嵌入式小书3-嵌入式平台软件搭建 | 觅知圈博客-觅的精彩 知而不凡 <https://blog.mizhiquan.com/the-little-embedded-system/ch3-embedded-system-platform>

定时任务调度

为了克服大循环方式的缺点（任务调度周期性无法保证，任务数量增加系统会变慢），提出了定时的任务调度的方式，不过需要使用单片机一个定时器，来实现一个简单的任务调度器，利用定时器将CPU切割为一个等周期的时间片调度单元，然后利用标志位控制在每个时间片只调用一个任务。整个系统代码结构如下所示：

#define TASK\_MAX\_LENGTH 10

typedef struct

{

Int16 Flag[TASK\_MAX\_LENGTH];

Int16 Timer;

Int32 Number;

} USERTASK;

USERTASK UserTask0={0,0,0,0,0,0,0,0,0,0,0,TASK\_MAX\_LENGTH};//任务初始化

//任务调度函数

void TaskScheduler(USERTASK\* v)

{

v->Flag[v->Timer++] = 1;

if(v->Timer >= v->Number)

{

v->Timer = 0;

}

}

//主函数

int main()

{

Dis\_Interrupt();

System\_Init();

En\_Interrupt();

while(1)

{

Task0\_Run();

Task1\_Run();

Task2\_Run();

Task3\_Run();

Task4\_Run();

}

}

//1ms定时中断

\_\_interrupt void Timer0\_INT\_MapedISR(void)

{

TaskScheduler(&UserTask0);

}

//单个任务示例函数

void Task0\_Run(void)

{

if(UserTask0.Flag[0])

{

Pot1Calc(); //加速器信号计算

Pot2Calc(); //制动器信号计算（保留）

TempCalc(); //电机及控制器温度计算

UserTask0.Flag[0] = 0;

}

}

......

定时任务调度的流程图如图2所示。与大循环调度方式对比，这种方式能够实现周期性的任务调度，同时随着任务的增加，依然能够保证调度的周期性，这种调度能够应对大多数的控制系统，比如TI的PMSM电机控制器，一般小的家电控制器，都可以搞定。但是使用时有几点要注意：

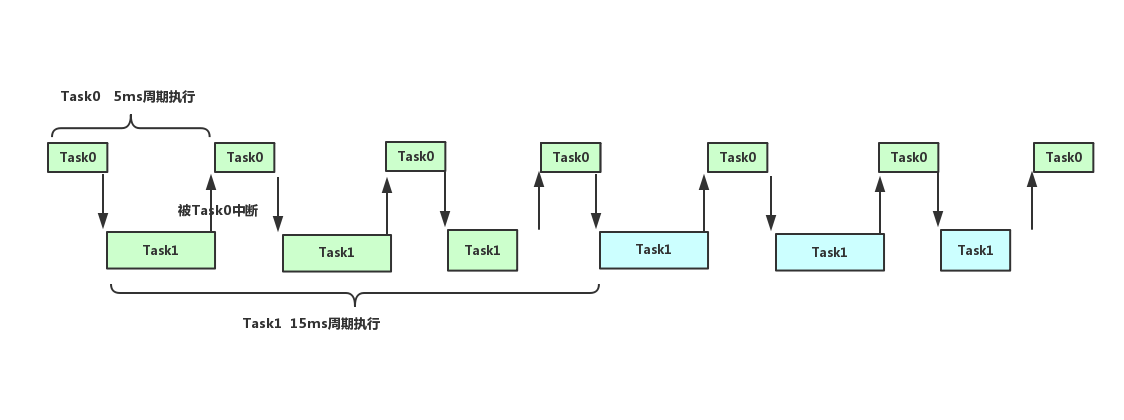
1. 单个任务的最长时间长度务必保证不超过单个时间片，否则会导致周期性延迟
2. 对于严格实时的控制周期任务，定时调度器不能够保证
3. 对于长周期任务（比如通讯等待等），定时任务调度器要么把任务切割为小任务，要么安排几个连续的空闲周期来执行

针对第1点，需要测试或者预估任务的最长执行时间，这个可以采用IO测试的方式解决（具体参见ch6）。

针对第2点，对于实时性要求高，并且周期控制快的任务（比如PID控制），只能将这个任务放到定时中断里做

针对第3点，我们可以将长周期任务放在最后面，如图3所示，可以把最后几个空闲周期都留给Task4执行。但是要注意，如果有多个长周期任务，依然会拖慢整个调度周期，于是就出现了基于优先级的任务调度方式，高优先级的任务可以中断低优先级的任务，在保证长周期任务调度的同时，短周期任务的调度依然能够保证，这就是RTOS。(Staok:大概短周期的优先级高，长周期的任务优先级低一些，但是还得根据重要程度按照实际情况而定)

#### 实时操作系统RTOS调度

实时操作系统，常用的小型RTOS有uCosII，FreeRTOS，Rt-thread，主要是任务优先级的调度方式不一样，这里感兴趣的同学，可以参见相关的专业书籍，对RTOS内核代码不做详细介绍。RTOS的对任务的调度方式如图4所示。Task0的优先级高，可以中断优先级低的Task1，等Task0执行完，然后RTOS会切换到Task1继续执行。

嵌入式小书(智能车)介绍 <https://zhuanlan.zhihu.com/p/32993527?iam=f2bb3a090b079bb51de9e2d79e95fd41&utm_source=qq&utm_medium=social>