IOOUT

1.介绍

IOOUT 是一个 GPIO 输出口控制扩展模块，提供 **启动** 、 **暂停** 、**设置** 和 **停止** 接口，可以根据你所需的周期、占空比和总时间，更加便捷地管理多个IO口时序。

1.1 文件结构

源文件	描述
\ioout	IOOUT操作接口和实现源码
\demo	stm32f10x平台基于HAL库的demo

1.2 资源占用

单片机资源： 定时器一个

ROM: 1Kb

RAM: 视IO口多少

2. 移植

2.1 定时器调用

需要用户设置 ioout_Time_Interval，初始化一个定时器，并且定时器周期和 ioout_Time_Interval 保持一致

```
const uint16_t ioout_Time_Interval = 10;    /**< 查询间隔     */

void ioout_CallBackProcRoutine(void);       /**< IO控制口处理函数   */
```

2.2 配置控制ID

```c
/**
  * @brief IO输出控制ID
  */
typedef enum
{
    ID_IOOUT_LED0 = 0,
    ID_IOOUT_LED1,

} iooutId_Typedef;
```

2.3 配置控制数量

```c
/**
  * @brief IO输出控制数量类型定义
  */
typedef enum
{
    IOOUT_MAX = (ID_IOOUT_LED1 + 1),

} ioOutIndexNum_Typedef;
```

2.4 初始化

初始化 IOOUT 管理结构体。

```
void ioout_Config(void)
{
    ioout_ListInit();

    /*---------------- 移植配置Start */

    if(IOOUT_NO_ERR != ioout_Init(ID_IOOUT_LED0,GPIO_SetLed0))
    {
        //do something
    }

    if(IOOUT_NO_ERR != ioout_Init(ID_IOOUT_LED1,GPIO_SetLed1))
    {
        //do something
    }

    /*---------------- 移植配置End */
}
```

3. API

API接口在 */ioout/ioout.h* 声明，下方内容可以使用 CTRL+F 搜索。

3.1 核心结构体配置

初始化的 IOOUT 的核心结构体，初始化后才可以使用下面的API。

> void ioout_Config(void)

3.2 端口函数初始化

查找空闲端口，并初始化该端口，初始化成返回 *IOOUT_NO_ERR* ，其中，IO口接口函数需要遵循 *void function(uint8_t)* 格式。

> iooutErrCode_Typedef ioout_Init(iooutId_Typedef iooutId,IOOUTCALLBACK timproc)

参数	描述
iooutId	输出口ID

timproc	IO口接口函数

示例：

```
if(IOOUT_NO_ERR != ioout_Init(ID_IOOUT_LED0,GPIO_SetLed0))
{
    //do something
}
```

3.3 时间参数设置

设置端口时间并启动，时间参数以 ms 为单位，参数必须是查询 ioout_Time_Interval 时间的整数倍，workTime 为 0 停止，ctlTime 为 0 一直保持周期，设置成功返回 IOOUT_NO_ERR 。

> iooutErrCode_Typedef ioout_Set(iooutId_Typedef iooutId,uint32_t interval,uint32_t workTime,uint32_t ctlTime)

参数	**描述**
iooutId	输出口ID
interval	间隔时间
workTime	持续时间
ctlTime	总时间

示例：

```
ioout_Set(ID_IOOUT_LED0,1000,1000,0);//设置LED0保持周期2s,占空比50%
ioout_Set(ID_IOOUT_LED1,200,100,2000);//设置LED1周期300ms，占空比1/3，总时间2s
```

> 注：ioout_Time_Interval = 10

3.4 启动

启动端口调用，启动成功返回 IOOUT_NO_ERR 。

```
iooutErrCode_Typedef ioout_Start(iooutId_Typedef iooutId)
```

参数	描述
iooutId	输出口ID

3.5 暂停

暂停端口调用，暂停端口 **不清除端口时间数据** ，暂停返回 IOOUT_NO_ERR 。

```
iooutErrCode_Typedef ioout_Pause(iooutId_Typedef iooutId)
```

参数	描述
iooutId	输出口ID

3.6 停止

停止端口调用，暂停端口将 **清除端口时间数据** ，暂停返回 IOOUT_NO_ERR 。

```
iooutErrCode_Typedef ioout_Stop(iooutId_Typedef iooutId)
```

参数	描述
iooutId	输出口ID

3.7 删除

将端口从结构体中删除，如果未再次初始化，将无法正常调用，删除成功返回 IOOUT_NO_ERR 。

```
iooutErrCode_Typedef ioout_Kill(iooutId_Typedef iooutId)
```

参数	描述
iooutId	输出口ID

4. DEMO

DEMO 使用芯片为 STM32F103RCT6，使用 HAL 库，通过延时方式测试参数是否设置成功。

5. 其它

If you have any question,Please connact redoc/619675912@qq.com