This thread has been locked.
If you have a related question, please click the "Ask a related question" button in the top right corner. The newly created question will be automatically linked to this question.

CC2541: Battery Indicator

sadasivam arumu... Intellectual 940 points
Community Member

Part Number: CC2541
Other Parts Discussed in Thread: CC2540

Hi,

I am using the battery service with following input settings:

ADCCON2_SREF_P0_7(External reference-3.32v)

ADCCON2_SDIV_256(10 bit ADC)

HAL_ADC_CHN_AIN0(Analog input - P0.0)

The percent calculation is errorred. Pls find the following calc:

batt_max = 464(3.0v)

batt_min = 402(2.6v)

Analog input supplied externally = 3.0v

The Percent result should be 100.

But i received 92%. how to reach 100%.

over 5 years ago

sadasivam arumugam over 5 years ago
Intellectual 940 points

// ADC voltage levels
#define BATT_ADC_LEVEL_3V 464 //3v with ref voltage of 3.3v
#define BATT_ADC_LEVEL_2V 402 //2.6v with ref voltage of 3.3v

#define ADCCON2_SREF_P0_7 (0x01 << 6) // External reference on AIN7 pin

```
static uint8 battServiceAdcCh = HAL_ADC_CHN_AIN0;

//battery measure function:

static uint8 battMeasure( void )
{
uint16 adc;
uint8 percent;

/**
* Battery level conversion from ADC to a percentage:
*
* The maximum ADC value for the battery voltage level is 511 for a
* 10-bit conversion. The ADC value is references vs. 1.25v and
* this maximum value corresponds to a voltage of 3.75v.
*
* For a coin cell battery 3.0v = 100%. The minimum operating
* voltage of the CC2540 is 2.0v so 2.0v = 0%.
*
* To convert a voltage to an ADC value use:
*
* (v/3)/1.25 * 511 = adc
*
* 3.0v = 409 ADC
* 2.0v = 273 ADC
*
* We need to map ADC values from 409-273 to 100%-0%.
*
* Normalize the ADC values to zero:
*
* 409 - 273 = 136
*
* And convert ADC range to percentage range:
*
* percent/adc = 100/136 = 25/34
*
* Resulting in the final equation, with round:
*
* percent = ((adc - 273) * 25) + 33 / 34
*/

// Call measurement setup callback
if (battServiceSetupCB != NULL)
{
battServiceSetupCB();
}


// Set [APCFG.APCFG0 = 1].
APCFG |= APCFG_APCFG0;


/*****************************************************************************
* ADC configuration:
* - [ADCCON1.ST] triggered
* - 12 bit resolution
* - Single-ended
* - Single-channel, due to only 1 pin is selected in the APCFG register
```

```
* - Reference voltage is VDD on AVDD pin
*/

// Set [ADCCON1.STSEL] according to ADC configuration.
//ADCCON1 = (ADCCON1 & ~ADCCON1_STSEL) | ADCCON1_STSEL_ST; // ADC trigger selection-external trigger
ADCCON1 = (ADCCON1 & ~ADCCON1_STSEL) | ADCCON1_STSEL_ST; // ADC trigger selection-
ADCON1.ST = 1;
ADCCON2 = ADCCON2_SREF_P0_7| ADCCON2_SDIV_256 | battServiceAdcCh; // ADCCON2_SREF_P0_7 = msp_vcc
// battServiceAdcCh = HAL_ADC_CHN_VDD3


/**************************************************************************
* ADC conversion :
* The ADC conversion is triggered by setting [ADCCON1.ST = 1].
* The CPU will then poll [ADCCON1.EOC] until the conversion is completed.
*/

// Set [ADCCON1.ST] and await completion (ADCCON1.EOC = 1).
ADCCON1 |= ADCCON1_ST;
while( !(ADCCON1 & ADCCON1_EOC));

/* Store the ADC result from the ADCH/L register to the adc_result variable.
* The conversion result resides in the MSB section of the combined ADCH and
* ADCL registers.
*/
//adc_result = (ADCL >> 4);
//adc_result |= (ADCH << 4);

adc = (int16) (ADCL);
adc |= (int16) (ADCH << 8);
adc >>= 6;

// Configure ADC and perform a read
//HalAdcSetReference( HAL_ADC_REF_125V );
//adc = HalAdcRead( battServiceAdcCh, HAL_ADC_RESOLUTION_10 );

// Call measurement teardown callback
if (battServiceTeardownCB != NULL)
{
battServiceTeardownCB();
}

/*if (adc >= battMaxLevel)
{
percent = 100;
}
else if (adc <= battMinLevel)
{
percent = 0;
}
else*/
{
if (battServiceCalcCB != NULL)
{
percent = battServiceCalcCB(adc);
}
```

```
else
{
uint16 range = battMaxLevel - battMinLevel + 1; //464-402 = 62+1 = 63

// optional if you want to keep it even, otherwise just take floor of divide
// range += (range & 1);
range >>= 2; // divide by 4 // 0xf; (15)d;

percent = (uint8) ((((adc - battMinLevel) * 25) + (range - 1)) / range); // (((adc - 402)*25)+(14)/15);
}
}

return percent;
}
```

Eirik V *over 5 years ago* TI__Guru 56590 points

Hello Sadasivam,

Please reference these threads:

https://e2e.ti.com/support/wireless-connectivity/bluetooth/f/538/t/901206?CC2541-Battery-Service

https://e2e.ti.com/support/wireless-connectivity/bluetooth/f/538/t/492893

Eirik V *over 5 years ago in reply to sadasivam arumugam* TI__Guru 56590 points

You need to measure and calibrate to the actual values you get in your design.for BATT_ADC_LEVEL_3V and BATT_ADC_LEVEL_2V.

sadasivam arumugam *over 5 years ago in reply to Eirik V* Intellectual 940 points

Have taken the battery status:

Received an analog value of:

1. 3v = 464 adc

2. 2.6 v = 459 adc.

Its not a correct solution. But, if it is getting the above values means, is there any settings to change. Can you verify the code attached in last reply.

Eirik V *over 5 years ago in reply to sadasivam arumugam* TI__Guru 56590 points

Why not use the original battMeasure function from battservice.c and the HalAdc driver that comes with the ble sdk?

You seem to set up external reference (ADCCON2_SREF_P0_7) ?

sadasivam arumugam *over 5 years ago in reply to Eirik V* Intellectual 940 points

I have tried with the same function you mentioned. But it fails and result in same manner.

Yes I am using external reference of 3.3 v

About TI

Quick links

Buying

Connect with us

Texas Instruments has been making progress possible for decades. We are a global semiconductor company that designs, manufactures, tests and sells analog and embedded processing chips. Our products help our customers efficiently manage power, accurately sense and transmit data and provide the core control or processing in their designs.