



Melis4.0 RTOS 配置文件开发指南

版本号: 0.1
发布日期: 2021.05.12

版本历史

版本号	日期	制/修订人	内容描述
0.1	2021.05.12	PDC-PD1	创建



目 录

1 前言	1
1.1 编写目的	1
1.2 使用范围	1
1.3 相关人员	1
2 sys_config_xxx.fex 文件的使用及参数获取接口	2
2.1 概述	2
2.2 函数接口	3
2.2.1 esCFG_GetSecCount	3
2.2.2 esCFG_GetSecKeyCount	3
2.2.3 esCFG_GetKeyValue	3
2.2.4 esCFG_GetGPIOSecKeyCount	4
2.2.5 esCFG_GetGPIOSecData	4



1 前言

1.1 编写目的

介绍 melis 4.0 的配置文件的使用方法。

1.2 使用范围

Allwinner 软件平台 Melis4.0 以上版本。

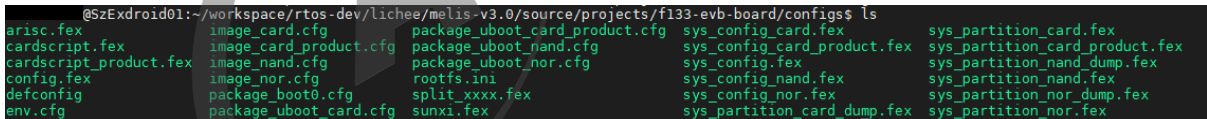
1.3 相关人员

使用 Melis4.0 以上版本进行产品开发的开发者。

2 sys_config_xxx.fex 文件的使用及参数获取接口

2.1 概述

- 功能：**sys_config_xxx.fex 文件是用来对方案进行配置的文本文件，xxx 是存储方案的标识，例如 sys_config_nor.fex、sys_config_card.fex、sys_config_nand.fex 依次对应着 nor-flash、sdcard、nand 存储方案的配置文件；
- 路径：**sys_config_xxx.fex 文件在 source/project/方案/configs/文件夹下；
- 转换：**sys_config_xxx.fex 文件在打包时，被 script 工具制作成一定格式的二进制 bin 文件 sys_config_xxx.bin，然后被复制为 sys_config.bin 文件，并通过 package_boot0.cfg 一起打包进固件。在 boot0 启动时将该 sys_config.bin 读取到内存，实现 sys_config_xxx.fex 从 PC 文本文件到小机端内存二进制数据的转换；
- 接口：**melis 提供了函数接口，用于从 sys_config.bin 数据中获取配置条目的键值；
- 类型：**配置条目以 [主键] + 子键形式定位，主键名都被中括号 [] 括起来，子键的类型有 3 种：整数、字符串、GPIO，一个主键下可以同时存在 3 中类型的子键；
- 注释：**sys_config_xxx.fex 文本文件使用分号 “;” 作为注释符；



```
@SzExdroid01:~/workspace/rtos-dev/lichee/melis-v3.0/source/projects/f133-evb-board/configs$ ls
arisc.fex          image_card.cfg      package_uboot_card_product.cfg  sys_config_card.fex      sys_partition_card.fex
cardscript.fex     image_card_product.cfg  package_uboot_nand.cfg         sys_config_card_product.fex  sys_partition_card_product.fex
cardscript_product.fex  image_nand.cfg      package_uboot_nor.cfg         sys_config.fex             sys_partition_nand_dump.fex
config.fex          image_nor.cfg        rootfs.ini                   sys_config_nand.fex        sys_partition_nand.fex
defconfig          package_boot0.cfg    split_xxxx.fex               sys_config_nor.fex         sys_partition_nor_dump.fex
env.cfg            package_uboot_card.cfg  sunxi.fex                   sys_partition_card_dump.fex  sys_partition_nor.fex
```

图 2-1: 配置文件路径

```
[mainkey]
subkey1  = 1234
subkey2  = 0xFF
subkey3  = "hello!"
; 4个<>分别表示<复用功能号><上下拉><驱动能力><输出值>
subkey4  = port:PB01<1><1><1><0>
```

对于非输入输出功能的 gpio，可以将上下拉、驱动能力、输出值设置成 default。例如：

```
配置：
[cir]
ir_pin   = port: PB07<5><default><default><default>

解析：
user_gpio_set_t   irpin={0};
cir_gpio_t        pin_cir={0,0,0};
```

```
esCFG_GetKeyValue("cir", "ir_pin", &irpin, sizeof(user_gpio_set_t)/sizeof(int));  
/* script工具在解析sys_config.fex文件时, PA=1, PB=2, PC=3, PD=4, ... */  
pin_cir->gpio=(irpin->port - 1) * PINS_PER_BANK + irpin->port_num;  
pin_cir->enable_mux=irpin->mul_sel;  
pin_cir->disable_mux = 0;  
即cir_gpio_t 结构体只用到了user_gpio_set_t结构体中的port和port_num两个内容。
```

2.2 函数接口

函数接口实现文件路径为source\ekernel\drivers\osal\src\hal_cfg.c，包括如下 5 个函数；

2.2.1 esCFG_GetSecCount

```
int32_t esCFG_GetSecCount(void)
```

获取整个 sys_config.bin 配置文件中，主键的个数；

2.2.2 esCFG_GetSecKeyCount

```
int32_t esCFG_GetSecKeyCount(char *SecName)
```

获取某个主键下，子键的个数，例如：

sys_config_nor.cfg中有如下配置：

```
[mainkey]  
string = "hello!"  
cs      = port:PC03<2><1><2><default>  
clk     = port:PC02<2><0><2><default>  
used    = 0  
mosi    = port:PC04<2><0><2><default>  
miso    = port:PC05<2><0><2><default>
```

函数调用：

```
int count = 0;
```

```
count = esCFG_GetSecKeyCount( "mainkey" ); //count应该等于6
```

2.2.3 esCFG_GetKeyValue

```
int32_t esCFG_GetKeyValue(char SecName, char KeyName, int32_t Value[], int32_t  
Count)
```

获取主键 + 子键的值，存放到 value 中，count 表示 value 的长度，count 的单位为字 (4bytes) ；

sys_config_nor.cfg中有如下配置：

```
[mainkey]
string = "hello!"
cs      = port:PC03<2><1><2><default>
clk     = port:PC02<2><0><2><default>
used    = 0
mosi    = port:PC04<2><0><2><default>
miso    = port:PC05<2><0><2><default>
```

函数调用：

```
int value= 0;
char str[10] = {0};
user_gpio_set_t gpio = {0};

esCFG_GetKeyValue( "mainkey" , "used" , &value, 1);
esCFG_GetKeyValue( "mainkey" , "string" , (int32_t*)str, (sizeof(str) + 3)/sizeof(int));
esCFG_GetKeyValue( "mainkey" , "mosi" , (int32_t*)&gpio, (sizeof(user_gpio_set_t) + 3) /
    sizeof(int));
```

2.2.4 esCFG_GetGPIOSecKeyCount

int32_t esCFG_GetGPIOSecKeyCount(char *GPIOSecName)

获取主键下 GPIO 类子键的个数，例如：

sys_config_nor.cfg中有如下配置：

```
[mainkey]
string = "hello!"
cs      = port:PC03<2><1><2><default>
clk     = port:PC02<2><0><2><default>
used    = 0
mosi    = port:PC04<2><0><2><default>
miso    = port:PC05<2><0><2><default>
```

函数调用：

```
int count = 0;
count = esCFG_GetGPIOSecKeyCount( "mainkey" ); //count应该等于4
```

2.2.5 esCFG_GetGPIOSecData

int32_t esCFG_GetGPIOSecData(char GPIOSecName, void pGPIOCfg, int32_t GPI-ONum)

获取某个主键下，从第一个 GPIO 类子键开始的连续 GPIONum 个 GPIO 类子键，如果在配置文件中多个 GPIO 类子键中间穿插了个别其他类型的子键，跳过，忽略；例如：

sys_config_nor.cfg中有如下配置：

```
[mainkey]
string = "hello!"
cs      = port:PC03<2><1><2><default>
clk     = port:PC02<2><0><2><default>
used    = 0
mosi    = port:PC04<2><0><2><default>
miso    = port:PC05<2><0><2><default>
```

函数调用：

```
user_gpio_set_t gpiocfg[4] = {0};
```

//获取mainkey下4组GPIO类子键的值存放到gpiocfg数组中；

```
esCFG_GetGPIOSecData( "mainkey" , gpiocfg, 4);
```






著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、、**全志科技**、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。