# [CC2541蓝牙学习——ADC](http://www.cnblogs.com/chenzhao207/p/4539197.html)

CC2541的ADC支持多达14位的模拟数字转换与高达12位的有效位数。它包括一个模拟多路转换器，具有多达8个各自可独立配置的通道，一个参考电压发生器。转换结果通过DMA写入存储器。还具有若干运行模式。

**ADC主要特性如下：**

1. 可选的抽取率，设置了7~12位的分辨率；
2. 8个独立输入通道，可接受单端或差分信号；
3. 参考电压可选为内部，外部单端，外部差分，或AVDD5；
4. 产生中断请求；
5. 转换结束时的DMA触发；
6. 温度传感器输入；
7. 电池测量功能。

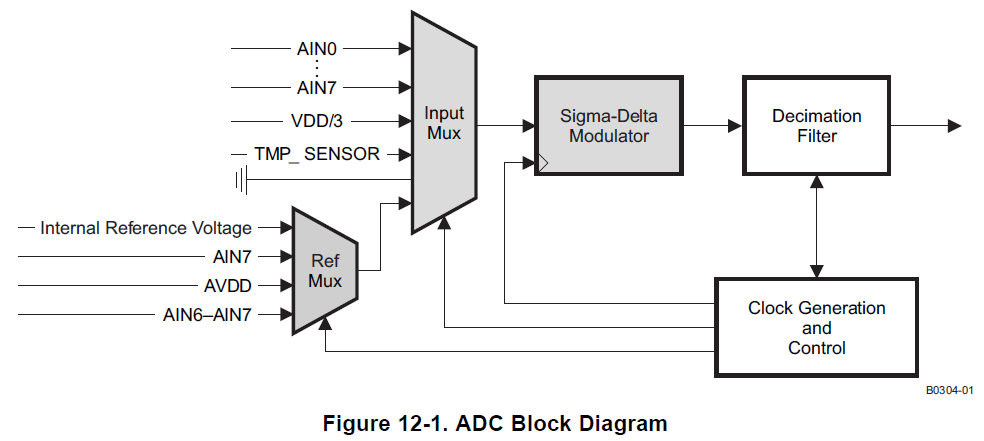


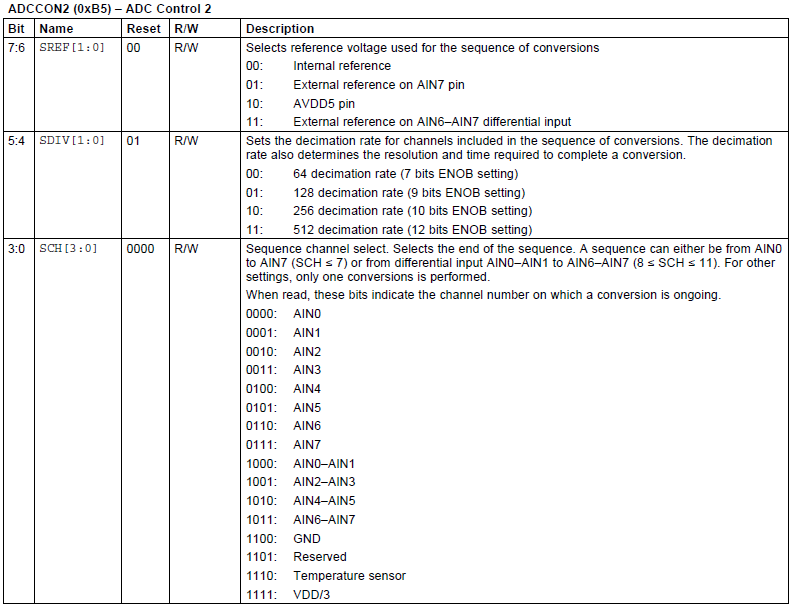
                                                                    图1

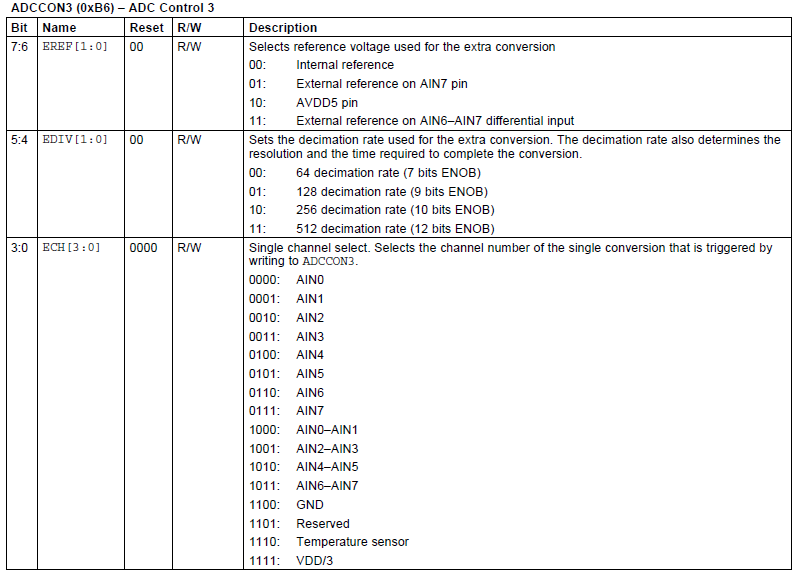
P0引脚上的信号可以作为ADC输入来使用。在下面，这些引脚叫做AIN0—AIN7引脚，输入脚AIN0—AIN7与ADC连接。

输入脚可配置成单端或差动输入。如选择差动输入，包含成对输入AIN0-AIN1，AIN2-AIN3，AIN4-AIN5和AIN6-AIN7；注意这些引脚既不能加载负电压，也不能加载大于VDD的电压。

除了输入脚AIN0-AIN7外，片上的温度传感器也可以用来作为ADC温度测量的输入。如要实现这个功能，需设置寄存器TR0.ADCTM和ATEST.ATESTCTRL。

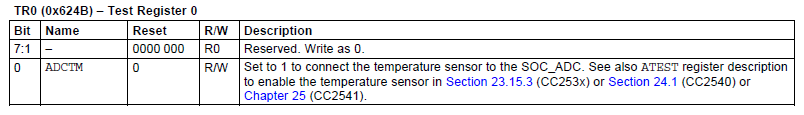
单端输入AIN0至AIN7可代表通道号0至7，通道号8至11分别代表差动输入AIN0-AIN1，AIN2-AIN3，AIN4-AIN5，AIN6-AIN7；通道12表示GND，通道13表示温度传感器，通道15表示AVDD5/3。这些值在ADCCON2.SCH和ADCCON3.SCH中设置。

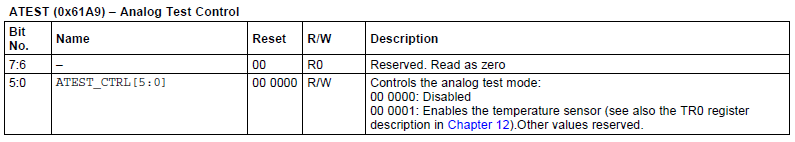




我们看到ADCCON2和ADCCON3这两个寄存器的定义基本相同，但是用法不同，ADCCON2用于ADC序列转换的配置，而ADCCON3则用于单个ADC通道的配置。所谓ADC序列就是多个ADC通道按照次序分别转换。注意：不是同时转换的，从图1我们也可以看出，ADC的模拟输入接一个选择器，同一时刻只能选择一个通道接入进行ADC转换。

如果选择片上的温度传感器作为ADC温度测量的输入，则需要通过配置寄存器TR0和ATEST来获得片上温度，不过这个温度测量误差很大，我们一般不用，这里也就不给出例程了。





启用片内温度采集配置寄存器：

1 TR0 |= 0x01;

2 ATEST |= 0x01;

**1、ADC序列转换**

ADC序列转换无需CPU的参与，ADC能够完成一个序列的转换，并通过DMA把结果写入内存。

寄存器APCFG影响转换序列，来自I/O引脚的8位模拟输入不一定是程序设置的模拟输入。如某一通道是序列的一部分，但在APCFG中相应模拟输入是禁止的，那此通道将被跳过。当使用差动输入时，两个输入脚在APCFG寄存器中必须被设置成模拟输入。

ADCCON2.SCH用来定义ADC输入的转换序列。如ADCCON2.SCH被设为小于8，转换序列包含一个通道（从0到ADCCON2.SCH中设置的通道号），当ADCCON2.SCH值设为8至12时，序列是差动输入，从通道8至程序设置的通道号；当大于12时，序列包含只选择的通道。

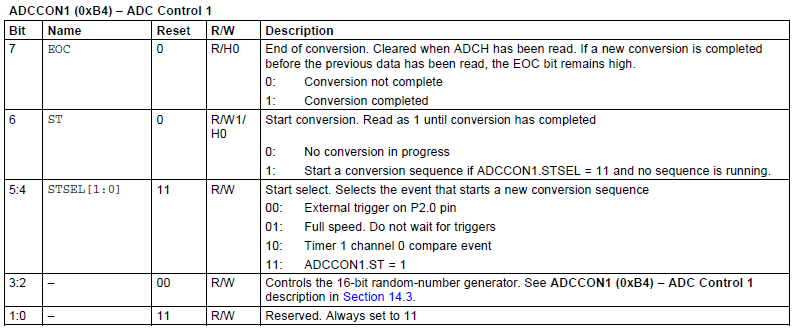
**2、单个ADC转换**

除了序列转换外，ADC可以通过编程执行单个转换。通过写入ADCCON3寄存器可以触发一个转换，转换立即启动，除非一个转换序列正在进行中，这种情况下，当序列完成后，马上执行单个转换。

**3、寄存器ADCCON1**

ADC的数字转换结果可以通过寄存器ADCCON1获得，寄存器ADCCON1的定义如下图所示。

* ADCCON1.EOC：转换结束状态位，当转换结束时设高电平，当读取ADCH时低电平。
* ADCCON1.ST位用来启动序列转换的，当这位设高电平、ADCCON1.STSEL是11且当前无转换运行时序列启动开始。当序列转换结束时，这位自动清除为低电平。
* ADCCON1.STSEL位用来选择哪个事件将启动一个新的序列转换。此项选择有：外部引脚P2.0上升沿事件，之前序列的结束事件，定时器通道0比较事件，或ADCCON1.ST设1事件。



**4、ADC转换结果**

数字转换结果以2进制补码形式表示的，最高位是符号位。

对于单端输入配置，由于ADC输入不能接负电压，转换结果总是正的当输入信号等于参考电压VREF时达到最大转换结果。

对于差分输入配置，ADC输入电压为两个引脚的电压之差，两脚的输入信号不同，结果可能是负的；当采样率为512，模拟输入Vconv=VREF时，12MSB的数字转换结果为2047，当模拟输入等于-VREF时，转换结果为-2048。

通过读ADCCON2.SCH位，知道正在转换的是哪个通道，在序列转换中，ADCL和ADCH中的结果是前一个通道ADC转换的值。如转换序列已结束，ADCCON2.SCH将有一个大于最后通道数一个以上的值，但如最后写入ADCCON2.SCH中的通道数是12或更大，读回的是相同的值。

**5、ADC参考电压**

模数转换的参考电压可选择于内部产生电压，AVDD5脚电压，应用于AIN7输入脚的外部电压，或应用于AIN6-AIN7输入的差动电压。内部参考电压对于CC2541来说是**1.25V**，比较小，能转换的最大模拟电压最大也只能是1.25V，AVDD5脚电压一般为3.3V，精度也不是很高。转换结果的准确度依靠于参考电压的稳定性和噪声度，所以对于要求较高的ADC转换建议从AIN7输入脚接入高精度的参考电压。

**6、ADC转换时间**

ADC只能运行于32MHZ XOSC。执行一个转换的时间依靠于被选择的采样率，一般上，转换时间由以下公式所得：

Tconv=(decimation rate+16)\*0.25us.

可见分辨率越高，转换时间越长。

**7、ADC中断**

只有单通道ADC转换才有ADC中断，序列ADC转换没有ADC中断。

The ADC generates an interrupt when a single conversion triggered by writing to ADCCON3 has completed.No interrupt is generated when a conversion from the sequence is completed.

**8、ADC DMA触发**

每完成一个序列转换，ADC将产生一个DMA触发。单独转换完成不产生DMA触发。

在ADCCON2.SCH中设置8个通道，每个通道都有一个DMA触发。当通道转换中准备好一个采样时，将激活一个DMA触发。DMA触发命名为ADC\_CHsd，s是单端通道，d是差动通道。

另外，当ADC序列转换通道中准备好一个新数据时，一个DMA触发（ADC\_CHALL）将激活。

**单个ADC转换读取ADC值的程序如下：**

[复制代码](javascript:void(0);)

1 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 \*函 数 名：InitADC

3 \*功 能：ADC初始化

4 \*入口参数：参考电压 reference、转换通道 channel、分辨率resolution

5 \*出口参数：ADC转换结果

6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

7 uint Read\_advalue(uchar reference, uchar channel, uchar resolution)

8 {

9 uint value;

10 uchar tmpADCCON3 = ADCCON3;

11

12 APCFG |= 1 << channel ; //设置ADC输入通道,模拟I/O使能

13

14 ADCCON3 = (reference | resolution | channel);

15 ADCIF = 0; //

16

17 while(!ADCIF); //等待 AD 转换完成

18 value = ADCL >> 2; //ADCL 寄存器低 2 位无效

19 value |= ((uint)ADCH << 6); //连接AD转换结果高位和低位

20

21 //根据分辨率获得ADC转换结果有效位

22 switch(resolution)

23 {

24 case ADC\_7\_BIT: value >>= 7;break;

25 case ADC\_9\_BIT: value >>= 5;break;

26 case ADC\_10\_BIT: value >>= 4;break;

27 case ADC\_12\_BIT: value >>= 2;break;

28 default:;

29 }

30

31 ADCCON3 = tmpADCCON3;

32 return (value);

33 }

[复制代码](javascript:void(0);)

**主程序：采集VDD值。**

[复制代码](javascript:void(0);)

1 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 \*程序入口函数

3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

4 int main(void)

5 {

6 uint vddvalue; //ADC转换值

7

8 InitClock(); //32MHz时钟

9 InitUART(); //UART0串口初始化

10

11 while(1)

12 {

13 //ADC参考电压AVDD5引脚电源电压：3.3V，分辨率12位，采集通道：VDD/3,VDD=3.3V

14 vddvalue = Read\_advalue(ADC\_REF\_AVDD5, 0x0f, ADC\_12\_BIT);

15 vddvalue = (vddvalue\*33) >> 11;

16 vddvalue = vddvalue\*3;

17 buf[0] = vddvalue/10 + '0';

18 buf[1] = '.';

19 buf[2] =vddvalue%10 + '0';

20

21 UartSendString(buf,strlen(buf)); //串口上传采样VDD值

22 Delay1ms(2000); //每隔2s上传一次值

23 }

24 }

[复制代码](javascript:void(0);)

**这里给出协议栈的adc转换函数参照对比。**

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

1 #include "hal\_adc.h"

2 uint16 u16cvalu=HalAdcRead(HAL\_ADC\_CHANNEL\_4,HAL\_ADC\_RESOLUTION\_12);

3 分辨率设置为12位时，从源码可以看出，可用位是ADCH 8位+ADCH高4位，其中ADCH最高位是符号位，所以有11位的分辨率，0-2047

4 默认基准电压3.3V

5 uint16 HalAdcRead (uint8 channel, uint8 resolution)

6 {

7 int16 reading = 0;

8

9 #if (HAL\_ADC == TRUE)

10

11 uint8 i, resbits;

12 uint8 adctemp;

13 volatile uint8 tmp;

14 uint8 adcChannel = 1;

15 uint8 reference;

16

17 /\* store the previously set reference voltage selection \*/

18 reference = ADCCON3 & HAL\_ADC\_REF\_BITS;

19

20 /\*

21 \* If Analog input channel is AIN0..AIN7, make sure corresponing P0 I/O pin is enabled. The code

22 \* does NOT disable the pin at the end of this function. I think it is better to leave the pin

23 \* enabled because the results will be more accurate. Because of the inherent capacitance on the

24 \* pin, it takes time for the voltage on the pin to charge up to its steady-state level. If

25 \* HalAdcRead() has to turn on the pin for every conversion, the results may show a lower voltage

26 \* than actuality because the pin did not have time to fully charge.

27 \*/

28 if (channel < 8)

29 {

30 for (i=0; i < channel; i++)

31 {

32 adcChannel <<= 1;

33 }

34 }

35

36 /\* Enable channel \*/

37 ADCCFG |= adcChannel;

38

39 /\* Convert resolution to decimation rate \*/

40 switch (resolution)

41 {

42 case HAL\_ADC\_RESOLUTION\_8:

43 resbits = HAL\_ADC\_DEC\_064;

44 break;

45 case HAL\_ADC\_RESOLUTION\_10:

46 resbits = HAL\_ADC\_DEC\_128;

47 break;

48 case HAL\_ADC\_RESOLUTION\_12:

49 resbits = HAL\_ADC\_DEC\_256;

50 break;

51 case HAL\_ADC\_RESOLUTION\_14:

52 default:

53 resbits = HAL\_ADC\_DEC\_512;

54 break;

55 }

56

57 /\* read ADCL,ADCH to clear EOC \*/

58 tmp = ADCL;

59 tmp = ADCH;

60

61 /\* Setup Sample \*/

62 adctemp = ADCCON3;

63 adctemp &= ~(HAL\_ADC\_CHN\_BITS | HAL\_ADC\_DEC\_BITS | HAL\_ADC\_REF\_BITS);

64 adctemp |= channel | resbits | (reference);

65

66 /\* writing to this register starts the extra conversion \*/

67 ADCCON3 = adctemp;

68

69 /\* Wait for the conversion to be done \*/

70 while (!(ADCCON1 & HAL\_ADC\_EOC));

71

72 /\* Disable channel after done conversion \*/

73 ADCCFG &= (adcChannel ^ 0xFF);

74

75 /\* Read the result \*/

76 reading = (int16) (ADCL);

77 reading |= (int16) (ADCH << 8);

78

79 /\* Treat small negative as 0 \*/

80 if (reading < 0)

81 reading = 0;

82

83 switch (resolution)

84 {

85 case HAL\_ADC\_RESOLUTION\_8:

86 reading >>= 8;

87 break;

88 case HAL\_ADC\_RESOLUTION\_10:

89 reading >>= 6;

90 break;

91 case HAL\_ADC\_RESOLUTION\_12:

92 reading >>= 4;

93 break;

94 case HAL\_ADC\_RESOLUTION\_14:

95 default:

96 reading >>= 2;

97 break;

98 }

99 #else

100 // unused arguments

101 (void) channel;

102 (void) resolution;

103 #endif

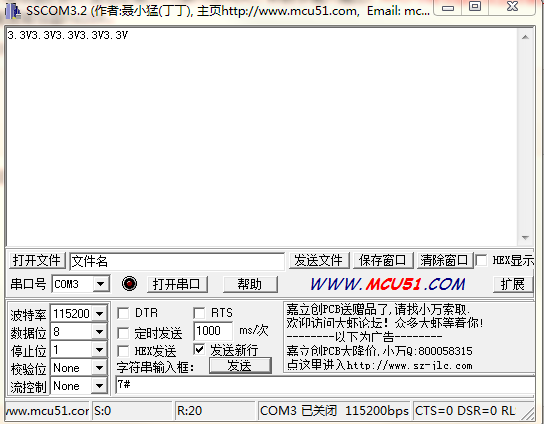
104

105 return ((uint16)reading);

106 }

View Code

**调试结果：显示VDD值3.3V。**



**关于程序注意以下几点：**

1、要配置一个端口0脚为一个ADC输入，APCFG寄存器中相应的位必须设置为1。这个寄存器的默认值选择端口0引脚为非ADC，即数字输入输出。APCFG寄存器的设置将覆盖P0SEL的设置，所以无需再配置P0SEL，另外对于I/O口作为外设功能，都无需配置方向，即无需配置寄存器PxDIR。

2、对于单次ADC转换的配置，只需要配置寄存器ADCCON3，无需配置寄存器ADCCON1和ADCCON2。对于判断转换是否结束，还有一种判断方法：

1 ADCCON1 |=0X30; //ADC启动方式选择为ADCCON1.ST=1事件

2 ADCCON1 |= 0x40; //启动转换

3 while(!(ADCCON1 & 0x80)); //等待 AD 转换完成

ADCCON1.STSEL是用于启动转换序列的触发方式的，对于单次ADC转换，个人感觉这样配置不好，以后对于单次ADC转换，不采用这种判断方式。

单次转换判断是否转换结束：判断ADC中断标志ADCIF。

3、ADCH的最高位是符号位，对于单次测量，结果总是正的，所以符号位总是0。14位的ADC转换值有效值并不是14位的。

有效分辨率如下：  
00: 64 decimation rate (7 bits ENOB)----ADCH低7位  
01: 128 decimation rate (9 bits ENOB)---ADCH低7位+ADCH高2位  
10: 256 decimation rate (10 bits ENOB)--ADCH低7位+ADCH高3位  
11: 512 decimation rate (12 bits ENOB)--ADCH低7位+ADCL高5位

例如：采集VDD/3值时，使用12位分辨率，参考电压AVDD5:3.3V

VDD/3 = vddvalue\*3.3/2^11  
扩大10倍

VDD/3 = vddvalue\*33/2^11  
为什么是除以2^11而不是2^12,因为最高位是符号位，12位分辨率实际上只有11位。

VDD = (vddvalue\*33/2^11) \* 3

4、差分输入可以用来做比较器。比如通道ADCCON3.ECH=1000，对应差分输入AIN0-AIN1。如果要比较一个模拟信号和另一个模拟信号的大小关系，只需要将这两个信号分别接入AIN0和AIN1，然后判断ADCH的最高位，如果是1，则AIN0<AIN1,如果是0，则AIN0>=AIN1。

5、最大转换电压等于参考电压，而参考电压的选择不能大于芯片的电源电压，一般为3.3V。虽然差分输入可以转换负电压，但是每一个模拟输入引脚都必须是正电压且小于电源电压VDD，负电压是指两个输入通道的差值。