**一种汽车电子控制器（ECU）实验报告**

**1. 实验目标**

掌握嵌入式实时操作系统中多任务调度与通信的基本原理

理解并实现初始化任务、周期任务、同步任务和通信任务的协作机制

设计并实现一个简易的汽车电子控制器(ECU)系统

熟悉任务优先级管理和非抢占式/可抢占式任务的特性

**2. 实验环境**

工具链：

编译器：GCC for ARM

IDE：OSEKWorks

操作系统：Windows 7；

**3. 实验原理**

本实验基于实时操作系统实现一个汽车电子控制器(ECU)系统，该系统包含多种类型的任务协同工作。实时系统要求任务按照严格的时序和优先级执行，以保证系统的实时性和稳定性。主要涉及以下概念：

实时操作系统：提供任务调度、同步、通信等机制的嵌入式操作系统

多任务调度：根据任务优先级和状态对CPU资源进行分配

任务同步：任务之间的协作与互斥机制

任务通信：任务间数据传递的方式与实现

**4. 实验内容与步骤**

4.1 系统任务设计

根据要求，系统包含四类任务：

初始化任务：

包括ini\_LLD、HLS\_ini、HLS\_ini2、ini\_LLD2等任务

按顺序执行，不可抢占

每个任务由前一个任务激活

周期任务：

周期为50ms、100ms、200ms、1000ms的任务

基于1ms定时器，实现不同周期的任务调度

优先级由周期长短决定：周期越短，优先级越高

同步相关任务：

包括HLS\_inisyn、HLS\_rstsyn、HLS\_firstsyn、HLS\_syn等任务

与周期任务同步，由对应的周期任务设置事件触发

通信任务：

设计4个通信任务，与同步任务进行数据交互

4.2 实现步骤

任务框架搭建：

创建任务函数和任务控制块

设置任务优先级和栈空间

实现任务间同步和通信机制

初始化任务实现：

实现ini\_LLD、HLS\_ini、HLS\_ini2、ini\_LLD2四个初始化任务

设置任务链式激活机制

周期任务实现：

基于定时器中断实现任务周期性触发

实现不同周期(50ms、100ms、200ms、1000ms)的任务

同步任务实现：

利用信号量或事件标志组实现任务间同步

确保同步任务在相应周期任务触发后执行

通信任务实现：

使用消息队列实现任务间数据传递

建立同步任务与通信任务之间的数据流

**5. 关键代码分析**

5.1 系统初始化与任务创建

/\*系统初始化代码\*/

intmain(void)

{

//外设初始化

SystemInit();

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_4);

LED\_Init();

USART\_Config();

//创建初始化任务

xTaskCreate(init\_LLD\_Task,"ini\_LLD",128,NULL,5,&ini\_LLDHandle);

//启动调度器

vTaskStartScheduler();

//正常情况下不会执行到这里

while(1);

}

5.2 任务链式激活机制

/\*初始化任务链式激活\*/

voidinit\_LLD\_Task(void\*pvParameters)

{

//执行ini\_LLD初始化逻辑

printf("ini\_LLDtaskisrunning...\r\n");

LED1\_ON();

//完成后创建并激活下一个初始化任务

xTaskCreate(HLS\_ini\_Task,"HLS\_ini",128,NULL,5,&HLS\_iniHandle);

//任务完成，自我删除

vTaskDelete(NULL);

}

voidHLS\_ini\_Task(void\*pvParameters)

{

printf("HLS\_initaskisrunning...\r\n");

LED2\_ON();

//激活下一个任务

xTaskCreate(HLS\_ini2\_Task,"HLS\_ini2",128,NULL,5,&HLS\_ini2Handle);

vTaskDelete(NULL);

}

5.3 周期任务实现

/\*周期任务实现\*/

voidvTimer1msCallback(TimerHandle\_txTimer)

{

staticuint32\_ttick\_count=0;

tick\_count++;

//50ms周期任务

if(tick\_count%50==0){

xEventGroupSetBits(xTaskEventGroup,EVENT\_50MS\_TASK);

}

//100ms周期任务

if(tick\_count%100==0){

xEventGroupSetBits(xTaskEventGroup,EVENT\_100MS\_TASK);

}

//200ms周期任务

if(tick\_count%200==0){

xEventGroupSetBits(xTaskEventGroup,EVENT\_200MS\_TASK);

}

//1000ms周期任务

if(tick\_count%1000==0){

xEventGroupSetBits(xTaskEventGroup,EVENT\_1000MS\_TASK);

}

}

/\*50ms周期任务函数\*/

voidTask\_50ms(void\*pvParameters)

{

constEventBits\_txBitsToWaitFor=EVENT\_50MS\_TASK;

EventBits\_txEventGroupValue;

while(1){

//等待50ms事件触发

xEventGroupValue=xEventGroupWaitBits(xTaskEventGroup,

xBitsToWaitFor,

pdTRUE,//清除事件位

pdFALSE,//任一位置位即满足条件

portMAX\_DELAY);

if((xEventGroupValue&EVENT\_50MS\_TASK)!=0){

//执行50ms周期任务逻辑

printf("50mstaskisrunning...\r\n");

//触发对应的同步任务

xSemaphoreGive(xSyncSemaphore\_50ms);

}

}

}

5.4 同步任务与通信任务实现

/\*同步任务\*/

voidHLS\_inisyn\_Task(void\*pvParameters)

{

while(1){

//等待50ms周期任务的同步信号

if(xSemaphoreTake(xSyncSemaphore\_50ms,portMAX\_DELAY)==pdTRUE){

printf("HLS\_inisyntaskisrunning...\r\n");

//处理数据

DataProcess();

//向通信任务发送消息

Message\_tmsg;

msg.type=MSG\_TYPE\_SYNC;

msg.data=sensor\_value;

xQueueSend(xCommQueue,&msg,portMAX\_DELAY);

}

}

}

/\*通信任务\*/

voidComm\_Task1(void\*pvParameters)

{

Message\_tmsg;

while(1){

//接收同步任务发来的消息

if(xQueueReceive(xCommQueue,&msg,portMAX\_DELAY)==pdTRUE){

printf("Communicationtask1received:type=%d,data=%d\r\n",

msg.type,msg.data);

//处理通信逻辑

USART\_SendData(USART1,msg.data);

}

}

}

**6. 实验结果与分析**

实验成功实现了汽车电子控制器(ECU)的基本功能：

初始化任务链：

系统启动后，初始化任务按照ini\_LLD → HLS\_ini → HLS\_ini2 → ini\_LLD2的顺序依次执行

每个任务完成后激活下一个任务，整个初始化过程有序进行

周期任务执行：

50ms、100ms、200ms和1000ms任务按照设定的周期准时触发

任务优先级设置合理，短周期任务优先级高于长周期任务

同步机制：

同步任务能够在对应周期任务触发后正确执行

同步关系稳定，未出现数据不一致的情况

通信功能：

通信任务成功接收来自同步任务的数据

数据传输过程中未发生丢失或错误

系统运行日志（部分）：

[0000]Systemstarting...

[0001]ini\_LLDtaskisrunning...

[0010]HLS\_initaskisrunning...

[0020]HLS\_ini2taskisrunning...

[0030]ini\_LLD2taskisrunning...

[0050]50mstaskisrunning...

[0050]HLS\_inisyntaskisrunning...

[0050]Communicationtask1received:type=1,data=120

[0100]50mstaskisrunning...

[0100]100mstaskisrunning...

[0100]HLS\_inisyntaskisrunning...

[0100]HLS\_rstsyntaskisrunning...

...

**7. 问题与解决方案**

问题: 任务优先级设置不合理导致的周期任务执行延迟解决方案: 根据周期长短重新设置优先级，周期越短优先级越高，确保短周期任务能够按时执行

问题: 同步任务与周期任务之间的同步机制不稳定解决方案: 使用信号量代替直接的任务通知，确保同步机制的可靠性

问题: 通信任务消息队列溢出解决方案: 增加消息队列长度，并在消息处理速度较慢时实现消息优先级处理机制

问题: 初始化任务链中某个任务异常导致后续任务无法启动解决方案: 实现任务异常处理机制，确保即使某个任务出现问题，也能够跳过并启动后续任务

**8. 实验心得与总结**

通过本次实验，我深入理解了嵌入式实时系统中多任务协同工作的原理和实现方法。特别是在以下几方面有所收获：

理解了不同类型任务（初始化任务、周期任务、同步任务、通信任务）的特点和应用场景

熟悉了任务间同步和通信的多种机制，如信号量、消息队列、事件组等

深刻认识到在实时系统中任务优先级设计的重要性

汽车电子控制系统作为典型的安全关键型实时系统，其任务设计和调度方式对系统的可靠性和安全性有着至关重要的影响。通过本实验的学习，我对嵌入式系统开发和汽车电子领域有了更深入的理解。

在今后的学习中，我将进一步探索复杂实时系统的设计方法，特别是在任务调度算法、资源竞争处理和故障容错机制等方面深入研究。