FAQ0059

常见问题解答

系统时钟初始化at32f4xx_clock.c中HEXT启动说明

Questions： 如何理解 at32f4xx_clock.c 中 HEXT 的启动过程？

Answer：

在 at32f4xx_clock.c 中，函数 system_clock_config()目的是配置系统时钟。其中若直接使用 HEXT 作为系统主时钟，或间接使用 HEXT 作 PLL 来源而使用 PLL 作为系统主时钟时，就会执行以下代码：

```
#define HEXT_STABLE_DELAY                (5000u)
#define PLL_STABLE_DELAY                 (500u)
/* reset crm */
crm_reset();
crm_clock_source_enable(CRM_CLOCK_SOURCE_HEXT, TRUE);
/* wait for hext stable ,specially for AT32F403*/
wait_stbl(HEXT_STABLE_DELAY);
/* wait till hext is ready */
while(crm_hext_stable_wait() == ERROR)
{
}

/* config pll clock resource */
crm_pll_config(CRM_PLL_SOURCE_HEXT_DIV, CRM_PLL_MULT_48, CRM_PLL_OUTPUT_RANGE_GT72MHZ);
/* enable pll */
crm_clock_source_enable(CRM_CLOCK_SOURCE_PLL, TRUE);
/* wait till pll is ready */
while(crm_flag_get(CRM_PLL_STABLE_FLAG) != SET)
{
}
/* config apb2clk */
crm_apb2_div_set(CRM_APB2_DIV_2);
/* config apb1clk */
crm_apb1_div_set(CRM_APB1_DIV_2);
/* 1step: config ahbclk div8 */
crm_ahb_div_set(CRM_AHB_DIV_8);
/* select pll as system clock source */
crm_sysclk_switch(CRM_SCLK_PLL);
/* wait till pll is used as system clock source */
while(crm_sysclk_switch_status_get() != CRM_SCLK_PLL)
```

```
{
}
/* delay */
wait_stbl(PLL_STABLE_DELAY);
```

红色加粗部分 wait_stbl(HEXT_STABLE_DELAY);函数是 AT32F403 独有的，这段代码的本质是延时空指令，其他型号不需要。原因是 AT32F403 就绪标志位置起过早，此时晶振的实际振荡情况尚不稳定，若立即进行设置 PLL 或切换系统主时钟操作，会造成系统异常。HEXT_STABLE_DELAY 设定值为 5000，执行延时时间约为 2 ms。经实测若 HEXT 晶振配置合宜，就绪标志位置起后再等 2 ms 即可进行后续操作而不会引发系统异常。

crm_hext_stable_wait()函数和 HEXT 起振计时非常相关，特别说明如下：

```
#define HEXT_STARTUP_TIMEOUT        ((uint16_t)0x3000) /*!< time out for hext start up */
error_status crm_hext_stable_wait(void)
{
  uint32_t stable_cnt = 0;
  error_status status = ERROR;
  while((crm_flag_get(CRM_HEXT_STABLE_FLAG) != SET) && (stable_cnt < HEXT_STARTUP_TIMEOUT))
  {
    stable_cnt ++;
  }
  if(crm_flag_get(CRM_HEXT_STABLE_FLAG) != SET)
  {
    status = ERROR;
  }
  else
  {
    status = SUCCESS;
  }
  return status;
}
```

在使能 HEXT 后，代码会进入等待循环等待 HEXT 就绪标志位置起后跳出循环，再进行后续配置动作。但此循环运行的同时也会进行超时计数。若 HEXT 就绪标志位迟迟不起且时间超出 HEXT_STARTUP_TIMEOUT 的设定，表示 HEXT 启动失败，代码会返回 ERROR，由用户在 system_clock_config()中自行处理这个错误情况。

经实测，特性优良的 HEXT 晶振在硬件匹配合适时，约 800 μs 就可以起振就绪；若特性及匹配差一些，最多 10 ms 也可起振就绪。HEXT_STARTUP_TIMEOUT 设定值为 0x3000，在最佳时间优化的编译设置下约为 20 ms。若 HEXT 启动后等待 20 ms 就绪标志位都还未置起，可以判定晶振有匹配问题、虚焊脱落、或是损坏。因此 HEXT_STARTUP_TIMEOUT 的设定值提供一个合理的超时检测，不但可以满足一般 HEXT 起振就绪时间，又可得知 HEXT 的硬件问题。

另外，硬件电路设计需符合规格。对于无源晶振，需要检查匹配电容是否符合规格。若不符合，需要用户调整电容容值以满足要求。更多硬件电路设计要点请参考《AN0078_AT32 微控制器硬件设计指南及抗 EMC/EFT 设计要点》及相关型号 Datasheet，这些资料都可以从雅特力官网获取。

类型： MCU 应用
适用型号： AT32F4 系列
主功能： HEXT
次功能： 无

文档版本历史

日期	版本	变更
2022.3.10	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。因雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。

© 2022 雅特力科技 (重庆) 有限公司 保留所有权利