**基于 OSEK-OS 的多任务系统设计与实现**

**一、实验目的**

本实验旨在通过开发一个 OSEK-OS 操作系统下的嵌入式多任务应用，帮助学生掌握如下技能和知识点：

理解 OSEK-OS 实时操作系统结构及任务管理机制；

掌握基本任务（Basic Task）与扩展任务（Extended Task）的区别与用途；

学习任务调度、优先级分配及事件机制的基本操作；

设计并实现任务之间的同步与通信；

培养系统抽象建模能力与代码组织能力。

**二、实验环境**

| 项目 | 内容 |
| --- | --- |
| 操作系统 | OSEK-OS v2.2.3 |
| 编程语言 | C 语言 |
| 编译器/IDE | OSEK Plugin |

**三、实验内容与要求**

根据实验要求，需设计一个 OSEK-OS 系统应用，满足以下功能：

创建 14 个任务，每个任务具有唯一优先级；

每个任务执行时打印输出信息；

选择其中 7 个任务用于任务间通信；

选择另外 7 个任务通过 事件机制进行同步。

**四、设计与实现**

**4.1 系统总体结构设计**

整个系统可分为三个子系统：

调度子系统：调度和切换各个任务；

通信子系统：通过共享缓冲区模拟任务间的点对点通信；

同步子系统：使用事件(Event)机制控制任务执行时序；

**4.2 OIL 文件配置简述**

OS embeddedOS {

STATUS = EXTENDED;

STARTUPHOOK = FALSE;

ERRORHOOK = FALSE;

SHUTDOWNHOOK = FALSE;

PRETASKHOOK = FALSE;

POSTTASKHOOK = FALSE;

USEGETSERVICEID = FALSE;

USERESSCHEDULER = FALSE;

USERESSOURCE = TRUE;

TASK Task1 {

PRIORITY = 1;

AUTOSTART = TRUE;

ACTIVATION = 1;

SCHEDULE = FULL;

STACK = SHARED;

TYPE = BASIC;

};

TASK Task3 {

PRIORITY = 3;

AUTOSTART = TRUE;

ACTIVATION = 1;

SCHEDULE = FULL;

STACK = SHARED;

TYPE = EXTENDED;

EVENT = DataReadyEvent;

};

};

**4.3 通信机制设计（7 个任务）**

使用结构体 + 全局缓冲区实现

typedef struct {

float temperature;

float pressure;

} SensorData;

SensorData sharedBuffer;bool dataReady = false;

示例：Task3 → Task5

TASK(Task3) {

sharedBuffer.temperature = readTemperatureSensor();

dataReady = true;

printf("Task3数据已写入共享缓冲区。\n");

TerminateTask();

}

TASK(Task5) {

if (dataReady) {

float temp = sharedBuffer.temperature;

printf("Task5: 接收到 Task3 数据 ");

dataReady = false;

}

TerminateTask();

}

通信链示意：

Task3 → Task5 → Task6 → Task7 → Task8

**4.4 同步机制设计（7 个任务）**

每组使用一个事件标志位作为同步信号。

示例：Task9 → Task10 同步

TASK(Task9) {

printf("Task9: 正在执行...\n");

SetEvent(Task10, SyncEvent1);

TerminateTask();

}

TASK(Task10) {

WaitEvent(SyncEvent1);

ClearEvent(SyncEvent1);

printf("Task10: 收到 Task9 事件，继续执行。\n");

TerminateTask();

}

同步组设定：

EventGroup1：Task9 ↔ Task10

EventGroup2：Task11 ↔ Task12

EventGroup3：Task13 ↔ Task14

**4.5 系统主函数**

int main(void) {

StartOS(OSDEFAULTAPPMODE);

return 0;

}

**五、测试与验证**

**5.1 实验现象**

**5.2 正确性验证**

所有任务被调度执行；

输出信息对应任务执行逻辑；

通信链路稳定，数据传递正确；

事件同步工作正常，任务按预定顺序触发。

**六、实验总结与体会**

本实验深入实践了 OSEK-OS 的任务调度与管理机制，加深了对多任务嵌入式系统架构的理解；实验中的任务通信采用了结构体与共享缓冲区的方法，既简单又实用，适合资源受限平台；使用事件同步机制对任务执行顺序进行控制，是构建稳定逻辑流程的关键；OSEK 的任务优先级管理机制需要精确把控，否则容易造成任务饥饿或执行异常；在实际项目中还需关注任务执行时间、调度开销与系统负载等性能指标。