第二十一天笔记

1. RTOS的概念

实时操作系统也被称为RTOS（Real Time Operating System），实时操作系统（RTOS）是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统做出快速响应，调度一切可利用的资源完成实时任务，并控制所有实时任务协调一致运行的操作系统。提供及时响应和高可靠性是其主要特点。

**实时操作系统是保证在一定时间限制内完成特定功能的操作系统**。如果按照时间的精准度而言，实时操作系统有**硬实时**和**软实时**之分，硬实时要求在规定的时间内必须完成操作，这是在操作系统设计时保证的；软实时则只要按照任务的优先级，尽可能快地完成操作即可。我们通常使用的操作系统在经过一定改变之后就可以变成实时操作系统

程序的运行方式

* 轮询系统

指的是在程序运行时，首先对所有的硬件进行初始化，然后在主程序中写一个死循环，需要运行的功能按照顺序进行执行，轮询系统是一种简单可靠的方式，一般适用于在只需要按照顺序执行的并且没有外部事件的影响的情况下。

程序的运行过程中出现如按键等需要外部检测的事件，轮询系统的实时响应能力变得很差。

int main(void)

{

//对硬件进行初始化

//编写死循环

while(1)

{

//程序1

//程序2

.....

}

}

* 前后台系统

相比于轮询系统，前后台系统增加**中断**的概念，如果外部事件发生，则在中断中进行处理，主程序在轮询系统中运行，**中断被称为前台**，**主程序中的while(1)就称为后台**。中断会终止后台程序的运行，然后跳转到对应的中断服务函数中去处理，处理完成后，在继续执行后台的程序。

如果使用前后台系统，会大大的提高程序的实时响应能力，避免造成外部事件的缺失。

int main(void)

{

//硬件的初始化

//注册中断，设置中断的触发条件

//在死循环进行执行后台程序

while(1)

{

//程序1

//程序2

.....

}

}

//中断服务函数

void XXX\_IQR(void)

{

//处理和响应中断

//返回主程序进行运行

}

* 多任务系统

相比于前后台系统，多任务系统的外部事件也是在中断中进行响应，但是外部事件的处理是任务中进行处理。任务具有优先级，优先级高的任务先处理，所以程序就会被分割为一个个的任务，任务是一个独立的死循环，并且不能返回，可以由操作系统进行任务的调度，程序段的实时响应能力又得到提升。

void task1(void) //任务1

{

while(1)

{

//执行任务1的功能

}

}

void task2(void) //任务2

{

while(1)

{

//执行任务2的功能

}

}

int main(void)

{

//对硬件进行初始化

//注册中断，设置中断的触发条件

//任务调度，由操作系统进行任务调度

//在死循环进行执行后台程序

while(1)

{

}

}

1. RTOS的种类

实时操作系统有很多的种类，比如常用的UCOSIII、FreeRTOS、RT-thread，各有特色，可以根据自己的实际需求去选择对应的操作系统来进行使用。

1. RTOS的移植
2. RTOS的使用
3. RTOS的拓展

练习：把自己的项目的代码先封装好，然后利用前后台系统实现对应的功能，然后优化代码，移植UCOSIII实时操作系统（可以使用已经移植好的代码），利用RTOS来实现代码的相关功能（通过创建多个任务的方式实现任务的运行）。