类别	内容
关键词	24位ADC API手册
摘要	本文档描述了ZML165 24位ADC操作API接口函数

修订历史

版本	日期	原因
1.0.00	2019/08/18	创建文档

目 录

1. 简介

本文档内容只适用于对 ZML165 内部 24 位 ADC（下文简称 ADC） 进行操作。ADC 接口以静态库的形式发布，本文将详细介绍 ZML165 中 ADC 所有的 API 接口，用户无需关心相关芯片资料，即可通过本文提供的 API 接口完成对 ZML165 内部 ADC 的相关操作。 ADC 操作相关静态库文件存放于 AMetal 仓库中， AMetal 已开源于 Github，其开源网址为“https://gitee.com/zlgopen/ametal” ，用户可直接通过此页面下载 AMetal 开源代码包来获取 ZML165 SDK 以及相关资料，具体操作参见《AMetal 代码仓库使用说明(TortoiseGit)》。

注意： AMetal 相关资料会持续更新，用户打开上面下载链接后，以打开的实际页面为准，选择最新的资料下载。

2. 相关 API 接口

2.1 初始化函数接口

在使用 ADC 之前， 可直接通过初始化函数完成对 ADC 的初始化， 初始化函数的定义
如下：

```
/**\brief 24 位 ADC 初始化函数 */
am_zml165_adc_handle_t am_zml165_adc_init (am_zml165_adc_dev_t                *p_dev,
                                           const am_zml165_adc_devinfo_t      *p_devinfo);
```

该函数将返回一个用户用以操作 ADC 的标准服务句柄 handle，其类型为
am_zml165_adc_handle_t，具体定义如下：

```
/**\brief          AM_ZML165_ADC   服务句柄定义   */
          typedef am_zml165_adc_dev_     *am_zml165_adc_handle_t;
```

由此可见，函数返回值为指向 ADC 设备的指针，也就相当于返回的 handle 等效于 p_dev。初始化函数共有 2 个参数： p_dev 和 p_devinfo。 其中 p_dev 为用户定义的设备结构体；p_devinfo 为用户自定义的 ADC 参数，其类型具体定义如下：

```
/**
* \brief AM_ZML165 ADC 设备信息结构体
*/
typedef struct am_zml165_adc_devinfo {
am_zml165_adc_config_t config; /**< \brief   ADC 初始配置/
uint32_t    vref;   /**<       \brief 参考电压,单位  mV   */
}      am_zml165_adc_devinfo_t;
```

其中 vref 为 ADC 的参考电压， 根据具体的电路决定其值， config 为 ADC 初始配置，其具体定义如下：

```
/**
\brief AM_ZML165 ADC 配置项结构体
*/
typedef struct     am_zml165_adc_config {
      int8_t pga; /** \brief 输出增益倍数设置,若采用上次设置值则填入-1 */
      int8_t speed; /** \brief 输出速率设置,若采用上次设置值则填入 -1 */
      int8_t channel; /** \brief 输出速率通道设置,若采用上次设置值则填入 -1 */
      int8_t refo_off; /** \brief Vout 是否使能,若采用上次设置值则填入 -1
*/}am_zml165_adc_config_t;
```

其中 pga 表示 ADC 的增益系数，其可选值有 1/2/64/128 四个值； speed 表示 ADC 的输出速率，其可选值有 10/40/640/1280 四个值； channel 表示 ADC 通道选择，可选择项有通道 A、通道 B 以及温度通道； refo_off 表示是否使能 ADC 内部 Vout 输出。 若在进行 ADC 配置时， 该四个参数中若存在采样上次设置值的配置项，直接在对应项填入 -1 即可。 设置值在 API 中已经做了相关宏定义，用户直接使用相关宏定义即可完成对 config 配置结构体的赋值操作；具体宏定义详见表 2.1。

注意：执行初始化函数时， ADC 无上次设置值，此时禁止填入-1。

配置项	宏定义
增益参数	AM_ZML165_ADC_PGA_1
	AM_ZML165_ADC_PGA_2
	AM_ZML165_ADC_PGA_64
	AM_ZML165_ADC_PGA_128
通道参数	AM_ZML165_ADC_CHANNEL_A
	AM_ZML165_ADC_CHANNEL_B
	AM_ZML165_ADC_CHANNEL_TEMP
输出速率	AM_ZML165_ADC_SPEED_10HZ
	AM_ZML165_ADC_SPEED_40HZ
	AM_ZML165_ADC_SPEED_640HZ
	AM_ZML165_ADC_SPEED_1280HZ
Vout 配置	AM_ZML165_ADC_VOUT_ENABLE
	AM_ZML165_ADC_VOUT_DISABLE

2.2 ADC 配置接口

在 ADC 的使用过程中，根据应用的场景不同， ADC 的配置可能随时进行修改，ZML165 中提供了 ADC 配置项修改函数，具体函数原型如下：

```
/**
* \brief ZML165 ADC 写入配置
*
* \param[in] p_dev : ZML165_ADC 操作句柄
* \param[in] speed : p_config 指向配置结构体的指针
*
* \retval AM_OK : 设置成功
* AM_ERROR : 设置失败， ADC 未准备好
*/
uint8_t am_zml165_adc_config_load(am_zml165_adc_dev_t *p_dev,
am_zml165_adc_config_t *p_config);
```

其中 p_dev：初始化函数返回的标准服务句柄（即 handle）；

- p_config： ADC 对应配置结构体
- 设置成功则返回 AM_OK;
- 参数失败则返回 AM_ ERROR 。

注意： ADC 参数设置后，其参数生效需要等待四个输出速率周期满足模拟输入信号的建立，所以调用该函数后，需要进行适当的延时操作，等待数据建立完毕。

2.3 读取 ADC 数据

ZML165 中 ADC 驱动，兼容 AMetal 中标准 ADC 驱动，若要获取 ADC 电压，可参照以下代码进行操作。

```
/* 定义 ZML165_ADC 实例 */
```

```
static am_zml165_adc_dev_t __g_zml165_adc_dev;
/* 定义 ZML165_ADC 实例信息 */
const am_zml165_adc_devinfo_t __g_zml165_adc_info = {
{
AM_ZML165_ADC_PGA_1,
AM_ZML165_ADC_SPEED_10HZ,
AM_ZML165_ADC_CHANNEL_A,
AM_ZML165_ADC_VOUT_ENABLE
},
2500
};
void demo_zml165_adc_measure_entry()
{
int32_t adc_val[10];
am_zml165_adc_config_t config;
am_zml165_adc_handle_t handle = am_zml165_adc_init(&__g_zml165_adc_dev,
&__g_zml165_adc_info);
config.pga = AM_ZML165_ADC_PGA_1;
config.speed = AM_ZML165_ADC_SPEED_10HZ;
config.channel = AM_ZML165_ADC_CHANNEL_B;
config.refo_off = AM_ZML165_ADC_VOUT_DISABLE;
am_zml165_adc_config_load(handle, &config);
am_mdelay(100);
am_adc_read(&handle->adc_serve, 0, (void *)adc_val, AM_NELEMENTS(adc_val));
for(i = 0 ; i < AM_NELEMENTS(adc_val); i++){
if(adc_val[i] >= 0x800000) {
adc_val[i] &= 0x7fffff;
adc_val[i] |= 0xff800000;
}
vol += (adc_val[i] / (double)AM_NELEMENTS(adc_val));
}
vol = (double)((double)(vol / ((1 << 24) -1)) * handle->p_devinfo->vref);
vol *= 10000;
vol /= gpa[config.pga];
if(vol > 0){
am_kprintf("Voltage is %d.%04d mV\r\n", (int32_t)vol/10000, (int32_t)vol%10000);
}else {
vol *= -1;
am_kprintf("Voltage is -%d.%04d mV\r\n", (int32_t)vol/10000, (int32_t)vol%10000);
}
}
```

2.4 休眠模式

在某些低功耗应用场景下， 需要降低 ADC 的功耗，可以通过以下函数来使 ADC 进

入以及退出低功耗模式。

```
/** \brief 进入休眠模式*/
void am_zml165_adc_power_down_enter (am_zml165_adc_dev_t *p_dev);
/** \brief 退出休眠模式 */
void am_zml165_adc_power_down_out (am_zml165_adc_dev_t *p_dev);
```

其中 p_dev：初始化函数返回的标准服务句柄（即 handle）；

3. 免责声明

应用信息：本应用信息适用于嵌入式产品的开发设计。客户在开发产品前，必须根据其产品特性给予修改并验证。

本着为用户提供更好服务的原则，广州致远微电子有限公司（下称“致远微电子”）在本手册中将尽可能地为用户呈现详实、准确的产品信息。但介于本手册的内容具有一定的时效性，致远微电子不能完全保证该文档在任何时段的时效性与适用性。致远微电子有权在没有通知的情况下对本手册上的内容进行更新，恕不另行通知。为了得到最新版本的信息，请尊敬的用户定时访问官方网站或者与致远微电子工作人员联系。感谢您的包容与支持！