Using internal VREF in STM32

The main issue with VREF generally is, that there are many "references" involved in related texts, and there's not a single unified nomenclature for them. ST often uses confusing nomenclature and their formulae are sometimes written from an impractical point of view, increasing the confusion. So let's just derive ours.

Let's introduce a convention, where dimensionless units (i.e. ADC readings, although sometimes their units are called "bins") are written in capital letters, and voltages (in, ehm, volts... :-) [1]) in lowercase.

The ADC physically works using voltage between VREF+ and VREF- pins. On most packages, VREF- is internally tied hard to common ground (namely VSSA), and even if it is brought out to pin on the largest packages, it is required to be tied to VSSA externally. VREF+ is brought out on larger packages as separate pin, but on most packages it is connected to VDDA[2].

Let's denote this VREF+/VDDA reference voltage **vrefext**.

If any arbitrary voltage **vmeas** is connected to ADC pin and ADC measurement is taken, with the readout result of **MEAS**, the formula to calculate the voltage from the readout is

vmeas = vrefext * MEAS / MAX (1)

where **MAX** is ADC reading for input voltage equal to reference voltage, i.e. number of ADC bins = *2^resolution - 1*[3]. For most STM32 ADCs are 12-bits so **MAX** = 4095, but note, that in some STM32 models the ADC resolution is adjustable, so it should not be taken as a constant.

Now the problem is, that for this formula to work we need to know the **vrefext** voltage (i.e. voltage at VREF+/VDDA pins), and it's not always the case, e.g. when the power supply is derived from a battery and we don't want to use an external voltage reference chip.

So we have an internal reference in the STM32 (another "reference" to increase the confusion). That's a supposedly stable (i.e. invariant in whole operation temperature range, supply voltage range, time) "bandgap" reference; but its voltage is subject to semiconductor manufacturing process variations, that's why there is a factory calibration for it (see below). In many STM32 it cannot be used directly as the reference for the ADC (which, as said above, is tied to VREF+ or VREF+/VDDA pins)[4], so if we don't have a precise enough external reference on VREF+ (or VREF+/VDDA) with known-to-us voltage, we can utilize this "bandgap" reference to indirectly establish VREF+/VDDA voltage by measuring the "bandgap" reference output and then use the internal calibration to calculate the VREF+/VDDA voltage.

So, let's denote the actual voltage of the internal "bandgap" reference **vrefint**, let's connect it to the ADC (by setting the appropriate register controlling ADC MUX), and let's take an ADC readout, **REFINT.** Then, applying and reversing eq.(1), we can calculate ADC's external reference **vrefext** as

vrefext = vrefint * MAX / REFINT (2)

This assumes **vrefint** is a known value, and while there is a typical value for it in the datasheet and we can use it as it is, for increased precision, we want to know its voltage as precisely as possible, as it's actual value is process dependent. ST did a measurement for us when manufacturing the chip, using a precise external reference voltage on the VREF+ pin, and stored the resulting ADC reading into the system memory, let's denote it ***CAL*** (it's denoted in different ways in different STM32 datasheets, e.g. VREFINT for 'L476, VREFINT_CAL for 'F042, or VREFIN_CAL for 'F4xx). The voltage used as external reference at calibration, let's denote it ***vcal*****,** depends on the particular STM32 model, usually one of 3.0V or 3.3V, see the same spot in datasheet where the address of calibration value is given. Again, applying eq.(1) for the moment when the calibration was taken at the factory

vrefint = *vcal* * *CAL* / MAX (3)

Now we simply substitute (3) into (2) and (2) into (1) and we get

vmeas = *vcal* * MEAS / MAX * *CAL* / REFINT (4)

where ***vcal*** is the voltage at which factory calibration was performed, taken from datasheet (3.0V or 3.3V); **MEAS** is the ADC reading taken for our actual input voltage; **MAX** is the number of ADC bins (i.e. 2^resolution - 1); ***CAL*** is the calibration value read out of system memory**; REFINT** is the ADC reading taken for the internal voltage reference[5].

Note, that for actually performing calculation in the mcu, possible overflow and rounding have to be taken into account. Resorting to *float* or *double* somewhat alleviates this problem (while making assessment of actual impact of the arithmetics to be more complicated), but it's impractical e.g. in Cortex-M0/M0+/M3 - based STM32 models which don't have floating-point unit and would need to perform the calculation in software, consuming time and memory.

When keeping calculations in integer arithmetics, to decrease impact of rounding, divisions should go after multiplications, so we'd multiply ***vcal* * MEAS * CAL** first. We can arbitrarily chose voltage units so that we avoid 32-bit overflow - for 12-bit ADC, **MEAS** is max. 4095 so it's 12 bits, and as **CAL** is at around middle of the full range, that's 11 bits. That leaves us with 9 bits for the 3.3V range, so we'd probably chose 10mV as the base voltage unit (this of course depends also on the particular application). That means, that for ***vcal*** we would use 300 or 330 depending on the STM32 model, and resulting **vmeas** would be in 10mV units.

As an added step for slightly better output at the cost of some additional calculation, we can properly round by adding half of the divisor before performing the division, i.e.

vmeas = (*vcal* * MEAS * *CAL* + *(*REFINT * MAX / 2)) / (REFINT * MAX)

For higher resolution, 64-bit integer arithmetics (or, as mentioned, floating-point arithmetics) would need to be chosen.

Internal reference and internal temperature sensor

STM32 incorporate a simple internal temperature sensor, which can be switched to one of the ADC input channels in the ADC MUX. This sensor outputs a voltage which is fairly linearly proportional to the internal temperature of the chip (i.e. not the ambient temperature), within 1°C-2°C in the entire working temperature range. However, the parameters of the sensor again are rather process dependent, so ST includes on most models (except on the "value lines", i.e. models which number end in 0) two calibration values, which again are the ADC readings taken from the temperature sensor at two given temperatures (usually at 30°C and 110°C, let's denote them ***t1*** and ***t2***) using the same precision voltage reference connected to VREF+ than with the internal reference calibration. These ADC readings (let's denote them ***TEMP1***/***TEMP2***) are again stored in the system memory, and the addresses are given in the datasheet for every STM32 model as TS_CAL1/TS_CAL2, together with the VREF+/VDDA voltages at which they were taken (let's denote it ***vtempcal***).

As the relationship between voltage and temperature is linear

temperature = A * voltage + B (5)

and we are given two points, for any voltage **vtemp** (corresponding to ADC reading **TEMP**) the temperature is

temperature = ***t1*** + (***t2*** - ***t1***) * (**vtemp** - ***vtemp1***) / (***vtemp2*** - ***vtemp1***) (6)

where ***vtemp1*** and ***vtemp2*** are the sensor's voltages corresponding to temperatures ***t1*** and ***t2***, and are thus according to (1)

vtemp1 = ***vtempcal* * *TEMP1* / MAX** (7)

***vtemp2* = *vtempcal* * *TEMP2* / MAX** (8)

and for the actual temperature measurement, according to (4)

vtemp = *vcal* * TEMP / MAX * *CAL* / REFINT (9)

Substituting (7) to (9) into (6), and considering that the factory calibrations were taken at the same VREF+/VDDA reference voltage (see datasheets) i.e. ***vcal*** = ***vtempcal***, we can eliminate ***vcal*****/MAX**, obtaining the resulting formula

temperature =

***t1* + (*t2* - *t1*) * (TEMP*CAL/REFINT - *TEMP1*) / (*TEMP2* - *TEMP1*)** (10)

where ***t1*** and ***t2*** are the temperatures at which calibration values were taken (i.e. usually ***t1*** = 30°C and ***t2*** = 110°C, **TEMP** is the ADC reading taken for our actual temperature; ***CAL*** is the internal reference calibration value read out of system memory (from VREFINT_CAL or similarly sounding address, see text above); **REFINT** is the ADC reading taken for the internal voltage reference; ***TEMP1***/***TEMP2*** are the temperature calibration values read out from system memory (from TS_CAL1/TS_CAL2 addresses).

We can again perform the calculation in floating point, or keeping things integer, we would reorganize the formula as

temperature = t1 + (t2 - t1) * (TEMP * CAL - TEMP1 * REFINT) / (REFINT * (TEMP2 - TEMP1))

Given TEMP * CAL is roughly 23 bits and (t2 - t1) is 7 bits, this formula should not overflow in 32-bit arithmetics. Caution should be exercised though, observing rules for integer promotions in C, as signed variables have to be used, given the result may be negative.

In case, when **vrefext** (i.e. voltage on VDDA/VREF+) is known (e.g. derived from a precise external voltage reference chip) and it's not desired to derive it from the internal reference, from (1)

vtemp = vrefext * TEMP / MAX (10)

We substite this together with (7) and (8) into (6), and, again eliminating ***vtempcal*****/MAX**, we obtain

temperature =
***t1* + (*t2* - *t1*) * (vrefext/*vtempcal*TEMP - *TEMP1*) / (*TEMP2* - *TEMP1*)** (11)

Note, that both the temperature sensor's output, and the ADC itself, are relatively noisy, as they work in close vicinity of heavy digital circuitry, often sharing ground, so there are several paths how digital noise impacts the analog measurements. Without any special consideration, it's not uncommon to see ADC noise in the range of several tens of bins, i.e. 1/100 of the range, i.e. a few tens of mV. As the temperature sensor's slope is typically 2.5 mV/°C, it's not uncommon to see the raw calculated temperature to vary within cca 10°C, which is quite a lot.

So, for reasonable temperature readings, it is vital not only to maintain all usual precautions for good ADC readings (i.e. careful ground layout, heavy VDDA/VREF+filtering, reducing digital activity (e.g. going to sleep during ADC conversions) etc.); but also some of the usual filtering techniques (e.g. averaging) should be considered.

---

1 This may sound funny in English, but in many languages, the expression for "voltage" (which is often akin to "potential" or "tension") does not have the name of the unit, i.e. "volt", in it. In fact, English is the exception rather than rule in this.

2 In the older STM32 families (such as 'F1, 'L1, 'F2, upper-end 'F4, 'F7) generally, the separate VREF+ pin starts to appear at package sizes having 100 pins and above (there may be a few exceptions, mainly in BGA-style packages at around 60 pins). The 'F0 family has no model with separate VREF+ pin. Generally, the smallest packages with separate VREF+ pin start at 48 pins - from the "non-G" families, only 'F401Cx comes in QFN48 with separate VREF+, and the same 'F401Cx and some 'L0xx models in BGA-style (Wafer-Level, WLxxx). However, in the 'G0 and 'G4 families, there are plenty of models starting at 48 pins with separate VREF+, even in QFP packages.

3 Many texts use in this place *2^resolution* instead of *2^resolution - 1,*i.e. for 12-bit ADC they divide the readout by 4096 rather than by 4095. Which one is more accurate?

Strictly speaking, none of these; but as we'll show, it does not really matter.

4096 would appear more appropriate as there are indeed 4096 "bins"; however, that would never result in output voltage equal to V_{ref}, so that speaks in favour of 4095.

Let's for brevity denote *resolution* as N. In fact, the ADC splits the whole V_{ref} range into 2^N bins, so the K-th bin corresponds to voltages between $K*V_{ref}/2^N$ and $(K+1)*V_{ref}/2^N$, so it is appropriate to take middle of that interval and write that ADC readout K corresponds to $(2K+1)/2*V_{ref}/2^N$ with error of $\pm V_{ref}/(2*2^N)$.

The simplified formula which still does output V_{ref} at the high end is $K*V_{ref}/(2^N-1)$ The difference between this and the "exact" formula above is $V_{ref}*(2^N-1-2K)/(2*2^N*(2^N-1))$. This is a linear function of K, crossing zero at $K=(2^N-1)/2$ thus for the practical range of K having absolute maxima at both K=0 and $K=2^N-1$, both being $V_{ref}/(2*2^N)$ in value. Note, that that's exactly what's the uncertainty of the "exact" formula above is, that's why the difference does not really matter, and we can use the simplified formula safely.

One more, and maybe more valid, reason, why the simplified formula can be used, is, that all practical ADCs have inaccuracies (nonlinearities) higher than one bin, i.e. higher than $V_{ref}/2^N$.

4 In some STM32 models (e.g. 'L4, 'H7, 'G0, 'G4), there's a VREFBUF module which allows to output the internal voltage reference to the VREF+ pin (in packages where it's not tied to VDDA internally). If the precision of the VREFBUF output given in datasheet is considered appropriate, eq.(1) could be simply used; however, the described calibration process against the internal reference still can be utilized if it yields higher measurement precision.

5 It may come as a surprise, that **vrefext** i.e. the actual VREF+/VDDA voltage, is not present in this formula. The reason for that is, that both the actual input volrage measurement and the internal reference measurement are ratiometric i.e. relative to **vrefext**, and putting them into ratio we indeed eliminate **vrefext**. However, as the two measurements are not taken in the same moment of time, for this all to work properly, it must be ensured, that **vrefext**, while unknown in value, remains stable.