文档版本	V1.0.0
发布日期	20221026

APT32F110x 基于 CSI 库 SYSCON 应用指南

目录

1 概述

本文介绍了在APT32F110x中SYSCON系统控制器。

2. 适用的硬件

该例程使用于 APT32F110x 系列

3. 应用方案代码说明

基于 APT32F110x CSI 库文件系统，进行配置 SYSYCON

3.1 SYSCON 配置

- **硬件配置**

SYSCON 模块可以管理和配置系统的时钟以及和系统工作相关的功能模块，比如看门狗设置、低电压报警和复位、RESET 历史记录。

在芯片上电的初始化时，系统将自动选择 IMOSC 最高频率作为缺省工作时钟，完成上电复位和硬件初始化后，可以通过软件设置到希望的时钟源。

需要注意高速时钟（**>16MHZ**）下工作，Flash 读取速度匹配问题，必须要设置合适的 Flash wait 节拍以匹配 CPU 速度。

	WAIT	SPEED
24MHz < SYSCLK ≤ 48MHz	2	1
16MHz < SYSCLK ≤ 24MHz	1	1
SYSCLK ≤ 16MHz	0	0

图 3.1.1 时钟等待

系统支持将内部时钟通过外部管脚（CLO）输出。

- **时钟结构示意图：**

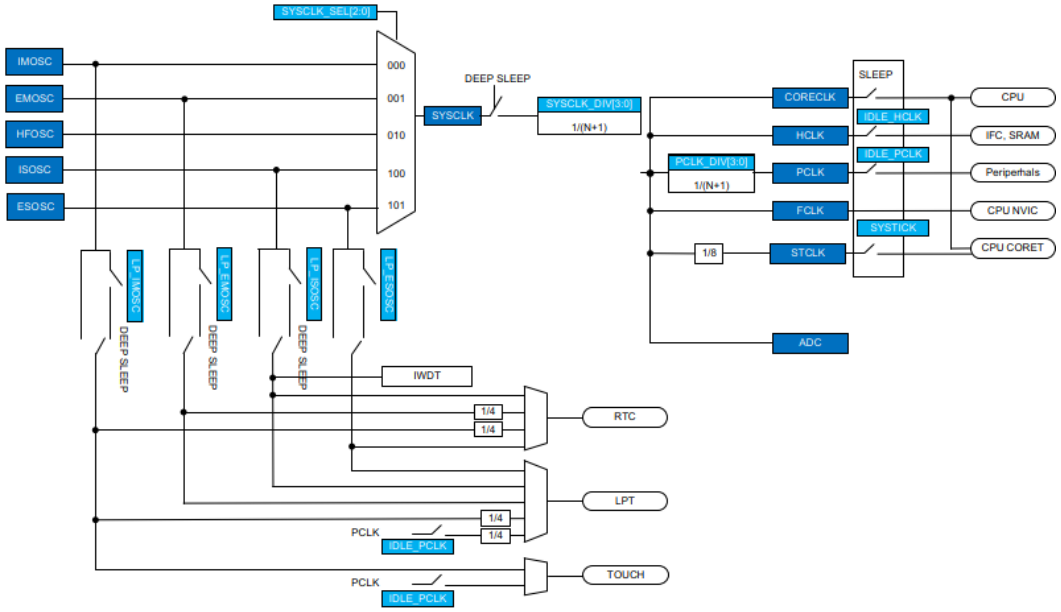


图 3.1.2 功能框图

内部主振时钟 **IMCLK**: Internal Main Clock：131.072KHz / 2.097MHz / 4.194MHz / 5.556MHz (default) (1%偏差@典型值)

内部高速振荡时钟 **HFCLK**: High Frequency Clock： 24MHz/48MHz (1%偏差@典型值)

外部晶振时钟 **EMCLK**: External Main Clock：400KHz 到 24MHz ， 支持独立的 32.768K 配置

内部辅振 **ISCLK**: Internal Sub Clock： 27KHz (5%偏差@典型值)

- **软件配置：**

可在 sdk_110x -> apt32f110x_evb -> src -> board_config.c 文件中对时钟进行修改。

```
csi_clk_config_t tClkConfig =
    {SRC_HFOSC, HFOSC_48M_VALUE, SCLK_DIV1, PCLK_DIV1, 5556000, 5556000};
    //{SRC_EMOSC, 20000000, SCLK_DIV1, PCLK_DIV2, 5556000, 5556000};
    //{SRC_IMOSC, IMOSC_5M_VALUE, SCLK_DIV1, PCLK_DIV2,5556000, 5556000};
```

```
    //{SRC_HFOSC, HFOSC_48M_VALUE, SCLK_DIV1, PCLK_DIV1,5556000, 5556000};
    //{SRC_IMOSC, IMOSC_4M_VALUE, SCLK_DIV1, PCLK_DIV1,5556000, 5556000};
    //{SRC_IMOSC, IMOSC_2M_VALUE, SCLK_DIV1, PCLK_DIV1,5556000, 5556000};
    //{SRC_IMOSC, IMOSC_131K_VALUE, SCLK_DIV1, PCLK_DIV1,5556000, 5556000};
    //{SRC_ESOSC, ESOSC_VALUE, SCLK_DIV1, PCLK_DIV1,5556000, 5556000};
```

- **函数参数说明：**

  csi_clk_config_t tClkConfig = {SRC_HFOSC, HFOSC_48M_VALUE, SCLK_DIV1, PCLK_DIV1, 5556000, 5556000};

  SRC_HFOSC ----选择时钟源

  HFOSC_48M_VALUE ----时钟源频率

  SCLK_DIV1 ----SCLK 分频值

  PCLK_DIV1 ----PCLK 分频值

  5556000 ----缺省工作时钟频率

3.2 HFOSC 做系统时钟

HFOSC 有 48M/24M/12M/6M 可选 ，这里设为 48MHZ, HCLK 分频设置为 1，PCLK 分频设置为 1。

```
    csi_clk_config_t tClkConfig =
      {SRC_HFOSC, HFOSC_48M_VALUE, SCLK_DIV1, PCLK_DIV1, 5556000, 5556000};
```

```
int main()
{
      mdelay(3000);
      //
      system_init();
      board_init();
      //
    csi_pin_set_mux(PA05,PA05_CLO);
    csi_clo_config(CLO_HFCLK,CLO_DIV8);
        while(1)
        {
        }
  }
```

```
__attribute__((weak)) void system_init(void)
{
    CK_CPU_DISALLNORMALIRQ;

    csi_iwdt_close();
    csi_sysclk_config();
    csi_get_sclk_freq();
    csi_get_pclk_freq();
    csi_tick_init();
    csi_clk_calib();

    CK_CPU_ENALLNORMALIRQ;
}
```

- 函数说明:

  CK_CPU_DISALLNORMALIRQ; ----关闭总中断

  CK_CPU_ENALLNORMALIRQ; ----开启总中断

  csi_iwdt_close(); ----关闭看门狗

  csi_sysclk_config(); ----设置时钟

  csi_get_sclk_freq(); ----获取系统时钟

  csi_get_pclk_freq(); ----获取外设时钟

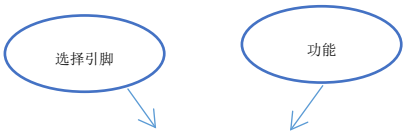
  csi_tick_init(); ----配置 cort 定时器

  csi_clk_calib(); ----校准频率

  csi_clo_config(); ----配置 CLO 输出

- 函数说明:

csi_clo_config(CLO_HFCLK,CLO_DIV8);



csi_pin_set_mux(PA05,PA05_CLO);

- **验证方法：**可以通过 CLO（PA0.5）输出系统时钟，但输出高频（>10MHZ）时需要设置分频.
- **波形图**:

CLO 选择 8 分频输出波形 48MHZ/8 = 6MHZ.

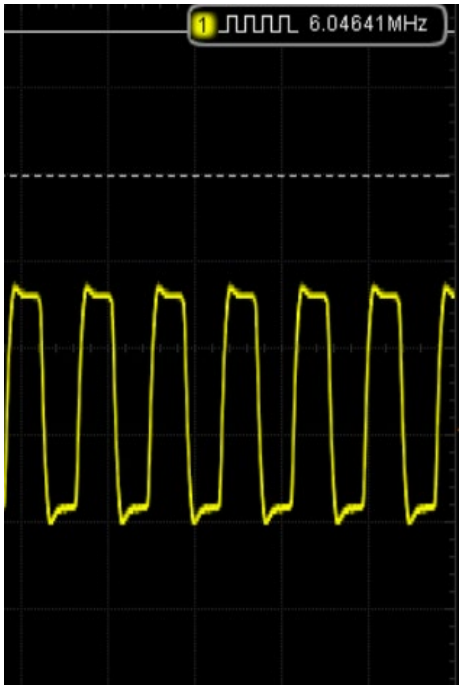


图 3.2.1 HFOSC 时钟

3.3 IMOSC 做系统时钟

IMOSC 有 5.556M/4.194M/2.097M/131K 可选，这里选择 5.556MHZ。

```
    csi_clk_config_t tClkConfig =
    {SRC_IMOSC, IMOSC_5M_VALUE, SCLK_DIV1, PCLK_DIV1,5556000, 5556000};
int main()
{
        mdelay(3000);
        //
        system_init();
        board_init();
        //
        csi_pin_set_mux(PA05,PA05_CLO);
        csi_clo_config(CLO_IMCLK,CLO_DIV1);
    while(1)
```

```
{
}
}
```

- **CLO 输出波形验证：**

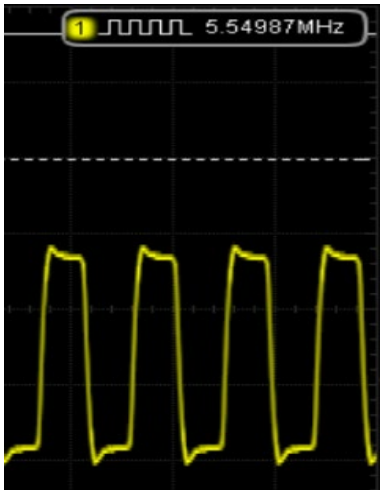


图 3.3.1 IMOSC 时钟

3.4 EMOS 做系统时钟

选择外部晶振 EMOSC，作为系统时钟。有 24M/16M/12M/8M/4M/32.768K 可选，这里选择外部 24MHZ.

- **硬件配置：**

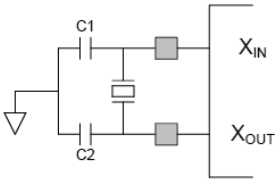
外接晶振（普通模式）	–		0.4	–	24	MHz

图 3.4.1 引脚配置

C1/C2 电容范围 20-30pF.

- **外接晶振引脚：**

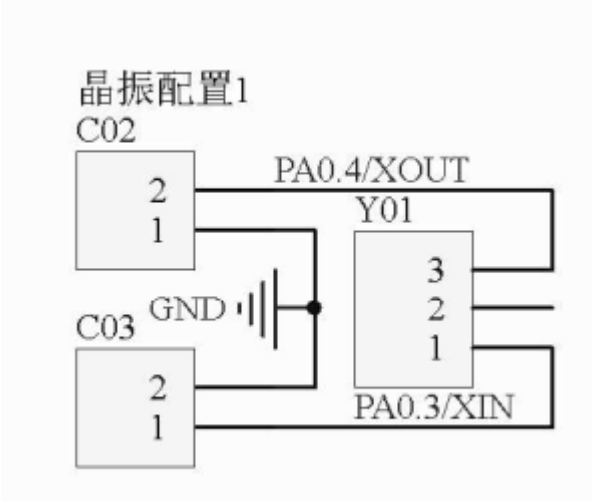


图 3.4.2 硬件配置

```
    csi_clk_config_t tClkConfig =
    {SRC_EMOSC, 24000000, SCLK_DIV1, PCLK_DIV1, 5556000, 5556000};
int main()
{
        mdelay(3000);
        //
        csi_pin_set_mux(PA03, PA03_OSC_XI);
        csi_pin_set_mux(PA04, PA04_OSC_XO);
        //
        system_init();
        board_init();
        //
        csi_pin_set_mux(PA05,PA05_CLO);
        csi_clo_config(CLO_EMCLK,CLO_DIV8);
    while(1)
    {
    }
}
```

- **波形图**:　　CLO 选择 8 分频输出波形 24MHZ/8 = 3MHZ.

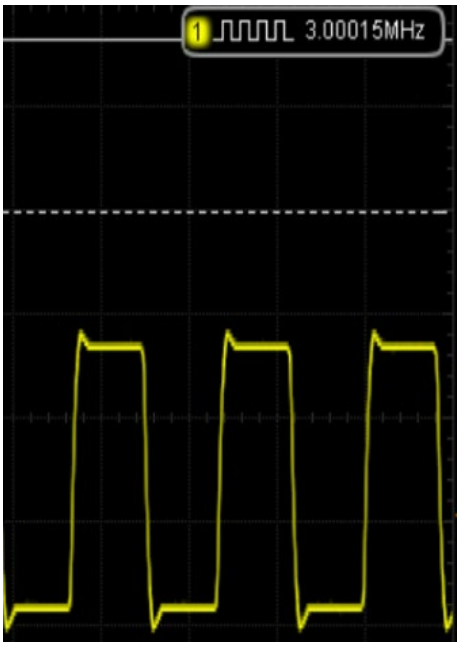


图 3.4.3 EMOS 时钟

3.5 IWDT 配置

```
csi_iwdt_init(IWDT_TO_1024);
csi_iwdt_open();
//
csi_iwdt_feed();

csi_iwdt_irq_enable(IWDT_ALARMTO_2_8, ENABLE);
```

IWDT 设置为 1s，则报警中断时间为 1*(2/8)=0.25S，用户需要在 IWDT 设置时间内进行清狗操作，若达到报警时间未清狗，在 IWDT 中断开启时会产生 IWDT 报警中断，报警中断产生后程序将进入中断程序,报警中断后需要进行喂狗操作，否则会产生 IWDT 复位。(在启用 IWDT 后需要及时清狗，避免由不及时清狗产生复位造成影响)

- **函数参数说明：**

溢出时间

csi_iwdt_init(IWDT_TO_1024);

csi_iwdt_open(); ----IWDT 使能

csi_iwdt_feed(); ----喂狗函数

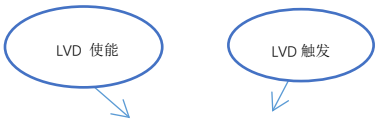
csi_iwdt_irq_enable(IWDT_ALARMTO_2_8, ENABLE); ----使能看门狗中断

3.7 LVD/LVR 配置

```
csi_lvd_int_enable(LVD_INTF,LVD_30);
byLevel = csi_get_lvdlevel();

my_printf("lvd level: %d\n", byLevel);
if(csi_lvd_flag())
{
}
else
{
}
csi_lvr_enable(LVR_31);
```

- **函数说明：**



csi_lvd_int_enable(LVD_INTF,LVD_30);

csi_get_lvdlevel(); ----用于获取 LVD 状态

csi_lvr_enable(LVR_31); ----用于开启 LVR

LVD 电压等级为 2.1/2.4/2.7/3.0/3.3/3.6/3.9V/ EXTIN_1.0V

LVR 电压等级为 1.9/2.2/2.5/2.8/3.1/3.4/3.7/4.0V

4. 程序下载和运行

1．将目标板与仿真器连接，分别为 VDD SCLK SWIO GND

2. 程序编译后仿真运行

3. 通过示波器查看，图 3.2.1、图 3.3.1、图 3.4.3 所示波形