AN0121
入门指南

AT32F425时钟配置

前言

本应用入门指南主要介绍两部分内容：
1、基于雅特力提供的 V2.x.x 的板级支持包来进行时钟源码的配置及修改
2、如何使用配套的时钟配置工具来进行时钟路径及参数的设定，生成相应的时钟流程代码并使用。

支持型号列表：

支持型号	AT32F425 系列

目录

表目录

图目录

1 简介

时钟是芯片正确高效运行的基础，正确的时钟配置是芯片能正确运行的必要条件，其重要性不言而喻。AT32 各系列产品的时钟配置部分可能存在细微的差异和需要注意的事项，本文档就着重针对各系列的情况来详细介绍如何结合雅特力提供的 V2.x.x 的板级支持包（BSP）来配置时钟。

以下介绍时钟配置的方法主要分两种：

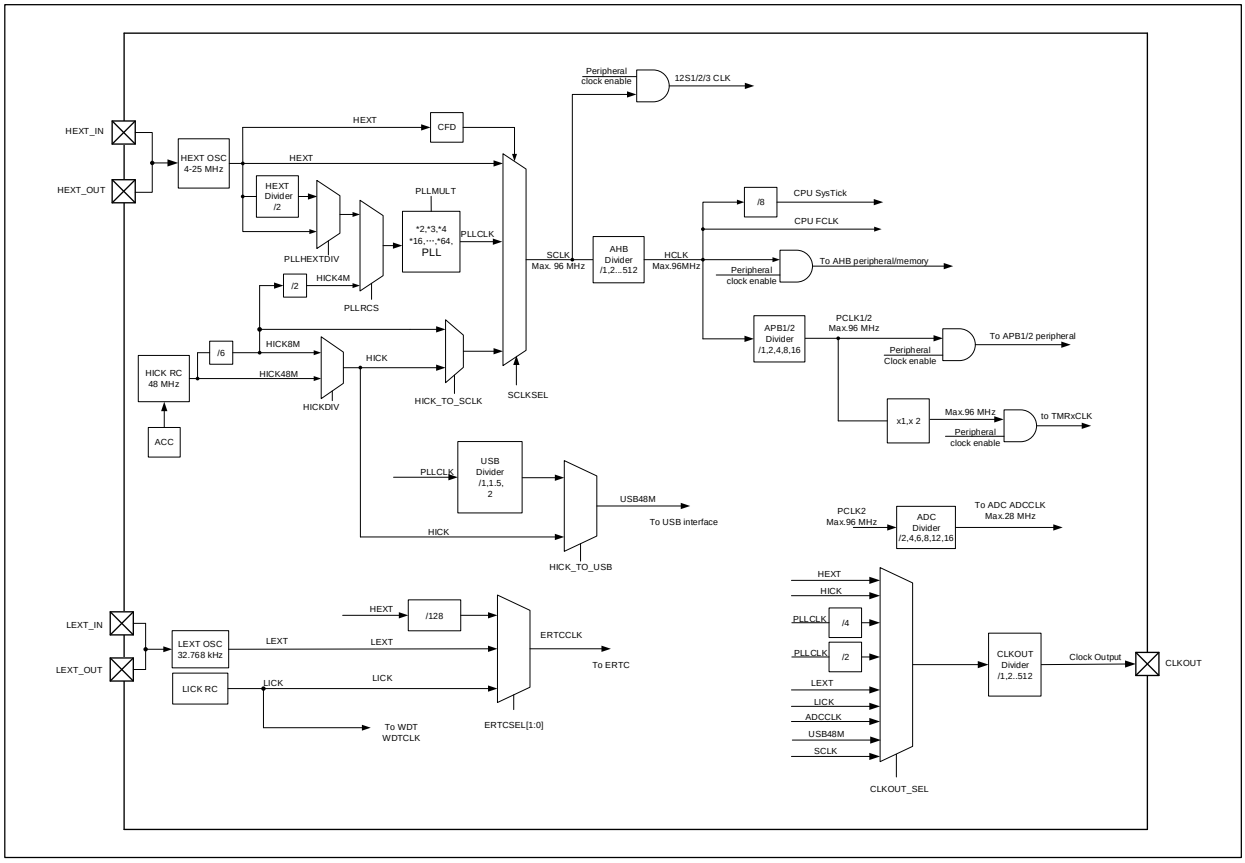
1、 以手动编写代码调用 BSP 中提供的驱动函数接口来进行时钟配置。

2、 采用时钟工具来配置并生成相应的源码文件。

2 时钟树

在进行时钟配置之前，应充分了解对应芯片的时钟树结构，这样在进行时钟配置时才会游刃有余。对于系统时钟频率及路径的配置我们需要关注时钟源、倍频及系统时钟部分。类似如下图：

图 1. 时钟框图



可由图中得到以下几个关键信息：

1) SCLKSEL：系统时钟可以由 HEXT、PLLCLK、HICK 三大时钟源提供。
2) HEXT：HEXT 是外部高速时钟，其可以外接范围是 4~25 MHz 的晶振或时钟源。
3) HICK：HICK RC 是内部高速振荡器，频率为 48 MHz。HICK 时钟由内部振荡器给出，但在初始情况下由 HICKDIV 控制并默认 6 分频后为 8 MHz，亦可配置为不分频，保持 48 MHz 的频率。
4) PLLCLK：PLL 时钟 = PLL 输入时钟 * PLL 倍频系数。
5) PLL 输入时钟：PLL 的输入时钟由 PLLRCS 及 PLLHEXTDIV 共同决定，其细分可分为三个来源：HICK 4MHz、HEXT 和 HEXTDIV，HEXTDIV 由 HEXT 时钟 2 分频。

3 代码配置解析

以下将以库函数接口为核心来对时钟配置流程和方法进行说明。

3.1 函数接口

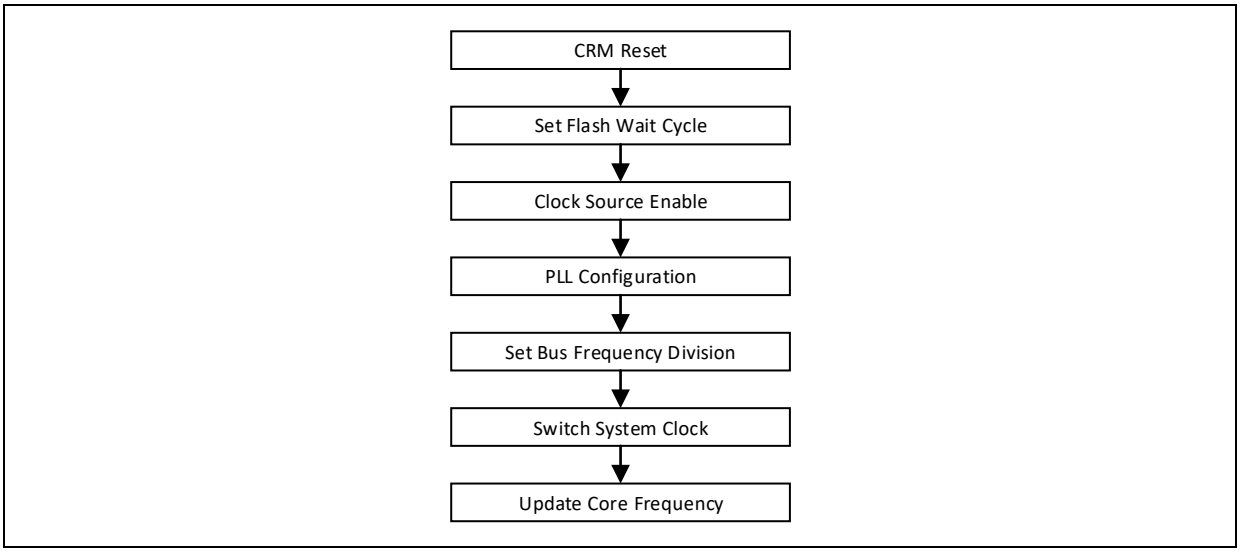
各系列产品对应提供的 BSP 中对硬件的时钟设置部分已封装好接口函数以供调用，以下罗列出时钟配置常用的函数接口，各函数的具体参数及返回值类型等请参考 at32f425_crm.c/.h 文件。

/* 时钟和复位管理模块的复位函数，将时钟配置部分恢复到默认值 */ void crm_reset(void);
/* 外部高速时钟的旁路使能函数 */ void crm_hext_bypass(confirm_state new_state);
/* 各状态标志的获取函数，如 PLL/HEXT/HICK 等时钟源的稳定标志等 */ flag_status crm_flag_get(uint32_t flag);
/* 等待外部高速时钟稳定 */ error_status crm_hext_stable_wait(void);
/* 时钟源使能函数，如 PLL/HEXT/HICK 等时钟源的使能 */ void crm_clock_source_enable(crm_clock_source_type source, confirm_state new_state);
/* PLL 配置函数，配置内容包括：PLL 时钟源、PLL 倍频系数等 */ void crm_pll_config(crm_pll_clock_source_type clock_source, crm_pll_mult_type mult_value);
/* 系统时钟切换函数 */ void crm_sysclk_switch(crm_sclk_type value);
/* 当前系统时钟切换状态获取函数 */ crm_sclk_type crm_sysclk_switch_status_get(void);
/* 内部高速时钟 6 分频配置函数，主要用于 hick 48 MHz 接入系统时钟 */ void crm_hick_divider_select(crm_hick_div_6_type value);
/* 内部高速时钟用做系统时钟时的频率路径选择函数，可设置固定路径（8 MHz），6 分频路径路径（8 MHz 或 48 MHz 由 6 分频配置函数决定） */ void crm_hick_sclk_frequency_select(crm_hick_sclk_frequency_type value);
/* 系统时钟到 AHB 时钟的分频设置函数 */ void crm_ahb_div_set(crm_ahb_div_type value);
/* AHB 时钟到 APB1 时钟的分频设置函数 */ void crm_apb1_div_set(crm_apb1_div_type value);
/* AHB 时钟到 APB2 时钟的分频设置函数 */ void crm_apb2_div_set(crm_apb2_div_type value);

3.2 时钟配置流程

按常规应用来讲解时钟配置流程，其内容可大致分为如下步骤：

图 2. 时钟配置流程图



3.2.1 复位（CRM Reset）

首先按规范流程应复位 CRM 配置参数，其主要是将系统时钟切换到 HICK，其余的系统时钟配置寄存器写入默认值，待后续进行新配置参数的写入。函数调用的代码实现如下：

```
crm_reset();                    /* CRM 复位 */
```

3.2.2 Flash 等待周期（Set Flash Wait Cycle）

AT32F425 片上采用的是嵌入式 Flash，当运行在不同的主频下时需对应设定 Flash 等待周期。flash 等待周期与运行主频关系如下：

System Clock Frequency	Flash Wait Cycle
0< sysclk <=32 MHz	FLASH_WAIT_CYCLE_0
32< sysclk <=64 MHz	FLASH_WAIT_CYCLE_1
64< sysclk <=96 MHz	FLASH_WAIT_CYCLE_2

函数调用的代码实现如下：

```
flash_psr_set(FLASH_WAIT_CYCLE_0);          /* 设置 flash 等待周期 0 cycle */
```

3.2.3 时钟源配置（Clock Source Configuration）

与系统时钟相关的高速时钟源主要包括 HEXT 和 HICK，PLL 也是使用以上时钟源来进行倍频。需要在配置使能 PLL 前将所使用的 PLL 参考时钟源开启并等待其稳定。

- HEXT

外部高速时钟如采用外接有源时钟的方式时，可开启旁路模式来进行使用，采用晶振时，不能开启旁路模式，旁路模式应在外部高速时钟源使能前进行设定，其默认情况为关闭。旁路模式使能代码实现如下：

```
crm_hext_bypass(TRUE);                          /* HEXT 时钟旁路模式开启 */
```

使能 HEXT 时钟源并等待 HEXT 时钟稳定，代码实现如下：

```
crm_clock_source_enable(CRM_CLOCK_SOURCE_HEXT, TRUE);   /* 开启 HEXT 时钟源 */
while(crm_hext_stable_wait() == ERROR)                  /* 等待 HEXT 时钟稳定 */
{
```

```
}
```

◆ HICK

内部高速时钟是由芯片内部振荡器提供，使能 HICK 时钟源并等待 HICK 时钟稳定，代码实现如下：

```
crm_clock_source_enable(CRM_CLOCK_SOURCE_HICK, TRUE);   /* 开启 HICK 时钟源 */
while(crm_flag_get(CRM_HICK_STABLE_FLAG) != SET)        /* 等待 HICK 稳定标志置起 */
{
}
```

3.2.4 PLL 配置（PLL Configuration）

PLL 配置主要包括：PLL 时钟源、PLL 倍频系数、PLL 倍频频率范围等的设置。倍频时钟公式为：PLLCLK = PLL 输入时钟 * PLL 倍频系数。

◆ PLL 时钟源

PLL 时钟源细分有三个来源：1、HICK（4 MHz），2、HEXT，3、HEXT 分频时钟，PLL 时钟源应在 PLL 配置使能前开启并等待稳定。以上 PLL 时钟源在 crm_pll_config 函数中对应的参数定义如下：

```
CRM_PLL_SOURCE_HICK
CRM_PLL_SOURCE_HEXT
CRM_PLL_SOURCE_HEXT_DIV
```

当选择 PLL 时钟源为 CRM_PLL_SOURCE_HEXT_DIV 时，HEXT 的分频系数默认为 2 分频。

◆ PLL 倍频系数

倍频系数为 2~64 倍可选，但应该注意最高主频限制，以此按实际情况来合适选择倍频系数，如 8 倍频使用参数 CRM_PLL_MULT_8。

当 PLL 参数设置完成后，即可开启 PLL 并等待 PLL 稳定。示例：外部时钟晶振 8 MHz，采用 HEXT 2 分频时钟作为 PLL 时钟源，PLLCLK 倍频到 96 MHz 的代码实现如下：

```
crm_pll_config(CRM_PLL_SOURCE_HEXT_DIV, CRM_PLL_MULT_24);   /* 配置 PLL 参数 */
crm_clock_source_enable(CRM_CLOCK_SOURCE_PLL, TRUE);        /* 开启 PLL 时钟源 */
while(crm_flag_get(CRM_PLL_STABLE_FLAG) != SET)             /* 等待 PLL 稳定标志置起 */
{
}
```

3.2.5 总线分频（Set Bus Frequency Division）

总线分频包含 SCLK 到 AHBCLK 分频、AHBCLK 到 APB1CLK 分频、AHBCLK 到 APB2CLK 分频。AHB 总线 1 分频、APB1/APB2 总线 1 分频的代码实现如下：

```
crm_ahb_div_set(CRM_AHB_DIV_1);          /* SCLK 1 分频作为 AHB 总线时钟 */
crm_apb2_div_set(CRM_APB2_DIV_1);        /* AHBCLK 1 分频作为 APB2 总线时钟 */
crm_apb1_div_set(CRM_APB1_DIV_1);        /* AHBCLK 1 分频作为 APB1 总线时钟 */
```

3.2.6 切换系统时钟（Switch System Clock）

系统时钟来源主要有三个：HICK、HEXT、PLLCLK。在切换系统时钟到如上时钟源时应提前确保对应时钟源已稳定。

◆ HICK 系统时钟

内部高速时钟在系统复位重新运行时默认作为系统时钟，后期代码进行设定时，可有两种频率值来进行设定（8 MHz 和 48 MHz）。如图 1 所述 HICK 默认情况下用的是 8 MHz，可配置为 48 MHz。

HICK 8 MHz 用作系统时钟的代码实现如下：

```
crm_sysclk_switch(CRM_SCLK_HICK);                          /* 切换系统时钟到 HICK */
while(crm_sysclk_switch_status_get() != CRM_SCLK_HICK)     /* 等待系统时钟状态为 HICK */
{
}
```

HICK 48 MHz 用作系统时钟的代码实现如下：

```
crm_hick_sclk_frequency_select (CRM_HICK_SCLK_48MHZ);      /* HICK 选择 hick48MHz */
crm_sysclk_switch(CRM_SCLK_HICK);                          /* 切换系统时钟到 HICK */
while(crm_sysclk_switch_status_get() != CRM_SCLK_HICK)     /* 等待系统时钟状态为 HICK */
{
}
```

◆ HEXT 系统时钟

外部高速时钟用作系统时钟时，其系统时钟频率以实际使用的外部时钟频率为准，范围为 4~25 MHz。HEXT 用作系统时钟的代码实现如下：

```
crm_sysclk_switch(CRM_SCLK_HEXT);                          /* 切换系统时钟到 HEXT */
while(crm_sysclk_switch_status_get() != CRM_SCLK_HEXT)     /* 等待系统时钟状态为 HEXT */
{
}
```

◆ PLLCLK 系统时钟

PLLCLK 用作系统时钟时，其系统时钟频率以实际的 PLL 倍频结果为准。其最高频率应满足芯片规格为基础。PLLCLK 用作系统时钟的代码实现如下：

```
crm_sysclk_switch(CRM_SCLK_PLL);                           /* 切换系统时钟到 PLL */
while(crm_sysclk_switch_status_get() != CRM_SCLK_PLL)      /* 等待系统时钟状态为 PLL */
{
}
```

3.2.7 更新核心频率（Update Core Frequency）

提供的 BSP 中，其代码框架内保留了一个表示系统核心频率的参数值 system_core_clock，其保存

的是 CPU 核心的运行频率值，应该在每次系统时钟配置完成后来进行更新。为的是在整个代码框架下，各外设驱动的频率配置能很快获取到当前核心运行频率值并使用。代码实现如下：

```
system_core_clock_update();                    /* 更新系统核心频率值 system_core_clock */
```

3.3 时钟配置示例

以下将以完整的时钟配置流程来进行说明，示例：由 8 MHz 外部时钟晶振作为时钟源，其 2 分频路径经 PLL 倍频到 96 MHz 并用做系统时钟，AHB 采用 1 分频，APB1/APB2 采用 1 分频。函数 system_clock_config 代码实现如下：

```
void system_clock_config(void)
{
  crm_reset();                                          /* CRM 复位 */
  flash_psr_set(FLASH_WAIT_CYCLE_2);                    /* 设置 flash 等待周期 2 cycle */
  crm_clock_source_enable(CRM_CLOCK_SOURCE_HEXT, TRUE); /* 使能 HEXT 时钟源 */
  while(crm_hext_stable_wait() == ERROR)                /* 等待 HEXT 时钟稳定 */
  {
  }
  crm_pll_config(CRM_PLL_SOURCE_HEXT_DIV, CRM_PLL_MULT_24);    /* 配置 PLL，PLL 时钟源选择 HEXT 分频路径，倍频系数 24 倍，公式：PLLCLK = 8 / 2 * 24 = 96 MHz */
  crm_clock_source_enable(CRM_CLOCK_SOURCE_PLL, TRUE);  /* 使能 PLL */
  while(crm_flag_get(CRM_PLL_STABLE_FLAG) != SET)       /* 等待 PLL 稳定 */
  {
  }
  crm_ahb_div_set(CRM_AHB_DIV_1);                       /* SCLK 1 分频作为 AHB 总线时钟 */
  crm_apb2_div_set(CRM_APB2_DIV_1);                     /* AHBCLK 1 分频作为 APB2 总线时钟 */
  crm_apb1_div_set(CRM_APB1_DIV_1);                     /* AHBCLK 1 分频作为 APB1 总线时钟 */
  crm_sysclk_switch(CRM_SCLK_PLL);                      /* 切换系统时钟到 PLL */
  while(crm_sysclk_switch_status_get() != CRM_SCLK_PLL) /* 等待系统时钟切换状态为 PLL */
  {
  }
  system_core_clock_update();                           /* 更新系统核心频率值 */
}
```

4 时钟工具

时钟配置工具是雅特力科技为方便对AT32系列MCU进行时钟配置而开发的一个图形化配置工具，其主旨是使用户清晰了解时钟路径和配置出期望的时钟频率并生成源码文件。

4.1 环境要求

- **软件要求**

需要Windows7及以上操作系统支持。

4.2 安装

- **软件安装**

本软件不需要安装，只需直接运行可执行程序AT32_New_Clock_Configuration.exe。

4.3 功能介绍

本章节将介绍此工具的基本操作，其主要的启动界面和配置界面如下所示

图 3. 启动界面

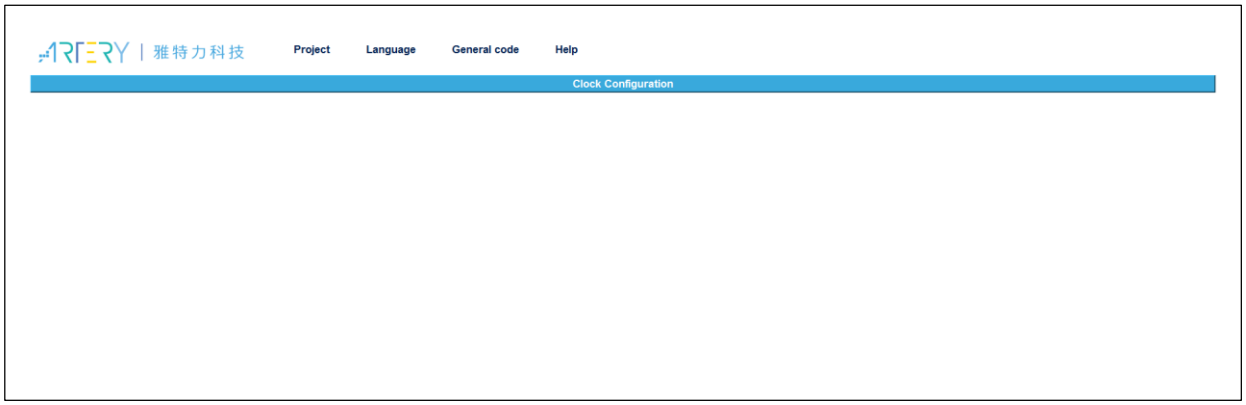
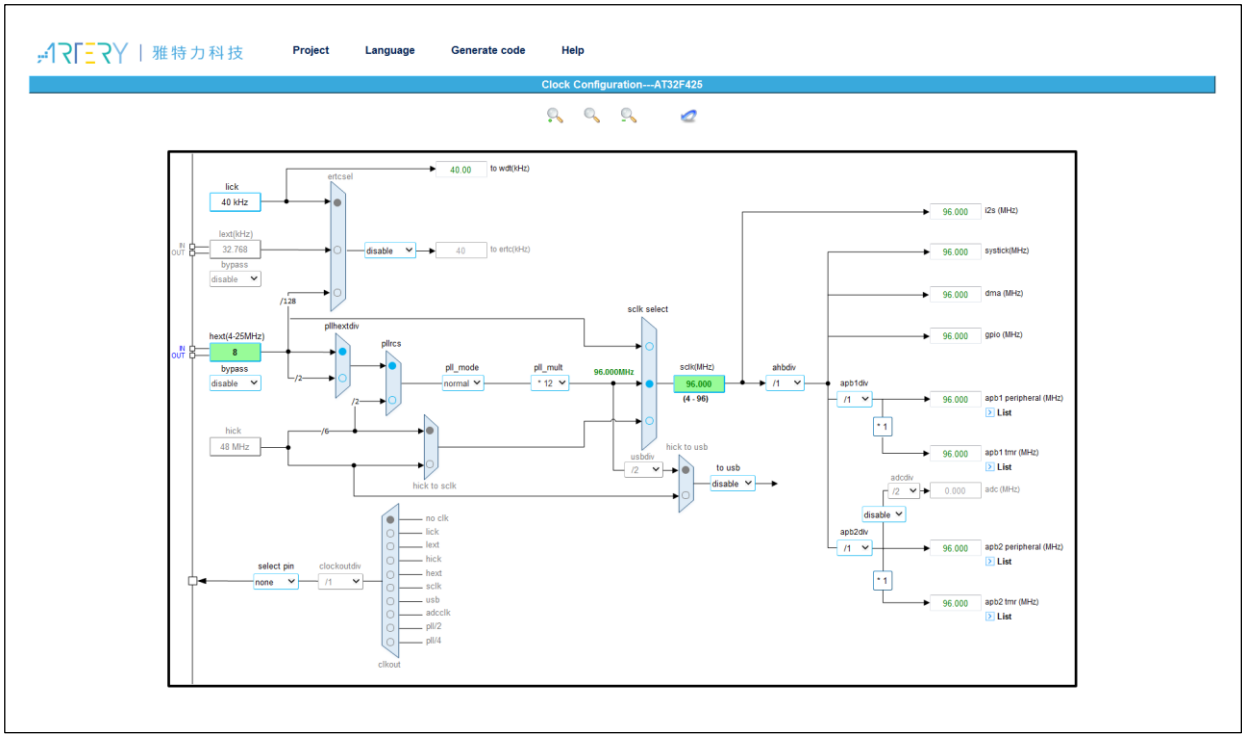


图 4. 配置界面



4.4 菜单栏

菜单栏内容如图所示：

图 5. 菜单栏

- **“项目”（Project）菜单：**
  - 新建： 新建时钟配置项目
  - 打开： 打开已存在的配置项目
  - 保存： 保存已打开的配置项目
- **“语言”（Language）菜单：**
  - English： 选择 English 作为显示语言
  - 简体中文： 选择简体中文作为显示语言
- **“生成代码”（General code）菜单：**

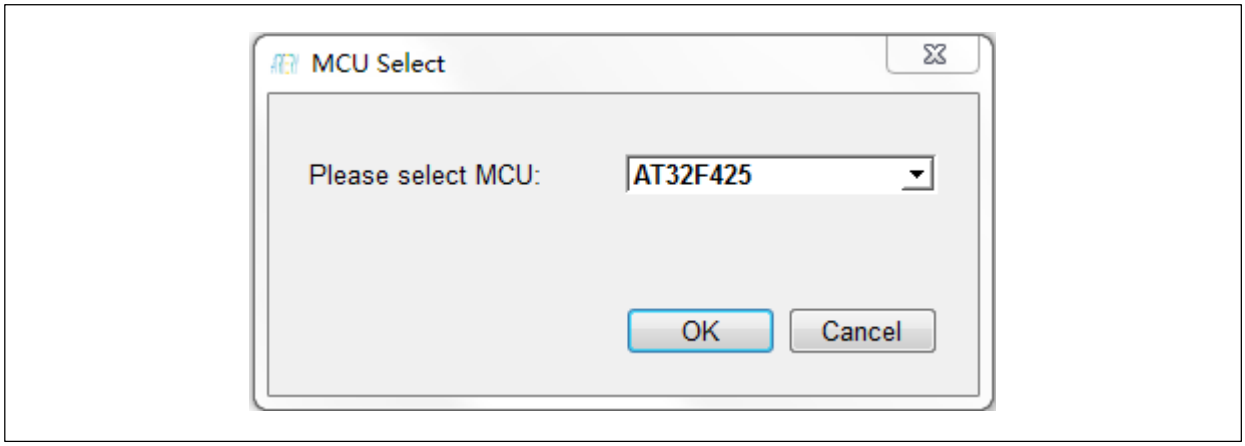
当在对应型号的操作配置界面将所期望的时钟路径和时钟频率配置完成之后，可点击“生成代码”菜单来选择源码文件的存储路径并生成相应的源码文件。

- **“帮助”（Help）菜单：**
  - 新版本下载： 联网进行新版本下载
  - 版本： 查看当前版本

4.5 新建配置项目

双击打开时钟配置工具，可看到图示的启动界面，可点击“项目”菜单-->“新建”，进行配置项目的新建，在新建配置项目的过程中需要对芯片的系列所属进行选择，操作方法如下图所示

图 6. MCU 选择界面



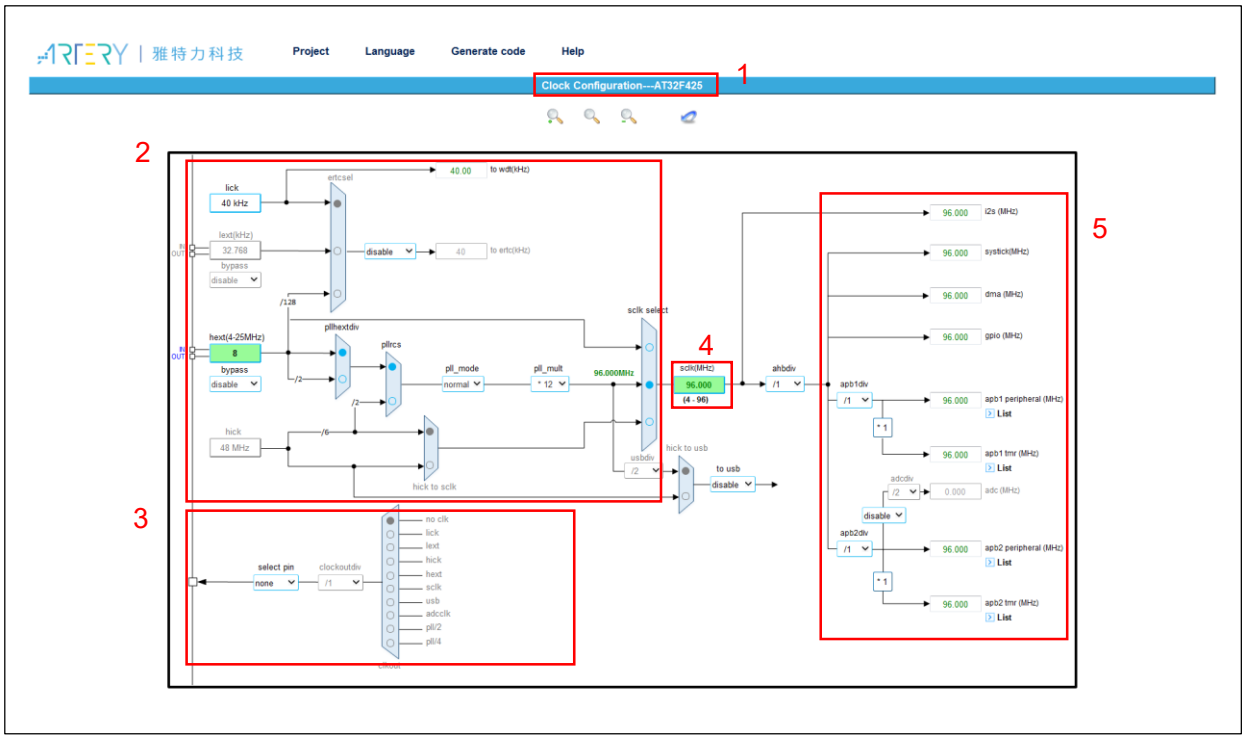
MCU 系列的选择，可点击下拉框来进行选择，当选择好 MCU 后点击“确定”可进入到时钟配置界面。

4.6 配置界面的使用

配置界面主要用来进行时钟路径及参数的配置，以下的介绍将以 AT32F425 系列作为示例来展开进行，其余系列的配置方法与此类似。

整个配置界面主要可以分为四个大块，如下图所示

图 7. 配置界面框架

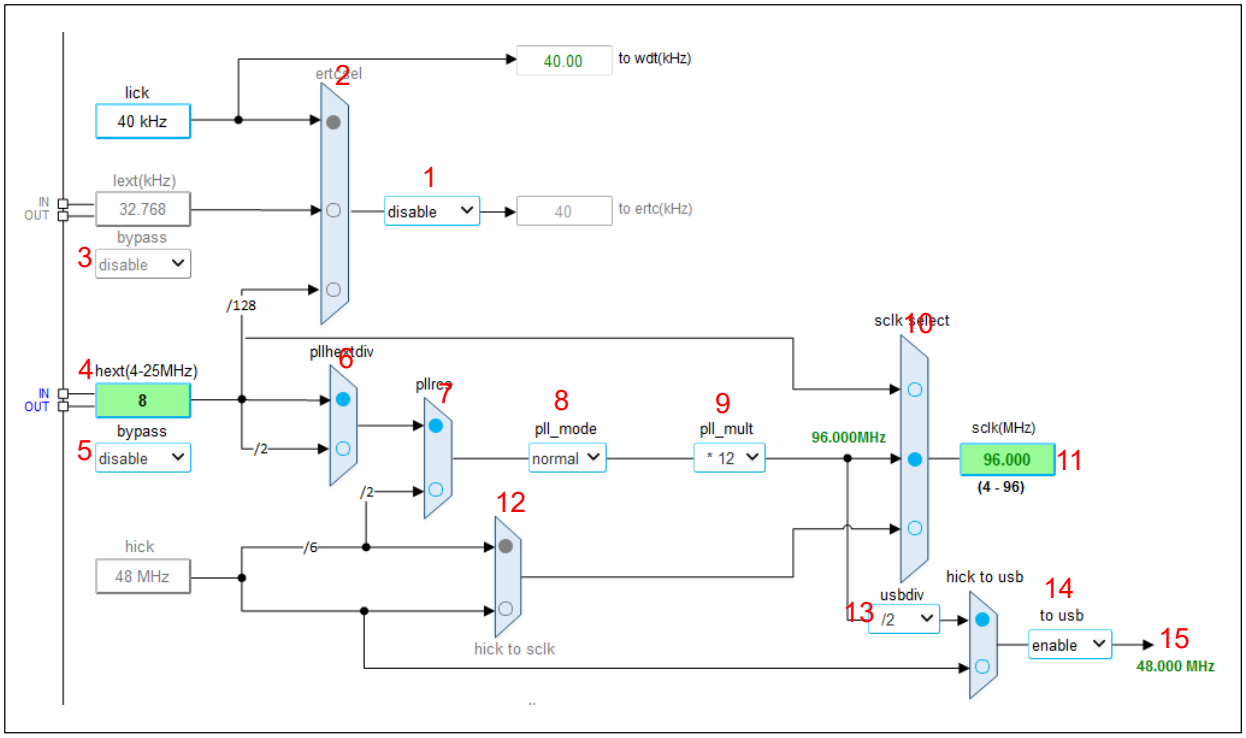


1. 标题部分：用于展示当前配置项目所选择的 MCU 系列。
2. 配置部分：用于对时钟路径和时钟参数进行选择和配置，以达到期望的应用需求。
3. 输出部分：用于时钟输出（CLKOUT）的配置。
4. 在 SCLK 栏也可在选中 PLL 为系统时钟时作为输入框，可输入期望的系统时钟频率来反向自动配置出倍频参数。
5. 结果部分：用于显示当前外设所使用的时钟频率及总线上的外设。

接下来就着重介绍一下配置部分的使用。配置部分的流程界面是对应着 MCU 时钟树来进行的，各系

列 MCU 的此部分可能存在着差异，但使用方式大同小异。时钟路径的配置可按流程对各开关进行点选来进行选择，配置部分如下图所示，将逐个流程点的功能及其注意事项进行介绍。

图 8. 时钟配置框



1. ertc 使能：ertc 时钟代码配置的使能下拉框。
2. ertcsel：点选框，ertc 时钟源选择。当 ertc 使能开启后，此点选框可配置。
3. lext bypass：外部低速时钟的旁路使能。
4. hext：此为输入框，8 MHz 为所采用外部时钟源的默认频率，用户可根据实际使用的外部时钟源频率进行修改。（注：此 8 MHz 被修改为其他频率值时，对应的 BSP 中 demo 目录下的 inc/at32f415_conf.h 文件内的 HEXT_VALUE 宏定义也应该一致修改，也可以采用工具生成的 at32f415_conf.h 文件来进行使用）。
5. hext bypass：高速外部时钟的旁路使能。
6. pllhextdiv：点选框，当 HEXT 作为 PLL 时钟源时，可配置输入频率为 HEXT 分频或 HEXT 不分频。
7. pllrcs：点选框，可配置 PLL 时钟源为 HEXT 或 HICK。
8. pll_mode：下拉框，可选择 PLL 的配置模式（normal 或 flexible）
9. 倍频系数：选择 normal 模式时使用 PLL_MULT 参数进行倍频，计算公式为：PLLCLK = PLL 输入时钟 * PLL_MULT，选择 flexible 模式时使用 PLL_MS、PLL_NS 和 PLL_FR 参数进行倍频，计算公式为：PLLCLK = PLL 输入时钟 / PLL_MS * PLL_NS / PLL_FR。为了用户的使用方便，在选定 PLL 输入时钟源后，结果部分的 sclk 框中输入目标时钟并按下键盘“Enter”键，会自动计算一组倍频参数以满足用户期望或相近的时钟频率 sclk select：点选框，可配置 HEXT、PLL 或 HICK 作为系统时钟。
10. sclk select：点选框，可配置 HEXT、PLL 或 HICK 作为系统时钟。
11. sclk 频率：当采用正向配置时，此作为系统时钟频率的配置结果显示，当将其用作输入框时，输入期望的频率后点击回车键，会根据此输入值反向计算一组合适的或最接近期望值的 PLL 配置参数。
12. hick to sclk：点选框，当 sclk select 选择 HICK 作为系统时钟时，可配置 HICK 的 8 MHz 或 48MHz 到系统时钟（注：当选择 48 MHz HICK 到系统时钟后，CLKOUT 输出 HICK 时的频率也

为 48 MHz）。

13. usbdiv：下拉框。当 PLL 时钟被选作为 USB 的时钟来源时，此处配置 PLL 时钟到 USB 时钟的分频系数。
14. USB 使能：USB 时钟代码配置的使能下拉框。
15. USB 时钟频率的显示。此显示栏会实时计算 USB 时钟的频率并显示，如果配置出来的 USB 时钟不等于 48 MHz 时，显示出来的 USB 时钟频率会标注为红色，而实际应用中没有用到 USB 时选择 disable 则不会显示。（注：此部分只针对 USB 时钟频率的配置，USB 外设时钟使能需自行额外打开）

4.7 生成代码

当时钟配置完成后，可点击生成代码，然后选择代码生成的路径并确认，最后会在所选目录下生成两个文件夹 inc 和 src，源文件存放在 src 文件夹下，头文件存放在 inc 文件夹下。这些文件可结合到 BSP_V2.x.x 内的工程来进行使用。可以采用新生成的时钟代码文件（at32f4xx_clock.c/ at32f4xx_clock.h/ at32f4xx_conf.h）将原 BSP demo 中的对应文件替换，在 main 函数中进行 system_clock_config 函数调用即可。

5 注意事项

5.1 外部时钟源(HEXT)修改

因本文档所示例的 demo 和配置工具都默认采用的 8 MHz 外部时钟频率，当实际硬件使用的外部时钟源是非 8 MHz 频率时需注意以下几点。

◆ 代码修改

1、 以实际的外部时钟频率按文中时钟配置流程章节所描述的时钟配置流程及方法来编写相应的代码，配置出期望的时钟配置及时钟路径。

2、 修改对应 demo 工程中 at32f4xx_conf.h 文件的 HEXT_VALUE 值，以实际使用的外部时钟源频率值来进行修改。如实际外部高速时钟使用 12.288 MHz 的晶振或时钟源时，at32f4xx_conf.h 文件应修改如下：

```
#if !defined  HEXT_VALUE
#define HEXT_VALUE                ((uint32_t)12288000)
#endif
```

◆ 工具修改

1、 在时钟配置工具中的 HEXT 输入框内填入外部时钟源实际频率值并按“Enter”键确认。

2、 配置好所需的时钟路径及时钟频率，生成代码。采用新生成的时钟代码文件（at32f4xx_clock.c/ at32f4xx_clock.h/ at32f4xx_conf.h）将原 BSP demo 中的对应文件替换或取其中函数内容进行替换，在 main 函数中进行 system_clock_config 函数调用即可。

5.2 工具使用

在使用本时钟配置工具时需注意：

1. 此工具生成的时钟配置源码文件需结合雅特力科技提供的 BSP_V2.x.x 进行使用。
2. 不同系列所生成的时钟配置源码文件不能型号混用，只能在相对应的工程项目中进行调用。
3. 配置工具中各输入框参数修改后，请以“Enter”键结束。

6 案例 系统时钟切换

6.1 功能简介

在系统运行过程中来进行系统时钟切换。

6.2 资源准备

1) 硬件环境:

对应产品型号的 AT-START BOARD

2) 软件环境

project\at_start_f425\examples\crm\sysclk_switch

6.3 软件设计

1) 配置流程

- 初始化按键。
- 配置 clkout 时钟输出 pll 4 分频。
- 编写从 hick 经 pll 倍频 64 MHz 到系统时钟的配置代码。
- 编写从 hext 2 分频经 pll 倍频 96 MHz 到系统时钟的配置代码。

2) 代码介绍

main 函数代码描述

```
int main(void)
{
  system_clock_config();                          /* 系统时钟配置，默认 96 MHz */
  at32_board_init();                              /* 初始化按键和 led */
  clkout_config();                                /* clkout 配置输出 pll 4 分频 */
  while(1)
  {
    if(at32_button_press() == USER_BUTTON)        /* 检测是否按键按下 */
    {
      switch_system_clock();                      /* 64 与 96 MHz 系统时钟交替切换 */
      at32_led_toggle(LED4);                      /* 切换一次，led4 toggle 一次 */
    }
    at32_led_toggle(LED2);                        /* led2 作为运行状态指示灯 */
    delay_ms(100);                                /* 延时 100 ms */
  }
}
```

hick 经 pll 倍频 64 MHz 到系统时钟的代码描述

```
static void sclk_64m_hick_config(void)
{
  crm_reset();                                    /* CRM 复位 */
  flash_psr_set(FLASH_WAIT_CYCLE_1);              /* 设置 flash 等待周期 1 cycle */
```

```
  crm_clock_source_enable(CRM_CLOCK_SOURCE_HICK, TRUE);      /* 使能 HICK 时钟源 */
  while(crm_flag_get(CRM_HICK_STABLE_FLAG) != SET)           /* 等待 HICK 稳定标志置起 */
  {
  }
  crm_pll_config(CRM_PLL_SOURCE_HICK, CRM_PLL_MULT_16);  /* 配置 PLL，PLL 时钟源选择
HICK，倍频系数 16 倍，公式：PLLCLK = 8 / 2 * 16 = 64 MHz */
  crm_clock_source_enable(CRM_CLOCK_SOURCE_PLL, TRUE);    /* 使能 PLL */
  while(crm_flag_get(CRM_PLL_STABLE_FLAG) != SET)          /* 等待 PLL 稳定 */
  {
  }
  crm_ahb_div_set(CRM_AHB_DIV_1);                    /* SCLK 1 分频作为 AHB 总线时钟 */
  crm_apb2_div_set(CRM_APB2_DIV_1);                  /* AHBCLK 1 分频作为 APB2 总线时钟 */
  crm_apb1_div_set(CRM_APB1_DIV_1);                  /* AHBCLK 1 分频作为 APB1 总线时钟 */
  crm_sysclk_switch(CRM_SCLK_PLL);                       /* 切换系统时钟到 PLL */
  while(crm_sysclk_switch_status_get() != CRM_SCLK_PLL)      /* 等待系统时钟切换状态为 PLL */
  {
  }
  system_core_clock_update();                        /* 更新系统核心频率值 */
  delay_init();                                      /* 因系统时钟频率切换，重新初始化 delay */
  clkout_config();                                   /* 因 crm 复位，重新配置 clkout */
}
```

hext 2 分频经 pll 倍频 96 MHz 到系统时钟的代码描述

```
static void sclk_96m_hext_config(void)
{
  crm_reset();                                           /* CRM 复位 */
  flash_psr_set(FLASH_WAIT_CYCLE_2);                     /* 设置 flash 等待周期 2 cycle */
  crm_clock_source_enable(CRM_CLOCK_SOURCE_HEXT, TRUE);  /* 使能 HEXT 时钟源 */
  while(crm_hext_stable_wait() == ERROR)                 /* 等待 HEXT 时钟稳定 */
  {
  }
  crm_pll_config(CRM_PLL_SOURCE_HEXT_DIV, CRM_PLL_MULT_24); /* 配置 PLL，PLL 时钟源选择
HEXT 分频路径，倍频系数 24 倍，公式：PLLCLK = 8 / 2 * 24 = 96 MHz */
  crm_clock_source_enable(CRM_CLOCK_SOURCE_PLL, TRUE);   /* 使能 PLL */
  while(crm_flag_get(CRM_PLL_STABLE_FLAG) != SET)            /* 等待 PLL 稳定 */
  {
  }
  crm_ahb_div_set(CRM_AHB_DIV_1);                    /* SCLK 1 分频作为 AHB 总线时钟 */
```

```
  crm_apb2_div_set(CRM_APB2_DIV_1);                    /* AHBCLK 1 分频作为 APB2 总线时钟 */
  crm_apb1_div_set(CRM_APB1_DIV_1);                    /* AHBCLK 1 分频作为 APB1 总线时钟 */
  crm_sysclk_switch(CRM_SCLK_PLL);                        /* 切换系统时钟到 PLL */
  while(crm_sysclk_switch_status_get() != CRM_SCLK_PLL)    /* 等待系统时钟切换状态为 PLL */
  {
  }
  system_core_clock_update();                          /* 更新系统核心频率值 */
  delay_init();                                        /* 因系统时钟频率切换，重新初始化 delay */
  clkout_config();                                     /* 因 crm 复位，重新配置 clkout */
}
```

6.4 实验效果

- 上电运行 led2 以间隔 100ms 时间进行闪烁，clkout（PA8）输出 24 MHz。
- 每次 USER 按键按下，系统时钟在 64 MHz 与 96 MHz 之间进行交替切换，clkout 输出对应的 4 分频频率，led4 toggle 一次。

7 案例 时钟失效检测

7.1 功能简介

在当 HEXT 时钟直接或间接作为系统时钟时，当 HEXT 时钟出现故障，且时钟失效模块侦测到失效后，时钟失效事件将产生 NMI 中断，在此中断中可完成系统的营救操作。

7.2 资源准备

1) 硬件环境:

对应产品型号的 AT-START BOARD

2) 软件环境

project\at_start_f425\examples\crm\clock_failure_detection

7.3 软件设计

1) 配置流程

- 配置 clkout 时钟输出 pll 4 分频。
- 开启时钟失效检测，并完善 void NMI_Handler(void)函数。
- 编写从 hick 经 pll 倍频 96 MHz 到系统时钟的配置代码。

2) 代码介绍

main 函数代码描述

```
int main(void)
{
  system_clock_config();                          /* 配置系统时钟 */
  at32_board_init();                              /* 初始化 led 和 delay 函数 */
  clkout_config();                                /* 配置 clkout 输出 pll 4 分频 */
  crm_clock_failure_detection_enable(TRUE);       /* 开启时钟失效检测 */
  while(1)
  {
    at32_led_toggle(LED2);                        /* led2 作为运行状态指示灯 */
    delay_ms(200);                                /* 延时 200 ms */
  }
}
```

hick 经 pll 倍频 96 MHz 到系统时钟的代码描述

```
static void sclk_96m_hick_config(void)
{
  crm_reset();                                                /* CRM 复位 */
  flash_psr_set(FLASH_WAIT_CYCLE_2);                          /* 设置 flash 等待周期 2 cycle */
  crm_clock_source_enable(CRM_CLOCK_SOURCE_HICK, TRUE);       /* 使能 HICK 时钟源 */
  while(crm_flag_get(CRM_HICK_STABLE_FLAG) != SET)            /* 等待 HICK 稳定标志置起 */
  {
  }
```

```
  crm_pll_config(CRM_PLL_SOURCE_HICK, CRM_PLL_MULT_24);      /* 配置 PLL，PLL 时钟源选择 HICK，倍频系数 24 倍，公式：PLLCLK = 8 / 2 * 24 = 96 MHz */
  crm_clock_source_enable(CRM_CLOCK_SOURCE_PLL, TRUE);     /* 使能 PLL */
  while(crm_flag_get(CRM_PLL_STABLE_FLAG) != SET)                /* 等待 PLL 稳定 */
  {
  }
  crm_ahb_div_set(CRM_AHB_DIV_1);                          /* SCLK 1 分频作为 AHB 总线时钟 */
  crm_apb2_div_set(CRM_APB2_DIV_1);                        /* AHBCLK 1 分频作为 APB2 总线时钟 */
  crm_apb1_div_set(CRM_APB1_DIV_1);                        /* AHBCLK 1 分频作为 APB1 总线时钟 */
  crm_sysclk_switch(CRM_SCLK_PLL);                                 /* 切换系统时钟到 PLL */
  while(crm_sysclk_switch_status_get() != CRM_SCLK_PLL)          /* 等待系统时钟切换状态为 PLL */
  {
  }
  system_core_clock_update();                              /* 更新系统核心频率值 */
  delay_init();                                            /* 因系统时钟频率切换，重新初始化 delay */
  clkout_config();                                        /* 因 crm 复位，重新配置 clkout */
}
```

NMI 中断实现

```
void NMI_Handler(void)
{
  clock_failure_detection_handler();
}

void clock_failure_detection_handler(void)
{
  if(crm_flag_get(CRM_CLOCK_FAILURE_INT_FLAG) != RESET)   /* 判断时钟失效标志 */
  {
    crm_clock_failure_detection_enable(FALSE);              /* 关闭时钟失效检测 */
    sclk_96m_hick_config();                                /* 系统时钟拯救，由 hick 倍频到 96 MHz */
    crm_flag_clear(CRM_CLOCK_FAILURE_INT_FLAG);    /* 清除时钟失效标志 */
  }
}
```

7.4 实验效果

- 在运行过程中将晶振拔掉或晶振脚接地，产生时钟失效。通常来说 hext 比 hick 更稳定，可观测 clkout（PA8）的输出，可发现时钟拯救回来后 hick 作为源时的频率上存在细微波动。

8 文档版本历史

表 1. 文档版本历史

日期	版本	变更
2022.01.20	2.0.0	最初版本
2022.06.10	2.0.1	更新部分时钟工具截图及操作步骤描述
2022.12.23	2.0.2	调整时钟配置流程

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，例如：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）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损害的应用。如果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，风险及法律责任仍将由采购商单独承担，且采购商应独力负责在前述应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。