

PY32F030_003_002A 的 ADC 应用 注意事项

前言

PY32F030_003_002A 微控制器具有 1 个 12 位的 SARADC (successive approximation analog-to-digital converter)。该模块共有 12 个要被测量的通道，包括 10 个外部通道和 2 个内部通道。支持多种使用方式。

本应用笔记将帮助用户了解 PY32F030_003_002A 的 ADC 模块应用的注意事项并快速着手开发。

表 1. 适用产品

类型	产品系列
微型控制器系列	PY32F030、PY32F003、PY32F002A

目录

1	ADC 禁能	3
1.1	注意事项	3
1.2	操作流程	3
1.3	代码示例	3
2	ADC 复位	4
2.1	注意事项	4
2.2	操作流程	4
2.3	代码示例	4
3	ADC 校准	5
3.1	注意事项	5
3.2	操作流程	5
3.3	代码示例	5
4	ADC 仅使用通道 0 的情况	7
4.1	注意事项	7
4.2	操作流程	7
4.3	代码示例	7
5	ADC 单次模式	9
5.1	注意事项	9
5.2	操作流程	9
5.3	代码示例	9
6	ADC 如何利用 VREFINT (1.2V) 测量目标电压值	11
6.1	注意事项	11
6.2	操作流程	11
6.3	代码示例	11
7	AD 口配置注意事项	12
7.1	注意事项	12
8	版本历史	13

1 ADC 禁能

1.1 注意事项

- ADC 使能后软件不能禁能，需要复位 ADC 模块，然后重新初始化 ADC，最后启动 ADC。

1.2 操作流程

- 配置 RCC_APBSTR2 寄存器 ADCRST = 1，再配置 ADCRST = 0 复位 ADC 模块；
- 初始化 ADC 模块；
- 启动 ADC 转换。

1.3 代码示例

```
__HAL_RCC_ADC_FORCE_RESET();  
__HAL_RCC_ADC_RELEASE_RESET();//ADC 复位  
ADC_Init();//ADC 初始化  
ADC_Cha_SW();//ADC 通道初始化  
if(HAL_ADCEx_Calibration_Start(&AdcHandle) != HAL_OK) while(1);//ADC 校准  
HAL_ADC_Start(&AdcHandle);//启动 ADC 转换
```

2 ADC 复位

2.1 注意事项

- 系统上电后硬件不会复位 ADC 模块寄存器, 需要软件复位, 然后重新初始化, 最后启动 ADC。

2.2 操作流程

- 配置 RCC_APBSTR2 寄存器 ADCRST = 1, 再配置 ADCRST = 0 复位 ADC 模块;
- 初始化 ADC 模块;
- 启动 ADC 转换。

2.3 代码示例

```
_HAL_RCC_ADC_FORCE_RESET();  
_HAL_RCC_ADC_RELEASE_RESET();//ADC 复位  
ADC_Init();//ADC 初始化  
ADC_Cha_SW();//ADC 通道初始化  
if(HAL_ADCEx_Calibration_Start(&AdcHandle) != HAL_OK) while(1);//ADC 校准  
HAL_ADC_Start(&AdcHandle);//启动 ADC 转换
```

3 ADC 校准

3.1 注意事项

- 当 ADC 的工作条件发生改变时 (VCC 改变是 ADC offset 偏移的主要因素, 温度改变次之), 推荐进行再次校准操作;
- 第一次使用 ADC 模块前, 必须增加软件校准流程。

3.2 操作流程

- 确认 ADEN = 0、CKMODE 选择系统时钟;
- 设置 ADCAL = 1;
- 等待到 ADCAL = 0;
- 校准完成后, 启动 ADC 的转换。

3.3 代码示例

Step1: ADC 初始化配置 CKMODE

```
void ADC_Init()
```

```
{
```

```
    AdcHandle.Instance                = ADC1;
```

```
    if (HAL_ADC_DeInit(&AdcHandle) != HAL_OK) while(1);
```

```
    //CKMODE 选择 PCLK/2
```

```
    AdcHandle.Init.ClockPrescaler      = ADC_CLOCK_SYNC_PCLK_DIV2;
```

```
    AdcHandle.Init.Resolution          = ADC_RESOLUTION_12B;
```

```
    AdcHandle.Init.DataAlign           = ADC_DATAALIGN_RIGHT;
```

```
    AdcHandle.Init.ScanConvMode        = ADC_SCAN_DIRECTION_BACKWARD;
```

```
    AdcHandle.Init.EOCSelection         = ADC_EOC_SINGLE_CONV;
```

```
    AdcHandle.Init.LowPowerAutoPowerOff = DISABLE;
```

```
    AdcHandle.Init.LowPowerAutoWait     = ENABLE;
```

```
    AdcHandle.Init.ContinuousConvMode   = ENABLE;
```

```
    AdcHandle.Init.DiscontinuousConvMode = DISABLE;
```

```
    AdcHandle.Init.ExternalTrigConv     = ADC_SOFTWARE_START;
```

```
    AdcHandle.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
```

```
    AdcHandle.Init.DMAContinuousRequests = DISABLE;
```

```
    AdcHandle.Init.Overrun              = ADC_OVR_DATA_OVERWRITTEN;
```

```
    AdcHandle.Init.SamplingTimeCommon   = ADC_SAMPLETIME_3CYCLES_5;
```

```
    if (HAL_ADC_Init(&AdcHandle) != HAL_OK) while(1);
```

```
}
```

Step2: ADC 校准

```
__HAL_RCC_ADC_FORCE_RESET();
```

```
__HAL_RCC_ADC_RELEASE_RESET(); //ADC 复位
```

```
ADC_Init();//ADC 初始化
```

```
ADC_Channels();//ADC 通道初始化
```

```
if(HAL_ADCEX_Calibration_Start(&AdcHandle) != HAL_OK) while(1);//ADC 校准
```

```
HAL_ADC_Start(&AdcHandle); //启动 ADC 转换
```

4 ADC 仅使用通道 0 的情况

4.1 注意事项

- ADC 在连续模式或不连续模式下，仅使用通道 0 时，必须选择扫描序列向下。

4.2 操作流程

- 当 CFGR1 寄存器中 CONT = 1 或 DISCEN = 1 时（禁止设置 CONT = 1 和 DISCEN = 1），仅使用通道 0 时，配置 SCANDIR = 1。

4.3 代码示例

ADC 连续模式 (CONT = 1, DISCEN = 0) , 仅使用通道 0:

```
void ADC_Init()
{
    AdcHandle.Instance                = ADC1;

    if (HAL_ADC_DeInit(&AdcHandle) != HAL_OK) while(1);

    AdcHandle.Init.ClockPrescaler      = ADC_CLOCK_SYNC_PCLK_DIV2;
    AdcHandle.Init.Resolution          = ADC_RESOLUTION_12B;
    AdcHandle.Init.DataAlign           = ADC_DATAALIGN_RIGHT;
    AdcHandle.Init.ScanConvMode         = ADC_SCAN_DIRECTION_BACKWARD;
    AdcHandle.Init.EOCSelection         = ADC_EOC_SINGLE_CONV;
    AdcHandle.Init.LowPowerAutoPowerOff = DISABLE;
    AdcHandle.Init.LowPowerAutoWait    = ENABLE;
    AdcHandle.Init.ContinuousConvMode   = ENABLE;
    AdcHandle.Init.DiscontinuousConvMode = DISABLE;
    AdcHandle.Init.ExternalTrigConv     = ADC_SOFTWARE_START;
    AdcHandle.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    AdcHandle.Init.DMAContinuousRequests = DISABLE;
    AdcHandle.Init.Overrun              = ADC_OVR_DATA_OVERWRITTEN;
    AdcHandle.Init.SamplingTimeCommon   = ADC_SAMPLETIME_3CYCLES_5;

    if (HAL_ADC_Init(&AdcHandle) != HAL_OK) while(1);
}
```

ADC 不连续模式 (CONT = 0, DISCEN = 1) , 仅使用通道 0:

```
void ADC_Init()
```

```
{  
    AdcHandle.Instance                = ADC1;  
  
    if (HAL_ADC_DeInit(&AdcHandle) != HAL_OK) while(1);  
  
    AdcHandle.Init.ClockPrescaler     = ADC_CLOCK_SYNC_PCLK_DIV2;  
    AdcHandle.Init.Resolution         = ADC_RESOLUTION_12B;  
    AdcHandle.Init.DataAlign          = ADC_DATAALIGN_RIGHT;  
    AdcHandle.Init.ScanConvMode        = ADC_SCAN_DIRECTION_BACKWARD;  
    AdcHandle.Init.EOCSelection        = ADC_EOC_SINGLE_CONV;  
    AdcHandle.Init.LowPowerAutoPowerOff = DISABLE;  
    AdcHandle.Init.LowPowerAutoWait    = ENABLE;  
    AdcHandle.Init.ContinuousConvMode   = DISABLE;  
    AdcHandle.Init.DiscontinuousConvMode = ENABLE;  
    AdcHandle.Init.ExternalTrigConv    = ADC_SOFTWARE_START;  
    AdcHandle.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;  
    AdcHandle.Init.DMAContinuousRequests = DISABLE;  
    AdcHandle.Init.Overrun             = ADC_OVR_DATA_OVERWRITTEN;  
    AdcHandle.Init.SamplingTimeCommon  = ADC_SAMPLETIME_3CYCLES_5;  
  
    if (HAL_ADC_Init(&AdcHandle) != HAL_OK) while(1);  
}
```


5 ADC 单次模式

5.1 注意事项

- ADC 在单次模式下，转换结束后，需重新使能 ADC 模块 (ADC_EN = 1)，才能开始下一次转换 (ADC_EN 置 1 到 ADSTART 置 1，时间间隔应大于 8 个 ADC 时钟)。

5.2 操作流程

- 初始化 ADC 模块为单次模式；
- 每次启动转换前配置 ADC_EN = 1；
- 配置 ADSTART = 1；
- 开始转换。

5.3 代码示例

Step1: ADC 初始化

```
void ADC_Init()
{
    AdcHandle.Instance          = ADC1;

    if (HAL_ADC_DeInit(&AdcHandle) != HAL_OK) while(1);

    AdcHandle.Init.ClockPrescaler    = ADC_CLOCK_SYNC_PCLK_DIV2;
    AdcHandle.Init.Resolution        = ADC_RESOLUTION_12B;
    AdcHandle.Init.DataAlign         = ADC_DATAALIGN_RIGHT;
    AdcHandle.Init.ScanConvMode      = ADC_SCAN_DIRECTION_BACKWARD;
    AdcHandle.Init.EOCSelection       = ADC_EOC_SINGLE_CONV;
    AdcHandle.Init.LowPowerAutoPowerOff = DISABLE;
    AdcHandle.Init.LowPowerAutoWait   = ENABLE;
    AdcHandle.Init.ContinuousConvMode = DISABLE;
    AdcHandle.Init.DiscontinuousConvMode = DISABLE;
    AdcHandle.Init.ExternalTrigConv   = ADC_SOFTWARE_START;
    AdcHandle.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    AdcHandle.Init.DMAContinuousRequests = DISABLE;
    AdcHandle.Init.Overrun            = ADC_OVR_DATA_OVERWRITTEN;
    AdcHandle.Init.SamplingTimeCommon = ADC_SAMPLETIME_3CYCLES_5;

    if (HAL_ADC_Init(&AdcHandle) != HAL_OK) while(1);
}
```

Step2: 软件启动

```
void software_trgmode()
{
    AdcHandle.Instance->CR |= ADC_CR_ADEN;
    HAL_Delay(1);
    AdcHandle.Instance->CR |= ADC_CR_ADSTART;
}
```

Step3: ADC 单次模式转换完整流程

```
_HAL_RCC_ADC_FORCE_RESET();
_HAL_RCC_ADC_RELEASE_RESET(); //ADC 复位
ADC_Init();//ADC 初始化
ADC_Cha_SW();//ADC 通道初始化
if(HAL_ADCEX_Calibration_Start(&AdcHandle) != HAL_OK) while(1);//ADC 校准
HAL_ADC_Start(&AdcHandle);//启动 ADC 转换
while(cnt--)//获取 cnt 个数据
{
    software_trgmode();
    if (HAL_ADC_PollForConversion(&AdcHandle,1000) != HAL_OK) while(1);
    *AdcBuff = HAL_ADC_GetValue(&AdcHandle);//数据保存到 AdcBuff 指定的地址中
    AdcBuff++;
}
```

6 ADC 如何利用 VREFINT (1.2V) 测量目标电压值

6.1 注意事项

- ADC 不能直接使用 VREFINT 测量目标电压值，可使用 VREFINT 测量 VCC 值，从而计算出目标电压值。

6.2 操作流程

- 以测量通道 0 的电压值为例；
- 先读出内部参考电压的 ADC 测量结果，记为 ADvrefint，再读出通道 0 的 ADC 测量结果，记为 ADch0；
- 要测量的电压为 $V_{ch0} = VREFINT * (ADch0 / ADvrefint)$ ，VREFINT 为 1.2V。

6.3 代码示例

```
float T_VCC;
uint16_t aADCxConvertedData[32];
AdcBuff = aADCxConvertedData;

__HAL_RCC_ADC_FORCE_RESET();
__HAL_RCC_ADC_RELEASE_RESET();//ADC 复位
ADC_Init();//ADC 初始化
ADC_Cha_SW();//ADC 通道初始化，开启通道 0 和通道 VREFINT
if(HAL_ADCEx_Calibration_Start(&AdcHandle) != HAL_OK) while(1);//ADC 校准
if (HAL_ADC_Start(&AdcHandle) != HAL_OK) while(1);//启动 ADC
i = 2;
while(i--)
{
    if (HAL_ADC_PollForConversion(&AdcHandle,1000) != HAL_OK) while(1);
    *AdcBuff = HAL_ADC_GetValue(&AdcHandle);
    AdcBuff++;
}
T_VCC = (aADCxConvertedData[0]*1.2)/aADCxConvertedData[1];
```

7 ADC 口配置注意事项

7.1 注意事项

- 所有 ADC 口输入电压不能超过 VCC，即使不配置 ADC 功能也不能超过 VCC。

PUYA CONFIDENTIAL

8 版本历史

版本	日期	更新记录
V0.1	2021.10.15	初版
V1.0	2022.06.22	增加 6. ADC 如何利用 VREFINT (1.2V) 测量目标电压值
V1.1	2023.08.16	增加 002A 内容
V1.2	2023.08.24	更新声明



Puya Semiconductor Co., Ltd.

声 明

普冉半导体(上海)股份有限公司（以下简称：“Puya”）保留更改、纠正、增强、修改 Puya 产品和/或本文档的权利，恕不另行通知。用户可在下单前获取产品的最新相关信息。

Puya 产品是依据订单时的销售条款和条件进行销售的。

用户对 Puya 产品的选择和使用承担全责，同时若用于其自己或指定第三方产品上的，Puya 不提供服务支持且不对此类产品承担任何责任。

Puya 在此不授予任何知识产权的明示或暗示方式许可。

Puya 产品的转售，若其条款与此处规定不一致，Puya 对此类产品的任何保修承诺无效。

任何带有 Puya 或 Puya 标识的图形或字样是普冉的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代并替换先前版本中的信息。

普冉半导体(上海)股份有限公司 - 保留所有权利