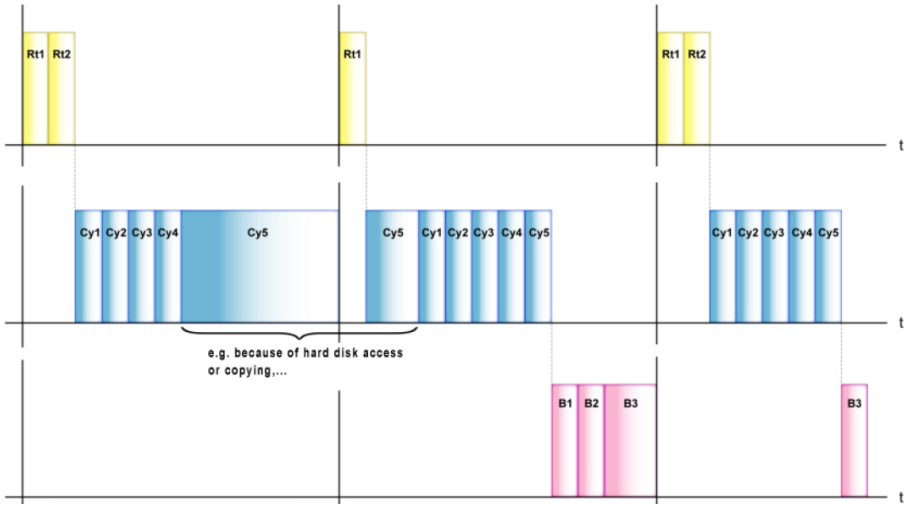



This cycle is dependant on the control and normally averages 1ms for industrial PCs and 2ms for processor modules.

According to this diagram, Rt1, Cy1 and Cy2 are processed every cycle while Rt2, Cy3 and Cy4 are processed every second cycle.

(Quelle : „Sigmatec : Training module Lasal Class.pdf" Seite 62)

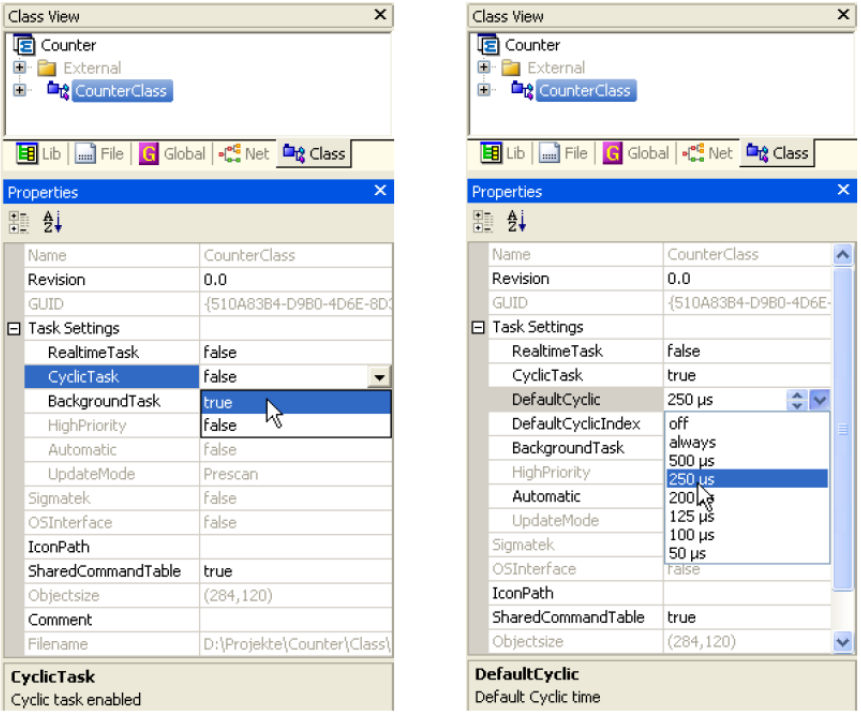
Falls ein Prozess niedriger Priorität für die Ausführung länger braucht, wird er durch die Jobs höherer Priorität unterbrochen




Der zyklische Prozess Cy5 hat länger gebraucht, und wurde deshalb unterbrochen, um die rechtzeitige Ausführung von Rt1 zu ermöglichen. Die niederprioritäten Prozesse B1 bis B3 werden somit nur dann ausgeführt, wenn noch Zeit bleibt.

Der Anwender legt also fest, welche Priorität die verschiedenen Prozesse bekommen, und wie häufig sie ausgeführt werden. Diese Einstellungen sind im Konfigurationsmenü der SPS auszuführen. (Wieder am Beispiel der Sigmatec-SPS gezeigt)

To set a **CyWork Task,** first select the class name in the **Class View** window to open the class **Properties** window then set **CyclicTask** to true. The **CyclicTask** is now activated.




As soon as **CyclicTask** is set to **true** the DefaultCyclic field appears. Here you can define a default **Cyclic time**.