类别	内容
关键词	24位ADC、API手册
摘要	

修订历史

版本	日期	原因
1.0.00	2019/09/03	创建文档

目 录

1. 适用范围

本文档内容只适用于对 ZML166 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 ZML166 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 ZML166 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://github.com/zlgopen/am-etal”，用户可直接通过此页面下载 AMetal 开源代码包来获取 ZML166 SDK 以及相关资料，具体操作参见《AMetal 代码仓库使用说明(TortoiseGit)》。

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_zml166_adc_handle_t     am_zml166_adc_init (am_zml166_adc_dev_t                *p_dev,
                                               const am_zml166_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_zml166_adc_handle_t，具体定义如下：

```
/**
 * \brief AM_ZML166_ADC 句柄定义
 */
typedef     am_zml166_adc_dev_t *    am_zml166_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，其具体定义如下：

```
/**
 * \brief AM_ZML166_ADC 设备信息结构体
 */
typedef struct am_zml166_adc_devinfo {
    uint32_t    vref;                                   /**< \brief ADC 参考电压，单位：mV */
    uint32_t    timeout;                                /**< \brief 超时时间，单位为毫秒 */
} am_zml166_adc_devinfo_t;
```

vref 为 ADC 的参考电压，timeout 为 ADC 转换的超时时间，用户需根据实际情况进行参数修改。

注意：执行初始化函数之后，ADC 处于默认配置状态，增益为 1，输出速率为 10Hz，未进行通道配置，用户若需要对参数进行修改，请参看第二小节内容。

2.2 ADC 配置接口

2.2.1 ADC 前置放大器增益配置

调用以下函数即可进行配置。

```
/** \brief   设置 ADC 增益配置值 */
am_err_t     am_zml166_adc_gain_set(am_zml166_adc_dev_t     *p_dev,
                                    uint16_t                gain);
```

其中 p_dev：初始化函数返回的标准服务句柄（即 handle）；

gain ：增益值。设置值有 1、 2、 4、 8、 16、 32、 64、 128;

设置成功则返回 AM_OK;

参数无效则返回 -AM_EINVAL。

2.2.2 ADC 前置放大器增益获取

调用以下函数即可获取前置放大器增益。

```
/** \brief   获取 ADC 增益值 */
am_err_t     am_zml166_adc_gain_get (am_zml166_adc_dev_t *p_dev,
                                     uint8_t                *p_gain);
```

其中 p_dev：初始化函数返回的标准服务句柄（即 handle）；

p_gain：保存返回增益值的指针；

获取成功则返回 AM_OK;

获取失败则返回 - AM_ENOTSUP。

2.2.3 ADC 通道配置

调用以下函数即可设置 ADC 的通道：

```
/** \brief   设置 ADC 通道值 */
am_err_t   am_zml166_adc_mux_set (am_zml166_adc_dev_t   *p_dev,
                                  uint8_t                chan);
```

其中 p_dev：初始化函数返回的标准服务句柄（即 handle）；

chan ： MUX 通道设置值，MUXN 与 MUXP 相关通道宏定义相或的值，

如 AM_ZML166_ADC_INNS_AIN3 | AM_ZML166_ADC_INPS_AIN1，

即为配置通道 1 与通道 3 作为 ADC 的输入。

设置成功则返回 AM_OK

参数无效则返回 -AM_EINVAL

2.2.4 ADC 通道配置值获取

调用以下函数即可获取 ADC 的通道配置值：

```
/** \brief   获取 ADC 通道设置值 */
am_err_t   am_zml166_adc_mux_get (am_zml166_adc_dev_t   *p_dev,
                                  uint8_t                *p_chan);
```

其中 p_dev：初始化函数返回的标准服务句柄（即 handle）；

p_chan：保存返回增益值的指针，返回值含义见通道设置配置函数；

设置成功返回 AM_OK

参数无效返回 -AM_EINVAL 无效参数

2.2.5 ADC 采样率设置

调用以下函数即可设置 ADC 的采样率：

```
/** \brief   设置 ADC 采样率*/
am_err_t   am_zml166_adc_speed_set (am_zml166_adc_dev_t     *p_dev,
                                    uint8_t                  speed);
```

其中 p_dev：初始化函数返回的标准服务句柄（即 handle）；

speed：ADC 采样速率。设置值有 AM_ZML166_ADC_DR_12_5

AM_ZML166_ADC_DR_25　　AM_ZML166_ADC_DR_50

AM_ZML166_ADC_DR_100　　AM_ZML166_ADC_DR_200

设置成功则返回　AM_OK

参数无效则返回 -AM_EINVAL

设置失败则返回　AM_ERROR

2.2.6 ADC 采样率获取

调用以下函数即可获取 ADC 的采样率：

```
/** \brief   获取 ADC 采样率*/
am_err_t    am_zml166_adc_speed_get (am_zml166_adc_dev_t   *p_dev,
                                     uint8_t                *p_ speed);
```

其中 p_dev：初始化函数返回的标准服务句柄（即 handle）；

p_ speed： 保存获取到的 speed 值的指针，其含义与设置时一致；

设置成功返回 AM_OK

参数无效返回 -AM_EINVAL 无效参数

2.2.7 寄存器相关操作

本文只介绍了部分常用操作的 API，不常用的配置项可以通过直接操作寄存器的方式来进行配置，具体的寄存器参见用户手册。

```
/** \brief   ADC 寄存器设置函数   */
am_err_t   am_zml166_adc_reg_set (am_zml166_adc_dev_t *p_dev,
                                  uint8_t              addr,
                                  uint32_t             data);

/** \brief   ADC 寄存器获取函数   */
am_err_t   am_zml166_adc_reg_get (am_zml166_adc_dev_t *p_dev,
                                   uint8_t              addr,
                                   void                *p_data);
```

以上两个函数分别为写寄存器以及读寄存器，其中

p_dev：初始化函数返回的标准服务句柄（即 handle）；

addr　：读写寄存器的地址；

data　：待写入的寄存器值；

p_data：表示保存读出的寄存器的值得指针。

操作成功返回 AM_OK；

参数无效返回 -AM_EINVAL。

2.3　ADC 工作接口

使用函数

```
/** \brief   获取 ADC 采样值*/
int   am_adc_read(am_adc_handle_t     handle,
                  int                 chan,
```

```
                    void                *p_val,
                    uint32_t            length);
```

即可获取当前 ADC 的采样值。其中，

handle ：ADC 的标准服务句柄中的 adc_serve 成员（即 handle-> adc_serve）；

chan ：通道号，在 ZML166 中，该参数为 0；

p_val ：转换结果存放的缓冲区，数据右对齐；

length ：采样的次数，采样次数必须小于或者等于缓冲区大小；

操作成功则返回 AM_OK；

ADC 通道不存在则返回 -AM_ENXIO；

无效参数则返回 -AM_EINVAL。

2.4 休眠模式

使用函数

```
/** \brief  进入休眠模式*/
am_err_t  am_zml166_adc_powerdown_entry(am_zml166_adc_dev_t *p_dev);
/** \brief  退出休眠模式 */
am_err_t  am_zml166_adc_powerdown_exit(am_zml166_adc_dev_t *p_dev);
```

即可实现对 ADC 休眠模式的操作。

3. 免责声明

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！