

SynthOS2

Ein Nanokernel

1. Funktionsbeschreibung

SynthOS2 ist ein kleines Echtzeitbetriebssystem für nicht unterbrechbare Tasks. Es ist nur eine Task definiert welche die LED am Port 1.0 blinken lässt. Es ist auch möglich Messages zwischen den Tasks auszutauschen.

2. Hardwarebeschreibung

Als Hardware kann auch das Elektor R8C/13 Application Board verwendet werden.

3. Softwarebeschreibung

Um Missverständnisse zu vermeiden: Bei der Erstellung des Projekts wurde ein Heap von 0x200 Bytes zugewiesen, da das Betriebssystem Funktionen malloc() und free() nutzt.

Ferner braucht das System den Timer X, und es verbiegt dessen Interruptvector (vector 22) auf den Scheduler. Daher sind neue Projekte ähnlich anzulegen!

Das Betriebssystem setzt den Timer auf eine bestimmte Tickfrequenz (im Archiv ~10Hz). Wenn eine Task nicht schlafen gelegt wurde, wird sie bei jedem Tick einmal ausgeführt.

Die nutzbaren Funktionen und Strukturen sind in der Datei RTOS.h abgelegt und in der Datei RTOS.c implementiert. In der Datei SynthOS2 wird gezeigt wie man Tasks erstellt. Mehr über jede einzelne Funktion:

void setup_rtos(void) Führt die Initialisierung der für das Betriebssystem nötigen Geräte (hier z.B. Timer X) durch

void scheduler(void) Scheduler im RT-Betriebssystem (hier Round Robin)

void add_task(const int id, void (*code)(void)) Erstellt eine neue Task aus einer Funktion der Art **void funktion(void)**. Aufzurufen mit TASK_ID (Integer-Konstante, wie z. B. **#define TASK1_ID 1**) als erstes Argument und der Funktion als zweites.

void delete_task(const int id) Löscht die Task mit ID TASK_ID.

void sleep_task(const int id, unsigned int ticks) Lässt eine Task **ticks** Ticks schlafen.

void send_message(const int id, int value) Schickt eine Message mit Wert **value** und Message-ID **id**.

int receive_message(const int id) Empfängt die Message und löscht diese.

unsigned int check_messages(const int id) Prüft, wieviele Messages mit der entsprechenden Message-ID noch vorhanden sind.

4. Nutzung

Überall wo ein kompaktes und verständliches Echtzeitbetriebssystem gebraucht wird.