In dieser Übung setzen wir als Echtzeit-Betriebssystem für die Mindstorm Roboter eine Implementierung des OSEK-Standards (http://www.osek-vdx.org) ein, nämlich NXT-OSEK (http://lejos-osek.sourceforge.net). Die Applikationen werden unter Standard-C geschrieben. Sie können unter

http://lejos-osek.sourceforge.net/api.htm

eine Dokumentation der API finden. Funktionen des Betriebssystems sind im OSEK-Standard definiert, den Sie auf Moodle oder unter

http://portal.osek-vdx.org/index.php?option=com_content&task=view&id=11&Itemid=14

finden können.

Aufgabe 1): Das erste eigene Programm unter NXT-OSEK

Arbeiten Sie das NXT-OSEK-Tutorial, welches Sie auf Moodle finden, durch und erstellen Sie Ihr erstes Programm unter NXT-OSEK.

Aufgabe 2): Aktivierung und Terminierung von Tasks

An diesem Beispiel sollen die Möglichkeiten zur Aktivierung und Terminierung von Tasks getestet werden. Das System besteht aus einer Task *Main*, den Tasks *One*, *Two*, *Three* und einer Task *Monitor*. Die Task *Monitor* besitzt die geringste Priorität, die anderen Tasks besitzen alle dieselbe, höhere Priorität. Da das AUTOSTART-Attribut bei der Task *Main* auf **`TRUE`** gesetzt ist, wird diese Task beim Systemstart automatisch aktiviert. *Main* startet die Task *Monitor* (mittels **`ActivateTask(Monitor)`**) sowie die Task *One* und anschließend die Task *Three*. Danach beendet sie sich mit einem Aufruf von **`ChainTask(Two)`**, wodurch die Task *Two* mit dem Beenden von *Main* in den ablaufbereiten Zustand überführt wird.

Zu beobachtender Ablauf: Da zu diesem Zeitpunkt jedoch bereits Task *One* und *Three* in der FIFO-Warteschlange des Schedulers stehen, werden sie auch in dieser Reihenfolge ausführend (*One*, *Three*, *Two*). Danach wird Task *Two* ausführend. Die Task *Monitor* wird erst ausgeführt, nachdem auch Task *Two* abgearbeitet ist und keine höherprioren Tasks mehr ablaufbereit sind. Sie beendet die Applikation.

Um dies beobachten zu können, soll jede Task ihren Namen in das Display schreiben. Zum Schreiben in das Display können Sie die folgende Funktion benutzen:

```
void myPrintln(const char *s)
{
   static int y = 0;
   display_goto_xy(0,y);
   display_string(s);
   display_update();
   y++;
}
```

Damit sollten Sie z.B. die folgende Ausgabe auf dem Display sehen:

```
Main
One
Three
Two
Monitor
```

Aufgabe 3): Zyklische Aktivierung

Für zyklische Aktivierung von Tasks benötigen Sie Alarme und Counter. Die Task (in unserem Fall *Main*) wird abhängig vom Counter durch einen Alarm auf ausführend gesetzt. Um in NXT-OSEK einen Timer benutzen zu können, müssen Sie das Weiterschalten des Timers über die 1ms-Basis ISR implementieren, die wir bisher leer gelassen hatten. Diese wird nun erweitert zu:

```
void user_1ms_isr_type2(void)
{
      StatusType ercd;
      ercd = SignalCounter( CounterName );
      if( ercd != E_OK ){
            ShutdownOS( ercd );
      }
}
```

wobei *CounterName* der Name des Counters ist, den Sie im OIL-File definiert haben. Counter müssen dem System im Applikationscode mit **`DeclareCounter(`*`CounterName`*`);`** bekannt gemacht werden.

a) ***Vordefinierte Zyklen:*** In dieser Teilaufgabe sollen Sie eine Task *Main* erstellen, die alle 200ms einen Ton der Länge 20ms abspielt (mittels der Funktion **`ecrobot_sound_tone(`***`Frequenz,Länge,Lautstärke`***`)`**). Hierzu definieren Sie im OIL-File einen Alarm, der die Task *Main* aktiviert (**`ACTION = ACTIVATETASK`**) und gleich wieder selbst aktiviert wird (**`AUTOSTART = TRUE`**). Die Task *Main* selbst soll initial nicht starten (**`AUTOSTART = FALSE`**).

b) ***Dynamische Zyklen:*** Schreiben Sie Ihre Implementierung aus Aufgabe a) so um, dass die Dauer des Alarms von der Task *Main* selbst bestimmt wird und sich zur Laufzeit ändert. Hierzu dürfen Sie den Alarm nicht mehr automatisch starten, Task *Main* hingegen muss initial gestartet werden. *Main* wird dann mittels der Funktion **`SetRelAlarm(…)`** den Timer jeweils neu starten. Erniedrigen Sie den Wert des Timers nach jedem Aufruf bitte um 1ms.

Aufgabe 4): Aktivierung durch Events

Hier soll die Verwendung von Events getestet werden. Das zu erstellende Tasksystem soll aus der Task *Main* mit der Priorität 10 und der Task *One* mit der Priorität 11 bestehen. Die Task *One* besitzt die Events *evt_one* und *evt_two*. Sie wartet nach ihrem Start auf das Event *evt_one*. Die Task *Main*, die nach der Aktivierung von Task *One* aufgrund der höheren Priorität unterbrochen wurde, sendet nun das Event *evt_one* und gibt eine entsprechende Meldung aus. Danach wiederholt sich der Vorgang für das zweite Event *evt_two*. Die Task *One* gibt nach jedem Empfang eines Events eine entsprechende Meldung aus. Für die Arbeit mit Events stehen die Funktionen **`SetEvent(…)`** und **`WaitEvent(…)`** zur Verfügung. Zusätzlich muss in der Implementierung jedes Event mit **`DeclareEvent(`*`Eventname`*`)`** deklariert werden. Verwendung Sie im OIL-File bitte das Attribut MASK = AUTO bei der Deklaration der Events. Für die Ausgabe können Sie wieder die Funktion *myPrintln* aus Aufgabe 2 benutzen. Die Ausgabe sollte z.B. so aussehen:

```
Main started
One started
Main send evt_one
One rec evt_one
Main send evt_two
One rec evt_two
```

Aufgabe 5): Anwendungsbeispiel

Als Beispiel soll ein Aufbau dienen, der mit einem oder zwei Motoren zur Fortbewegung und 2 Tastsensoren ausgestattet ist. Die Applikation, die Sie implementieren sollen, soll das folgende leisten:

- Bei Betätigung des linken Tasters soll der Roboter für 1 Sekunde mit einer Motorkraft von 35% in die definierte Fahrtrichtung fahren und dann wieder stoppen
- Bei Betätigung des rechten Tasters soll die Fahrtrichtung für das nächste Mal fahren verändert werden (es soll nicht bereits in die andere Richtung gefahren werden, auch wenn schon gefahren wird und es soll nicht gefahren werden, wenn der Roboter steht; es soll nur die Richtung für das nächste Mal fahren geändert werden)
- Initial soll die Fahrtrichtung auf „Vorwärts“ stehen

Führen Sie hierfür bitte die folgenden Aufgaben durch:

a) Erstellen Sie zwei Tasks, die die beiden Tastsensoren abfragen und genau dann, wenn der jeweilige Tastsensor gedrückt (nicht losgelassen) wird, ein entsprechendes Event erzeugen. Die Tastsensoren sollen mit einer Periode von 100ms abgefragt werden.
b) Implementieren Sie nun eine Task, die die oben angegebene Aufgabe durchführt. Beachten Sie, dass Sie den Timer für die Zeitspanne von 1s irgendwo starten müssen. Warten Sie in dieser Task auf die jeweiligen Events und führen Sie die notwendigen Aufgaben dann durch.