PHY62XX ADC 应用指南

Phyplus Confidential

Phyplus Inc.

2018/6/13

版本控制信息

版本/状态	作者	参与者	起止日期	备注
V0.2	付晓亮		6/13/2018	文档初稿

Phyplus Confidential

目录

1 简介 4
1.1 模式选择 4
1.2 采集精度 5
1.3 Buffer 缓冲机制 5
1.4 采集速率 5
1.5 工作方式 5
1.5.1 中断方式 5
1.5.2 轮询方式 5
2 API 6
2.1 枚举&宏 6
2.1.1 adc_CH_t 6
2.1.2 ADC 事件 6
2.1.3 adc_CLOCK_SEL_t 7
2.2 数据结构 7
2.2.1 adc_Cfg_t 7
2.2.2 adc_Evt_t 7
2.2.3 adc_Hdl_t 8
2.2.4 adc_Ctx_t 8
2.3 API 8
2.3.1 void hal_adc_init(void) 8
2.3.2 int hal_adc_config_channel (adc_CH_t channel, adc_Cfg_t cfg, adc_Hdl_t evt_handler) 8
2.3.3 int hal_adc_clock_config(adc_CLOCK_SEL_t clk); 9
2.3.4 int hal_adc_start(void) 9
2.3.5 int hal_adc_stop(void) 9
2.3.6 float hal_adc_value(uint16_t* buf, uint8_t size, bool high_resol, bool diff_mode) 10
2.3.7 void __attribute__((weak)) hal_ADC_IRQHandler(void) 10
2.3.8 float hal_adc_value_cal(uint16_t* buf, uint8_t size, bool high_resol, bool diff_mode) 10
3 软件应用 11
3.1 中断模式 11
3.2 轮询模式 13
4 外围电路 16
4.1 被测电压小于等于芯片工作电压 AVDD33 16
4.2 被测电压大于芯片工作电压 AVDD33 16

图表

图表 1 ADC 通道和 GPIO 对应关系 4

图表 2: NORMAL ADC 不同模式下采集精度 5

图表 3: 当被测点电压大于 AVDD33 时外围需要分压电路 16

Phyplus confidential

1 简介

本文档介绍了 PHY62XX ADC 模块的原理和使用方法。

PHY62XX ADC 一共有 9 个 ADC 通道：1 个 PGA，1 个 Temp Sensor，6 个 Normal ADC，1 个 Voice。

本文档主要介绍 6 个 Normal ADC 的使用方法。

ADC 通道和 GPIO 对应关系如下图：

Channel No	Channel Feature	GPIO	Comments
ADC_CH0			PGA
ADC_CH1			Temp Sensor
ADC_CH2	1P/1DIFF	P12	Normal ADC
ADC_CH3	1N	P11	
ADC_CH4	2P/2DIFF	P14	
ADC_CH5	2N	P13	
ADC_CH6	3P/3DIFF	P20	
ADC_CH7	3N	P15	
ADC_VOICE			Voice

图表 1 ADC 通道和 GPIO 对应关系

1.1 模式选择

6 个 Normal ADC 通道有两个工作模式供选择：

- 单端模式：ADC_CH2~ADC_CH7 每个通道都可以独立工作，采集其引脚上的单端电压。
- 差分模式：ADC_CH2~ADC_CH3、ADC_CH4~ADC_CH5、ADC_CH6~ADC_CH7 要成对出现，采集的电压是 ADC_CH2、ADC_CH4、ADC_CH6 上相对于 ADC_CH3、ADC_CH5、ADC_CH7 上的差分电压。

6 个 Normal ADC 通道有两个测试量程供选择：

- bypass mode：测试电压范围为【0V，1V】。
- attenuation mode：测试电压范围为【0V，4V】。

1.2 采集精度

不同模式下，Normal ADC 的采集精度也不相同，详见下表：

	单端模式(unit:V)	差分模式(unit:V)
Bypass	ADC_code/4096	ADC_code/2048-1
Attenuation	(ADC_code/4096)*4	(ADC_code/2048-1)*4

图表 2: Normal ADC 不同模式下采集精度

1.3 Buffer 缓冲机制

Normal ADC 是 12bit 长度的模数转换器，硬件为每个 ADC 通路提供了 32 个 word 长度的连续 buffer，用来缓存 ADC 采集结果，每一个 word 保存 2 次 ADC 转换结果。

1.4 采集速率

Normal ADC 支持采样速率有：80K，160K，320K，默认速率为 320K。

1.5 工作方式

Normal ADC 有两种工作方式：中断方式和轮询方式。

1.5.1 中断方式

9 个 ADC 通道共用一个中断入口，中断号为：CM0（29）。

每个 Normal ADC 通道的中断都能单独 mask 和 clear。

中断触发条件为 buffer 满，即数据填满整块 memory 会触发中断。

中断方式软件处理流程如下，很多流程已经被封装到了 API 中，直接调用使用即可。

1. system initial
2. ADC initial
3. ADC enable
4. irq enable
5. enable ADC interrupt
6. wait interrupt
7. Collect Data
8. calculate ADC value
9. mask interrupt
10. clear interrupt
11. disable ADC

1.5.2 轮询方式

轮询方式下软件处理流程：

1. system initial

Phyplus Confidential

2. ADC initial
3. ADC enable
4. wait a few us
5. Collect Data
6. calculate ADC value
7. disable ADC

2 API

ADC 驱动提供异步 AD 采集功能，采集完成之后通过回调函数回传 ADC 采集结果。

2.1 枚举&宏

2.1.1 adc_CH_t

ADC 物理通道。

ADC_CH0	暂不支持此通道。
ADC_CH1	暂不支持此通道。
ADC_CH1P 或 ADC_CH1DIFF	单端模式下中作为通道 1P，独立工作。 差分模式下作为 1DIFF 通道，和 1N 通道组合使用。
ADC_CH1N	单端模式下作为通道 1N，差分模式下无效。
ADC_CH2P 或 ADC_CH2DIFF	单端模式下中作为通道 2P，独立工作。 差分模式下作为 2DIFF 通道，和 2N 通道组合使用。
ADC_CH2N	单端模式下作为通道 2N，差分模式下无效。
ADC_CH3P 或 ADC_CH3DIFF	单端模式下中作为通道 3P，独立工作。 差分模式下作为 3DIFF 通道，和 3N 通道组合使用。
ADC_CH3N	单端模式下作为通道 3N，差分模式下无效。
ADC_CH_VOICE	语音通道，采集模拟麦克风使用。

2.1.2 ADC 事件

ADC 事件，会在 ADC 驱动的回调函数抛出。

HAL_ADC_EVT_DATA	ADC 采样数据，如果采样数据就绪，会调用已注册的 ADC

Phyplus confidential

	回调函数，送出该事件。
HAL_ADC_EVT_FAIL	ADC 采样失败。

2.1.3 adc_CLOCK_SEL_t

ADC 速率设置。

HAL_ADC_CLOCK_80K	采样速率 80K。
HAL_ADC_CLOCK_160K	采样速率 160K。
HAL_ADC_CLOCK_320K	采样速率 320K。

2.2 数据结构

2.2.1 adc_Cfg_t

ADC 配置参数。

bool	is_continue_mode	是否连续采集模式，如果为 TRUE，ADC 采集在手动停止之前会一直工作。
bool	is_differential_mode	是否为差分模式，如果是差分模式，需要 P 端和 N 端成对配置。 采集的是 P 端电压。
bool	is_high_resolution	是否采用衰减，如果该值为 TRUE，则采用衰减，量程为 0V~1V，否则量程为 0V~4V。
bool	is_auto_mode	暂不支持此功能。

2.2.2 adc_Evt_t

ADC 驱动事件的数据结构。

int	type	ADC 事件类型。 HAL_ADC_EVT_DATA：采样成功，数据有效。 HAL_ADC_EVT_FAIL：采样失败，数据无效。
adc_CH_t	ch	ADC 通道。 通道范围见 adc_CH_t。

Phyplus Confidential

uint16_t*	data	ADC 采样数据指针入口。
uint8_t	size	ADC 采样数据个数。

2.2.3 adc_Hdl_t

ADC 回调函数类型。

typedef void (*adc_Hdl_t)(adc_Evt_t* pev)

2.2.4 adc_Ctx_t

ADC 模块配置。

bool	enable	ADC 模块使能标志。
bool	is_continue_mode	ADC 是否连续工作模式。
bool	is_differential_mode	ADC 通道是否差分工作模式。
adc_Hdl_t	evt_handler	ADC 采集完成回调函数。

2.3 API

2.3.1 void hal_adc_init(void)

ADC 模块初始化，模块其他函数需要配置之后方可使用，否则无法预测结果，或者返回错误。

- 参数

无。

- 返回值

无。

2.3.2 int hal_adc_config_channel (adc_CH_t channel, adc_Cfg_t cfg, adc_Hdl_t evt_handler)

配置 ADC 采集通道。

- 参数

类型	参数名	说明
adc_CH_t	channel	ADC 通道。
adc_Cfg_t	cfg	ADC 配置信息。
adc_Hdl_t	evt_handler	事件回调函数。

Phyplus Confidential

- 返回值

PPlus_SUCCESS	成功。
其他数值	参考<error.h>

2.3.3 int hal_adc_clock_config(adc_CLOCK_SEL_t clk);

配置 ADC 模块的采样频率，在 ADC 启动之前调用。

- 参数

类型	参数名	说明
adc_CLOCK_SEL_t	clk	ADC 采样频率选择，可以选择为 80K、160K、320K。默认值为 320K。

- 返回值

PPlus_SUCCESS	成功。
其他数值	参考<error.h>

2.3.4 int hal_adc_start(void)

开始采集。

- 参数

无。

- 返回值

PPlus_SUCCESS	成功。
其他数值	参考<error.h>

2.3.5 int hal_adc_stop(void)

停止采集。

- 参数

无。

- 返回值

PPlus_SUCCESS	成功。
其他数值	参考<error.h>

Phyplus Confidential

2.3.6 float hal_adc_value(uint16_t* buf, uint8_t size, bool high_resol, bool diff_mode)

计算 ADC 数值，输出为浮点数，为采样点的电压值，并且是输入 buffer 的算术平均值。

- 参数

类型	参数名	说明
uint16_t*	buf	ADC 原始采样数据。
uint8_t	size	ADC 采样数据个数。
bool	high_resol	是否使用衰减（TRUE:量程为 0~1V；FALSE:量程为 0~4V）。
bool	diff_mode	采样数据是否为差分数据。

- 返回值

float	采样点的电压值。

2.3.7 void __attribute__((weak)) hal_ADC_IRQHandler(void)

ADC 中断处理函数。

- 参数

无。

- 返回值

无。

2.3.8 float hal_adc_value_cal(uint16_t* buf, uint8_t size, bool high_resol, bool diff_mode)

计算 ADC 数值，输出为浮点数，为采样点的电压值，并且是输入 buffer 的算术平均值。

预留接口暂时可以忽略。

- 参数

类型	参数名	说明
uint16_t*	buf	ADC 原始采样数据。
uint8_t	size	ADC 采样数据个数。
bool	high_resol	是否使用衰减（TRUE:量程为 0~1V；FALSE:量程为 0~4V）。
bool	diff_mode	采样数据是否为差分数据。

- 返回值

Phyplus Confidential

float	采样点的电压值。

3 软件应用

3.1 中断模式

以下为 Normal ADC 工作在中断、单端、attenuation mode 的示例代码：

测试参考硬件：PHY6200_32_V1.4，ADC 采集通道 P14。 测试参考软件：peripheral\adc，在该示例上修改测试。

```
//ADC初始化
static void adcMeasureTask( void )
{
  int ret;
  adc_Cfg_t cfg = {
      .is_continue_mode = FALSE,
      .is_differential_mode = FALSE, //单端模式
      .is_high_resolution = FALSE,//不衰减
      .is_auto_mode = FALSE,
    };

  ret = hal_adc_config_channel(ADC_CH2P, cfg, adc_evt);//通道ADC_CH2P
  if(ret)
    return;

  hal_adc_start();
  osal_start_timerEx( adcDemo_TaskID, adcMeasureTask_EVT, 2000 );
}
```

```
//ADC采集完成回调函数
static void adc_evt(adc_Evt_t* pev)
{
  if(pev->type == HAL_ADC_EVT_DATA){
    float value = hal_adc_value(pev->data, pev->size, FALSE, FALSE);//
不衰减
    LOG("batt_measure_evt %d\n",(int)(value*1000));
  }
}
```

```
//每2000ms触发一次ADC采集
```

Phyplus Confidential

```
uint16 adc_ProcessEvent( uint8 task_id, uint16 events )
{
  ......
  if ( events & adcMeasureTask_EVT )
  {
    // Perform periodic heart rate task
    LOG("adcMeasureTask_EVT\n");
    adcMeasureTask();
    return (events ^ adcMeasureTask_EVT);
  }
   ......
}
```

测试结果:

```
UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
ADC IRQ
CH4
batt_measure_evt 0//CH4接GND时，ADC测得电压为0V

UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
ADC IRQ
CH4
batt_measure_evt 14

UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
ADC IRQ
CH4
batt_measure_evt 4

UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
ADC IRQ
CH4
batt_measure_evt 3324//CH4接3.3V时，ADC测得电压为3.324V

UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
ADC IRQ
CH4
batt_measure_evt 3322
```

Phyplus Confidential

```
UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
ADC IRQ
CH4
batt_measure_evt 3323
```

3.2 轮询模式

以下为 Normal ADC 工作在轮询、单端、attenuation mode 的示例代码:

测试参考硬件：PHY6200_32_V1.4，ADC 采集通道 P14。

测试参考软件：peripheral\adc，在该示例上修改测试。

轮询方式，CPU 效率低，使用的场景少。

```
//通道配置
int hal_adc_config_channel(adc_CH_t channel, adc_Cfg_t cfg, adc_Hdl_t
evt_handler)
{
//  if(mAdc_Ctx.enable){//注释仅仅为了演示Normal ADC轮询使用方式
//    return PPlus_ERR_BUSY;
//  }

//  if(evt_handler == NULL) //注释仅仅为了演示Normal ADC轮询使用方式

//    return PPlus_ERR_INVALID_PARAM;
  ……
}
```

```
//ADC启动时关闭ADC中断使能，以轮询方式读取ADC采集结果
int hal_adc_start(void)
{
……
//  //ADC_IRQ_ENABLE;
//  NVIC_EnableIRQ((IRQn_Type)ADCC_IRQ); //注释仅仅为了演示Normal ADC轮询
使用方式

//  //ENABLE_ADC_INT;
//  AP_ADCC->intr_mask = 0x1ff; //注释仅仅为了演示Normal ADC轮询使用方式
……
}
```

PhyPlus Confidential

```
//ADC初始化
static void adcMeasureTask( void )
{
  int ret;
  adc_Cfg_t cfg = {
      .is_continue_mode = FALSE,
      .is_differential_mode = FALSE,//单端模式
      .is_high_resolution = FALSE,// 不衰减
      .is_auto_mode = FALSE,
    };

  ret = hal_adc_config_channel(ADC_CH2P, cfg, NULL);//ADC通道为CH2P
  if(ret)
    return;

  hal_adc_start();//启动ADC模式
  osal_start_timerEx( adcDemo_TaskID, adcMeasureTask_EVT, 2000 );
}
```

```
//每2000ms读取一次ADC采集结果
uint16 adc_ProcessEvent( uint8 task_id, uint16 events )
{
    ……
    int ch;
    int n;
    uint16_t adc_data[MAX_ADC_SAMPLE_SIZE];
    float value;
    LOG("adc_ProcessEvent: 0x%x\n",events);

......

  if ( events & adcMeasureTask_EVT )
  {
    // Perform periodic heart rate task
    LOG("adcMeasureTask_EVT\n");
    ch = ADC_CH2P;

    for (n = 0; n < (MAX_ADC_SAMPLE_SIZE-1); n++)
    {
    adc_data[n] = (uint16_t)(read_reg(ADC_CH_BASE + (ch * 0x80) + ((n+1)
* 4))&0xfff);
    }
```

Phyplus Confidential

```
    //for (n = 0; n < (MAX_ADC_SAMPLE_SIZE-1); n++)
    //{
    //LOG("%d %d \n",n,adc_data[n]);
    //}

    value = hal_adc_value(adc_data, MAX_ADC_SAMPLE_SIZE-1, FALSE,
FALSE);
    LOG("batt_measure_evt %d\n",(int)(value*1000));
    adcMeasureTask();

    return (events ^ adcMeasureTask_EVT);
  }

  // Discard unknown events
  return 0;
}
```

测试结果:

```
UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
batt_measure_evt 3323

UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
batt_measure_evt 3322

UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
batt_measure_evt 3323//CH4接3.3V时，ADC测得电压为3.323V

UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
batt_measure_evt 6

UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
batt_measure_evt 0//CH4接GND时，ADC测得电压为0V

UART RX : adc_ProcessEvent: 0x80
adcMeasureTask_EVT
batt_measure_evt 0

UART RX : adc_ProcessEvent: 0x80
```

Vivis Confidential

```
adcMeasureTask_EVT
batt_measure_evt 0
```

4 外围电路

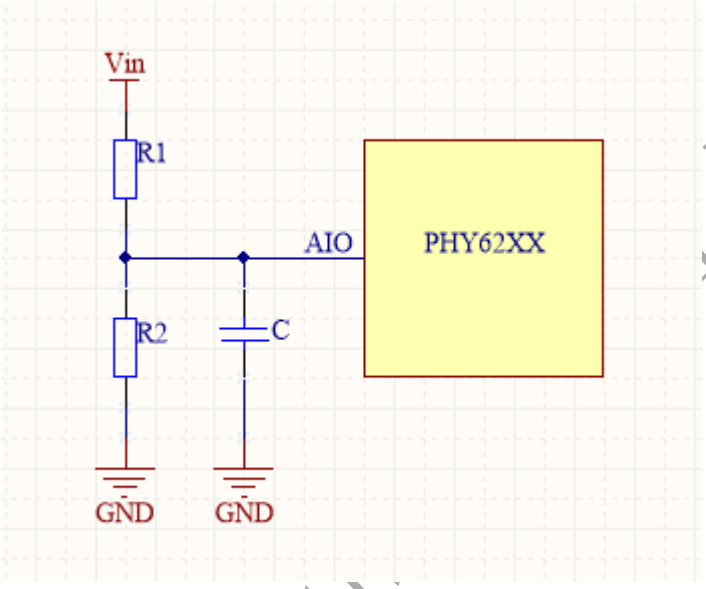
在实际设计 ADC 外围电路中，需要注意被测电压的范围。

4.1 被测电压小于等于芯片工作电压 AVDD33

被测点外围电路不需要分压。

4.2 被测电压大于芯片工作电压 AVDD33

被测点外围电路需要分压，如下图：



图表 3: 当被测点电压大于 AVDD33 时外围需要分压电路

- 模式选择 bypass，测试量程为【0V，1V】。
- 检测电压V_{AIO}需要小于 1V 。

计算公式如下:

$$V_{AIO} = \frac{\frac{R2}{R1+R2}}{1 + jw\frac{R1R2}{R1+R2}C} Vin$$

1、 Vin 检测频率 $f_{in} < \frac{1}{2\pi\frac{R1R2}{R1+R2}C}$

2、 增益$\text{Gain} = \frac{R2}{R1+R2}$

3、 Vin 驱动使能 R1//R2//C

Phyplus Confidential