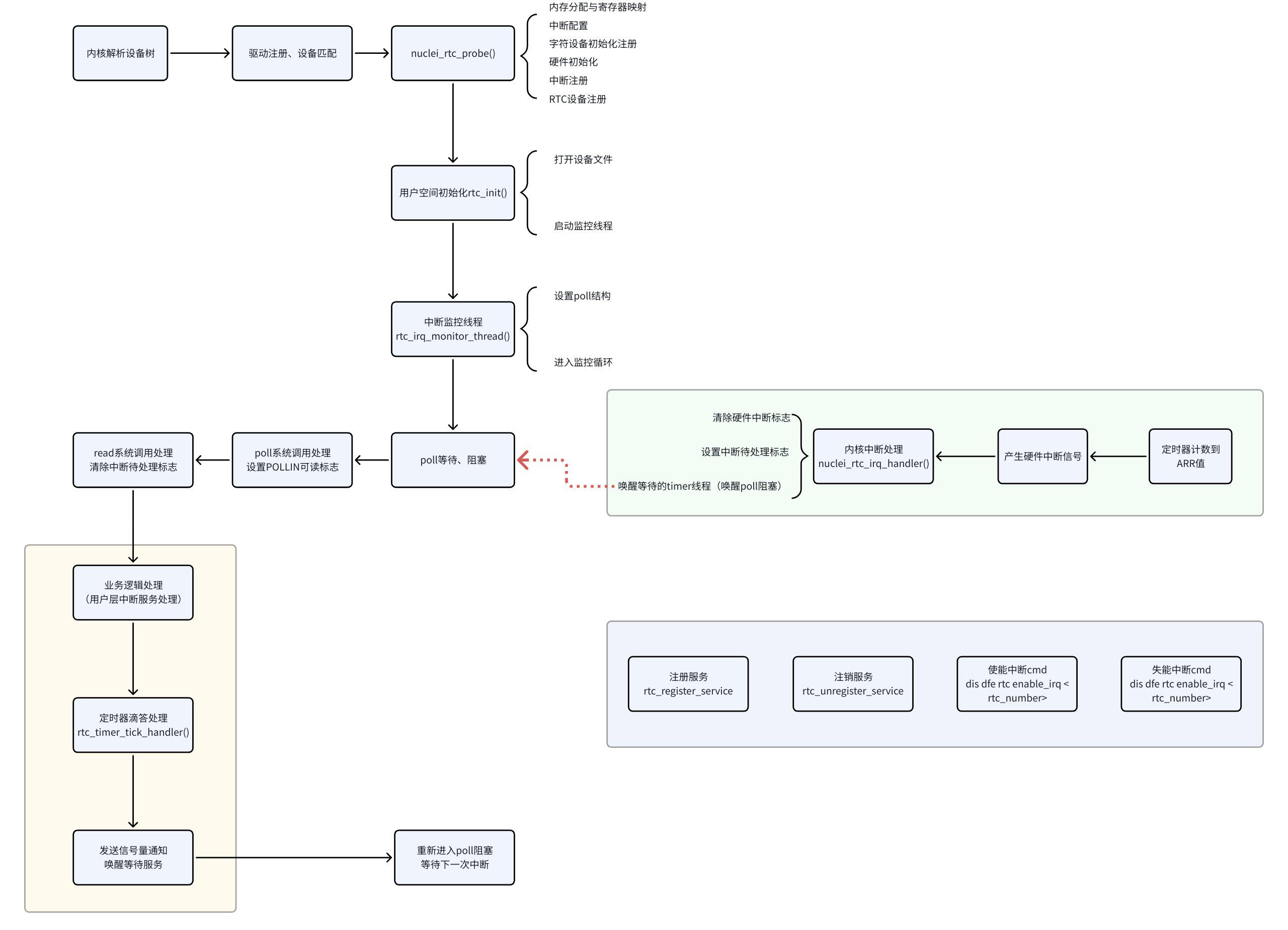
**定时器驱动配置说明**

**一、RTC驱动完整时间线**



**二、RTC驱动配置**

**2.1 设备树配置**

|  |
| --- |
| C++ #ifdef TIMER  btimer0:btimer@f01e0000 {  compatible = "nuclei,rtc";  reg = <0x0 0xf01e0000 0x0 0x1000>;  clocks = <&periphclk>;  counter-value = <1000000>; *//单位为us*  interrupt-parent = <&plic0>;  interrupts = <13>;  status = "okay";  };   btimer1:btimer@f01e2000 {  compatible = "nuclei,rtc";  reg = <0x0 0xf01e2000 0x0 0x1000>;  clocks = <&periphclk>;  counter-value = <1000000>; *//单位为us*  interrupt-parent = <&plic0>;  interrupts = <14>;  status = "okay";  }; #endif |

配置了两个定时器的设备节点，在/dev下表示为nuclei\_rtc0，nuclei\_rtc1，分别设置其硬件中断号为13和14，可在/proc/interrupt或/proc/irq下面查看。其中counter-value可配置定时器的中断触发间隔时间，单位为us。目前配置为1000’000us，即1s。推荐中断触发间隔设置为秒级，最多可到100ms，这样保证触发间隔误差控制到us级。

**2.2 服务线程调用Demo**

|  |
| --- |
| C++ *// 任务1* void **task1\_callback**(void) {  **sleep**(1);  **printf**("Task1: Complex work completed\n"); }  *// 任务2* void **task2\_callback**(void) {  **sleep**(3);  **printf**("Task3: System monitoring completed\n"); }  void **timerTestSimpleInit**() {  *// 注册到timer0*  if (**rtc\_register\_service**(0, "SimpleTask1", 1, **task1\_callback**) != 0) {  **printf**("Failed to register SimpleTask1\n");  return;  }    *// 注册到timer1*  if (**rtc\_register\_service**(1, "SimpleTask2", 5, **task2\_callback**) != 0) {  **printf**("Failed to register SimpleTask2\n");  return;  } } |

上面是注册回调函数的Demo，通过**rtc\_register\_service**接口可以注册定时器的回调函数。完整的接口如下：int **rtc\_register\_service**(int timer\_id, const char \*name, int interval, void (\*callback\_func)(void))

* timer\_id表示注册到哪个定时器设备，0表示timer0，1表示timer1；
* name表示服务线程的名字；
* interval表示服务线程的唤醒阈值，定时服务的总时间间隔为：interval\* counter-value；
* callback\_func表示注册的回调函数；

**2.3 RTC中断使能**

提供了cmd命令来启用定时器中断，使用dis dfe rtc enable\_irq <rtc number>使能定时器中断；使用dis dfe rtc disable\_irq <rtc number>失能定时器中断。