



**Buildwin**  
**建荣科技**

Member of AppoTech Group

## **CW6685X SDK 二次开发文档**

---

**Versions: 0.1.1**

**Release Date: 2014.12.05**

## 目录

<b>1</b>	<b>SDK 架构说明 .....</b>	<b>2</b>
1.1	概述.....	2
1.2	工程目录结构.....	2
1.2.1	APP.....	2
1.2.2	TOOL.....	2
1.2.3	DOC.....	2
1.3	变量类型说明.....	2
<b>2</b>	<b>GUIDELINE - 快速使用指引.....</b>	<b>3</b>
2.1	SDK 功能及按键定义 .....	3
2.1.1	SDK 所有功能的宏定义 .....	3
2.1.2	用户目录选择.....	3
2.2	GUIDELINE - 第一次使用 .....	4
2.2.1	下载工具的使用.....	4
2.3	GUIDELINE - 添加一个新任务.....	4
2.4	GUIDELINE - 添加语音播报资源.....	5
2.5	GUIDELINE - 生成 EQ 系数表.....	5
2.6	GUIDELINE - 添加过滤器.....	5
2.7	GUIDELINE - LED 屏显控制 .....	6
2.8	GUIDELINE - 音乐文件的加解密.....	6
<b>3</b>	<b>基于 DEMO 板开发.....</b>	<b>7</b>
3.1	硬件准备.....	7
<b>4</b>	<b>模块使用说明 .....</b>	<b>7</b>
4.1	过滤器的使用.....	7
4.1.1	过滤器简介.....	7
4.1.2	过滤器配置.....	7
4.1.3	过滤器函数的使用.....	7
4.1.4	文件过滤器与目录过滤器的区别.....	8
<b>5</b>	<b>应用开发心得 .....</b>	<b>8</b>
5.1	添加任务.....	8
5.2	后续开发.....	8
5.2.1	任务初始化 void task_aux_enter(void) .....	8
5.2.2	任务事务处理 void task_aux_event(void).....	9
5.2.3	按键消息处理 void task_aux_deal_msg(void).....	9
5.2.4	任务显示处理 void task_aux_display(void) .....	9

## 1 SDK 架构说明

### 1.1 概述

本方案是采用 CW6685X 芯片+SPIFlash 的一个方案。程序的开发环境是 IAR720H。程序主要采用 Banked Model 的模型,即函数默认采用\_\_banked\_func。如果非 Bank 的函数,则需人工加上\_\_near\_func 字段。

在第一次使用开发环境之前,请先阅读“[GuideLine-快速使用指引](#)”。

### 1.2 工程目录结构

#### 1.2.1 APP

应用程序,主要存放 CW6685X 应用开发的程序。客户配置文件存放在 APP\config\user\_14\_164(用户配置文件夹)中,以 user\_14\_164 为例,其中主要有:

- 1) IO 文件夹: 用户常用 IO 配置,如按键,显示屏 IO 等;
- 2) mp3res 文件夹: 用户需要用到的语音播报资源;
- 3) ui 文件夹: 用户可根据实际需求自定义按键功能及显示;
- 4) config\_setting.h 文件: 用户功能配置文件,可以根据需要打开或者关闭相关功能。

#### 1.2.2 TOOL

- 1) MPTOOL : 程序下载工具;
- 2) MP3RESTOOL : 语音播报生成工具;
- 3) EQGAIN : EQ 系数生成工具。

#### 1.2.3 DOC

CW6685X 的相关说明文档。

### 1.3 变量类型说明

为了直观、方便地使用变量。我们使用以下变量类型:

u8: unsigned char 型变量,长度为 8bit;

s8: signed char 型变量,长度为 8bit;

char: 主要用于字符及字符串,如果用于运算,则使用 u8 与 s8;

u16: unsigned short 型变量, 长度为 16bit;  
s16: signed short 型变量, 长度为 16bit;  
u32: unsigned long 型变量, 长度为 32bit;  
s32: signed long 型变量, 长度为 32bit;  
string: unsigned short 型变量, 长度为 16bit;

## 2 GuideLine - 快速使用指引

### 2.1 SDK 功能及按键定义

#### 2.1.1 SDK 所有功能的宏定义

SDK 所有功能的宏定义位于 config\_setting.h 文件中, 用户可自行选择屏蔽或打开所需要的模块。

#### 2.1.2 用户目录选择

在 config.h 中选择用户目录, 以 NO.14-164 为例:

若需要使用匹配 NO.14-164 样机的 SDK, 则在 config.h 中选择

```
#define CFG_DIR    user_14_164
```

CW6685X 主要采用标准开发板的 10 按键进行功能定义。从上到下, 从左到右依次为:

	PREV/VOL-	PWR/P/P	NEXT/VOL+	HSF	
VOL-	PREV	P/P	NEXT	VOL+	MODE

- 1) 上一曲/音量减;
- 2) 开关机/暂停/播放;
- 3) 下一曲/音量加;
- 4) 接听/挂断电话/模式切换;
- 5) 音量减;
- 6) 上一曲;
- 7) 播放/暂停;
- 8) 下一曲;

- 9) 音量加;
- 10) 切换模式;

## 2.2 GuideLine - 第一次使用

### 2.2.1 下载工具的使用

- 1) 给底板接 5V 直流电, 打开 SPIFlash 下载工具。
  - 2) 选择 Debug/Exe 目录下, app.cod 文件。
  - 3) 用 USB 线连接底板与 PC。
  - 4) 按住 PREV 键不放手, 打开电源开关, 给底板供电。
  - 5) 在 SPIFlash 下载工具上可以发现设备。
  - 6) 下载测试程序, 等待完成。
  - 7) 移除 USB 线, 复位芯片, 则可以正常运行 SDK 程序。
  - 8) 可以在 SDK 中修改程序, 再次下载, 进行程序调试。
- 更加具体的使用说明请阅读《SPIFLASH 工具使用说明》中 MPTOOL 工具说明部分。

## 2.3 GuideLine – 添加一个新任务

- 1) 复制 task\_template.c 文件, 改成自己需要的任务名;
- 2) 打开该文件, 根据自己的任务名搜索替换 “task\_template” 字符串 (替换的时候注意大小写字母的匹配), 这里用 task\_user 替换 task\_template 为例说明;
- 3) 修改文件头的功能说明信息;
- 4) 修改任务结构体的名字及注释;
- 5) 如果有版本控制软件, 将该文件添加到版本中, 如果没有, 则跳过这一步;
- 6) 打开工程文件, 在左侧工程视图中将该文件加入 task 组, 注意: 这个时候务必记得保存整个工程, 以免版本管理时造成工程文件未添加;
- 7) 在 task.h 中, 增加 TASK\_USER 任务的编号;
- 8) 在 task.h 中, 增加 extern void task\_user(void) 函数声明定义;
- 9) 在 task\_user 和 task\_user\_enter 函数中用 TASK\_USER 替换 TASK\_TEMPLATE;
- 10) 在 task.c 中的 run\_task 函数里, 增加 task\_user 的函数入口;
- 11) 在 task.c 中的 tbl\_task\_order 表中根据任务切换的顺序增加 TASK\_USER;
- 12) 测试该任务是否正常运行, 继续完成任务的后续开发;
- 13) 理解分段机制的用户可以对函数进行分段, 并在 APP.xcl 中加入相应的段, 这里要注意的是: 如果没有常量段, 则可在 -P 中添加。如果有定义常量段, 则需要用 -Z 将

常量段和程序段定义为同一段。

## 2.4 GuideLine – 添加语音播报资源

- 1) 把要添加的 MP3 音乐文件拷贝到软件所在目录下，例如：

```
#define CFG_DIR      user_14_164(客户配置文件)
```

那么将要添加的 MP3 音乐拷贝到\APP\config\user\_14\_164(客户配置文件夹)\mp3res 文件夹内；

注意：MP3 音乐文件的文件名必须为英文。

- 2) 打开 MP3RESTOOL.exe，点击“生成语音波报表”按钮，弹出“Finish”后生成新的 mp3res.h 和 mp3res.bin 文件。



- 3) 重新编译程序即可。

或可阅读《SPIFLASH 工具使用说明》中 MP3RESTOOL 工具说明部分。

## 2.5 GuideLine – 生成 EQ 系数表

使用方法请阅读《建荣 MP3 主控 EQ 调试工具使用说明》。

## 2.6 GuideLine – 添加过滤器

过滤器是 SDK 对文件扫描模块进行封装的一大特色。通过过滤器，可以定制出不同的满足用户需要的扫描方式。

过滤器模块的相关说明，可阅读“[过滤器的使用](#)”。

添加一个过滤器的顺序如下：

- 1) 在 user/file\_filter.c 文件中，复制 file\_filter\_music 与 dir\_filter\_music 两个函数。根

据自己的实际需要，修改文件名。

- 2) 按照自己的需要，对过滤器函数进行修改。
- 3) 在相应的初始化位置，调用 `f_scan_set` 来进行过滤器配置。
- 4) 可以使用文件系统的相关函数，来进行相应的应用。

## 2.7 GuideLine – LED 屏显控制

SDK 支持 LED 两种显示屏，分别是 LED\_5COM7SEG 和 LED\_7PIN 屏，以 LED\_7PIN 屏为例说明：

- 1) 在 `config_setting.h` 中，  
`#define THEME_SELECT` `THEME_LEDSEG_7PIN`
- 2) 在 `led_7p7s.c` 中定义了此屏的显示驱动函数；
- 3) SDK 为了兼容不同类型的显示屏，定义了统一的接口函数，主要有以下三个：
  - a) `#define ledseg_init()` `led_7p7s_init()`  
此函数为初始化函数，主要设置显示用到的 IO；
  - b) `#define ledseg_value_set()` `led_7p7s_value_set()`  
此函数为 7 脚 LED 屏的真值映射表，主要用于更新显示 BUF；
  - c) `#define ledseg_scan()` `led_7p7s_scan()`  
此函数为 LED 的扫描函数，在中断中调用。

## 2.8 GuideLine – 音乐文件的加解密

在使用到一些用户自定义教材文件时，可能需要对音乐文件以及歌词进行加解密处理。加解密处理的流程如下：

- 1) 使用自定义算法加密音乐文件及歌词文件。这里需要注意的是，为了方便解密程序的编写，一般建议将加密后的文件大小对齐至 512Byte；
- 2) 在 `user_fat.c` 文件中增加解密算法。如果解密速度达不到要求的话，必要时可采用汇编进行加速。
- 3) 音乐的文件系统接口函数所调用的函数，必须放在公共区，并加上 `__near_func` 的定义。

## 3 基于 DEMO 板开发

### 3.1 硬件准备

所需的硬件包括:

- 1) NO.14-164;
- 2) 5V 电源;
- 3) USB 连接线;
- 4) 串口线;
- 5) 耳机或小喇叭;

## 4 模块使用说明

### 4.1 过滤器的使用

#### 4.1.1 过滤器简介

在 SDK 中引用了一个过滤器的概念。过滤器主要应用于文件的查找、打开、文件导航列表等。

#### 4.1.2 过滤器配置

```
void f_scan_set(u8 scan_mode, bool (*file_filter)(void), bool (*dir_filter)(void))
```

传入: scan\_mode: 文件扫描模式。为 SCAN\_SUB\_FOLDER 时, 则扫描子目录。为 NULL 时, 则只扫描本层目录

file\_filter 为文件过滤器的函数指针。主要用于过滤扫描的文件。其返回值为 bool 型, 为 true, 则表示找到符合条件的文件, 结束查找。返回 false, 则继续查找。

dir\_filter 为文件夹过滤器的函数指针。同样, 返回 true 则找到符合条件的文件, 停止扫描。否则也将继续。

#### 4.1.3 过滤器函数的使用

在 APP 中, 我们的过滤器函数均放在 user/file\_filter.c 文件下。其间, 可能用到的函数或变量结构体有:

get\_scan\_dept(), 用来获取当前文件/文件夹所在的目录级数, 如果在根目录下则为 0;

fat\_file\_opt.fname, 文件的短文件名。一般可以用来作为文件扩展名过滤;

fs\_apiinfo.file\_number 与 fs\_apiinfo.file\_count, 一般用来统计扫描的有效文件个数。主要用于控制 f\_open 打开第 N 个文件时使用。



`fat_file_opt.fattrib`, 文件项的类型。AM\_DIR 表示该项为目录项。

#### 4.1.4 文件过滤器与目录过滤器的区别

文件过滤器, 主要是用来获取文件夹下的目录下, 用于文件的打开、文件或文件夹的列表等。而目录过滤器则是用来判断是否需要进入该目录进行扫描, 对一些目录进行屏蔽扫描等。

简单的说, 文件过滤器是用于本层目录下, 所有文件或文件夹的筛选。而目录过滤器则是用于筛选是否要进入本层目录下的子目录。

这里需要注意的是: 如果是列出目录下的所有文件夹, 则使用的为文件过滤器, 并非目录过滤器。

## 5 应用开发心得

以添加 AUX 任务为例详细说明如何添加任务及各函数的使用。

### 5.1 添加任务

- 1) 复制 `task_template.c` 文件, 把文件名改成 `task_aux.c`;
- 2) 打开该文件, 用 `task_aux` 搜索替换 “`task_template`” 字符串, 注意区分大小写;
- 3) 修改文件头的功能说明信息;
- 4) 修改任务结构体的名字及注释;
- 5) 打开工程文件, 在左侧工程视图中将该文件加入 `task` 组, 并保存工程;
- 6) 在 `task.h` 中, 增加 `TASK_AUX` 任务的编号;
- 7) 在 `task_aux.c` 中, 用 `TASK_AUX` 搜索替换 `TASK_TEMPLATE`;
- 8) 在 `task.c` 中, 增加 `task_aux` 的函数入口;
- 9) 在 `task.c` 中的 `tbl_task_order` 表中根据任务切换的顺序增加 `TASK_AUX`;
- 10) 测试该任务是否正常运行, 继续完成任务的后续开发。

### 5.2 后续开发

此流程可参考 `task_music.c`。

#### 5.2.1 任务初始化 `void task_aux_enter(void)`

```
#if LINEIN_DETECT_EN
```

```

        if (!device_activate(DEVICE_LINEIN)) {           //检查linein

            task_ctl.work_sta = TASK_EXIT;              //设备无效，返回主菜单

            return;

        }

    #endif

```

若AUX有检测IO，先确定设备是否有效，如果无效返回主菜单，需要用到存储设备的任务都要先进行检测。

### 5.2.2 任务事务处理 void task\_aux\_event(void)

```

comm_event();                                           //调用公共事件
#if LINEIN_DETECT_EN
    if (!device_activate(DEVICE_LINEIN)) {
        task_ctl.work_sta = TASK_EXIT;
    }
#endif

```

- 1) comm\_event(): 公共事件处理，插入其它设备如 U 盘/TF 卡等会退出 AUX 任务进入相应的任务中；
- 2) 检查设备是否在线，若不在线退出此任务，进入下一个任务。

### 5.2.3 按键消息处理 void task\_aux\_deal\_msg(void)

主要有四个按键消息 KU\_MODE, KU\_PLAY, KU/KL\_VOL\_DOWN, KU/KL\_VOL\_UP:

- 1) KU\_MODE: 切换到下一个任务；
- 2) KU\_PLAY: 播放与静音切换；
- 3) KU/KL\_VOL\_DOWN: 音量减；
- 4) KU/KL\_VOL\_UP: 音量加。

### 5.2.4 任务显示处理 void task\_aux\_display(void)

```

ledseg_ocx_event();
switch (ledseg_ocx.disp_status) {
case LEDSEG_OCX_NULL:

```

```
    ledseg_disp(MENU_LINEIN);  
    break;  
default:  
    ledseg_ocx_display();  
    break;  
}
```

ledseg\_ocx\_event()是 LED 控件事件，控制当前的显示状态及显示的时间。