# Der Scheduler

Der Scheduler ist der Kern des Projekts und besteht aus verschiedenen Teilen, auf welche im Verlauf der Dokumentation weiter eingegangen wird. Die Aufgabe des Schedulers besteht darin, Prozesse zu verwalten und sie entsprechend ihrer Priorität auszuführen.

## Datensegmente

Der Scheduler verfügt über mehrere Datensegmente, die intern verwendet werden, jedoch größtenteils von externen Prozessen gesetzt werden.

Die genutzten Datensegmente setzen sich wie folgt zusammen:

* **processTable**  
  processTable stellt die vom Scheduler genutzte Prozesstabelle dar. Der genaue Aufbau kann Kapitel „Aufbau der Prozesstabelle“ entnommen werden.
* **processStartAdress**  
  processStartAdress wird von externen Prozessen beim Anlegen bzw. Löschen eines Prozess dazu verwendet, dem Scheduler die Startadresse des zu startenden bzw. zu löschenden Prozesses mitzuteilen. Dieses Datensegment wird nur in Kombination mit dem newBit verwendet.
* **newBit**  
  Das newBit-Datensegment wird von externen Prozessen gesetzt. Es zeigt dem Scheduler an, ob es einen zu löschenden bzw. anzulegenden Prozess gibt.
* **index**  
  Das index- Datensegment zeigt auf die aktuelle Zeile innerhalb der Prozesstabelle und wird ausschließlich intern verwendet.

## Konstanten

Die konstanten Datensegmente werden dazu verwendet, um auszuführende Aktionen zu kodieren und die Lesbarkeit zu erhöhen. Benutzt werden die folgenden Konstanten:

* **isNew**  
  isNew wird in Kombination mit dem Datensegment newBit verwendet und zeigt an, ob der Scheduler einen neuen Prozess in der Prozesstabelle anlegen soll.
* **isDel**  
  isDel wird in Kombination mit dem Datensegment newBit verwendet und zeigt an, ob der Scheduler einen neuen Prozess aus der Prozesstabelle entfernen soll.
* **isNon**  
  isNon wird in Kombination mit dem Datensegment newBit verwendet und zeigt dem Scheduler an, dass er weder anlegen noch löschen soll.
* **isProcessA**  
  isProcessA wird ausschließlich intern verwendet und entspricht der Adresse des Prozess innerhalb der Prozesstabelle.
* **isProcessB**  
  isProcessB wird ausschließlich intern verwendet und entspricht der Adresse des Prozess innerhalb der Prozesstabelle.
* **isProcessC**  
  isProcessC wird ausschließlich intern verwendet und entspricht der Adresse des Prozess innerhalb der Prozesstabelle.

## Aufbau der Prozesstabelle

Im Projekt wird eine statische, 78 Byte große, Prozesstabelle verwendet, in der die Positionen der potentiellen Prozesse fest vorgegeben sind (vgl. Tabelle 1).



Tabelle 1: Aufbau der Prozesstabelle

Das „ist Aktiv“-Flag wird dafür genutzt um dem Scheduler anzuzeigen, ob der jeweilige Prozess läuft oder nicht. Entsprechend wird es beim Anlegen eines Prozesses auf den Wert 1 gesetzt bzw. beim Stoppen auf den Wert 0.  
Bei der Initialisierung der Tabelle wird die gesamte Spalte auf den Wert 0 gesetzt, da anfangs noch kein Prozess gestartet wurde.

Das Feld „aktueller Stackpointer“ wird verwendet um den Stackpointer eines Prozesses nach Ablauf von dessen Zeitscheibe zu sichern. Wenn der Prozess erneut an der Reihe ist, wird der Stackpointer wiederhergestellt, um sicherzustellen, dass der Prozess an der Stelle weiterläuft, an der er unterbrochen wurde.

Die Prozessstartadresse wird bereits bei der Initialisierung der Tabelle auf die entsprechende Adresse gesetzt und bleibt konstant. Die Adresse wird zur Identifikation eines Prozesses verwendet, wenn dieser gestartet oder gestoppt werden soll.

Die Prozessdaten stellen den Stack des jeweiligen Prozesses dar. Um zu gewährleisten, dass Prozesse ihre Daten in der Tabelle speichern, werden deren Stackpointer anfangs auf eine Adresse relativ zur Startadresse der Prozesstabelle gesetzt.

## Die Interrupt-Routine

Die Interrupt-Routine, die nach Ablauf einer Zeitscheibe ausgeführt wird, enthält die Hauptfunktionalität des Schedulers. In ihr wird zunächst der Stack des unterbrochenen Prozesses in der Tabelle gespeichert. Anschließend wird solange die Prozesstabelle durchlaufen, bis ein Prozess gefunden wird, dessen „ist Aktiv“-Feld gesetzt ist. Ebenfalls werden Prozesse erstellt bzw. gelöscht, falls das newBit entsprechend gesetzt wurde. Nach erfolgreicher Identifikation eines aktiven Prozesses, wird zunächst seine Zeitscheibendauer (Priorität) festgelegt (vgl. Kapitel „Zeitscheiben (Prioritäten)“). Anschließend werden Stackpointer und Stack des Prozess wiederhergestellt und es wird zurück in den Prozess gesprungen, damit dieser seine Arbeit fortsetzen kann.

## Zeitscheiben (Prioritäten)

Im Projekt werden die Zeitscheiben über einen 16-Bit-Timer realisiert, um diesen komfortabel manipulieren zu können. Da in ProzessA ein 13-Bit-Timer (8-Bit mit prescaling) verwendet wird, werden die Zeiten relativ zu diesem verteilt und wurden folgendermaßen berechnet:

Relative Priorität = , mit P = absolute Priorität eines Prozesses

Die hierzu verwendeten, absoluten Prioritäten lauten wie folgt:



Ein geringerer Wert bedeutet eine höhere Priorität. Die Prioritäten wurden so gewählt, dass ProzessA möglichst häufig an der Reihe ist, um die Ausgabe der „a“ möglichst nahe bei einer Sekunde zu halten. Da ProzessB ein relativ kurzer Prozess ist, da keine Endlosschleife enthalten ist, besitzt er die nächst höhere Priorität. Aufgrund der Tatsache, dass Benutzereingaben verhältnismäßig selten sind, hat somit ProzessC die niedrigste Priorität erhalten.

## Mehrfaches Starten eines Prozesses

Wenn ein Prozess bereits gestartet wurde, jedoch der Konsolenprozess einen erneuten Start dieses Prozesses initiiert, wird der Prozess innerhalb der Prozesstabelle zurückgesetzt, da er überschrieben wird.