A/D převodník a bargraf

1 Zadání

- Nakonfigurujte ADC na vývojové desce pro čtení údaje z trimru R2. Použijte driver sedmisegmentového displeje, na displej vypisujte pozici trimru přepočítanou na rozsah 0-500. **(0,5b)**. Upravte driver tak, aby z LED nad displejem vytvořil osmistupňový bargraf. Průběh ukazatele bude lineární, musí umožňovat zobrazení od minimální do maximální hodnoty **(0,5b)**.
- Pro hodnotu čtenou z ADC použijte filtraci pomocí exponenciální kumulace. Koeficient Q volte cca 12 **(0,5b)**.
- Přidejte mezi kanály ADC také interní referenci a interní teplotní čidlo. Po stisku S1 zobrazte na displeji vypočtené napájecí napětí AVCC s přesností na dvě desetinná místa. Po stisku S2 zobrazte teplotu mikrokontroléru ve °C **(1b)**. Po uvolnění tlačítka zůstane údaj zobrazený po 1 sekundu, poté se vrátí zpět na hodnotu trimru R2 **(0,5b)**.
- Hodnocena bude i úprava zdrojového kódu, zejména odsazování bloků.

2 Návod

2.1 Základní seznámení

- Vytvořte si pracovní kopii svého repozitáře z Githubu (Git Clone), příp. aktualizujte repozitář ze serveru (Git Pull).
- Založte nový projekt přes File / New / STM32 Project / Board Selector / NUCLEO-F030R8. Budeme využívat HAL knihovny, proto ponechte Targeted Project Type na STM32Cube. Potvrďte inicializaci všech periferií do výchozího nastavení.

2.2 Konfigurace ADC a zobrazení

- V CubeMX aktivujte příslušný vstup ADC (IN0). Zvolte kontinuální režim převodu (Continuous Conversion Mode) a maximální dobu vzorkování (Sampling Time = 239.5 Cycles). Dále je vhodné zvolit přepisování dat při přetečení (Overrun data overwritten, nastává během ladění). Povolte ADC global interrupt.
- Dále nastavte výstupní GPIO piny pro LED driver (PB3 = SCT_CLK, PB4 = SCT_SDI, PB5 = SCT_NLA, PB10 = SCT_NOE). Nezapomeňte na inicializaci LED driveru pomocí **`sct_init()`**.
- Ještě před spuštěním AD převodníku je velice vhodné jednorázově spustit jeho autokalibraci, která kompenzuje výrobní tolerance. Následně budeme používat režim ADC s voláním přerušení po dokončení převodu::

```
HAL_ADCEx_Calibration_Start(&hadc);
HAL_ADC_Start_IT(&hadc);
```

- Po dokončení každého převodu je zavolán callback, ve kterém si uložte hodnotu z ADC do statické volatilní proměnné:

```
static volatile uint32_t raw_pot;

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    raw_pot = HAL_ADC_GetValue(hadc);
}
```

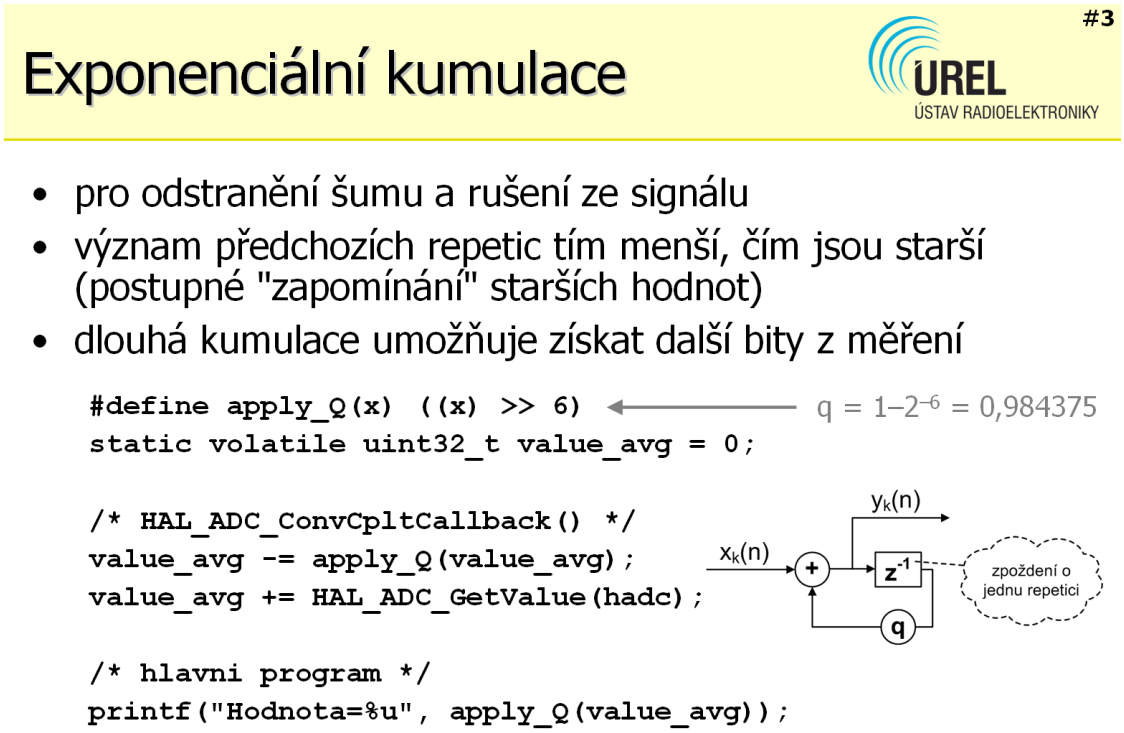
- V hlavním kódu v nekonečné smyčce přepočítávejte získanou 12bitovou hodnotu na rozsah 0 až 500, zobrazte na displeji (funkce **`sct_value()`** z minulého cvičení) a doplňte krátké čekání (50ms).
- Funkci **`sct_value()`** rozšiřte o parametr **`uint8_t led`**, který bude v rozsahu 0-8 udávat počet rozsvícených LED bargrafu. Doporučuji rozšířit tabulku **`reg_values[]`** o další pozici, která bude představovat LED diody, a výsledek vytvářet binárním OR spolu s ostatními segmenty:

```
reg |= reg_values[3][led];
```

- Nápověda k zapojení LED bargrafu: `//----43215678---- @ LED`

2.3 Exponenciální kumulace ADC hodnoty

- Implementujte v callbacku ADC exponenciální kumulaci. Použijte bitový posun (ADC_Q) o cca 12 (tj. dělení 2^{12}).



- Kód callbacku bude obsahovat:

```
static uint32_t avg_pot;

raw_pot = avg_pot >> ADC_Q;
avg_pot -= raw_pot;
avg_pot += HAL_ADC_GetValue(hadc);
```

- Otestujte filtraci kumulací – hodnoty na displeji budou „líné“, měnit se s mírným zpožděním po přenastavení trimru.

2.4 Vícekanálové ADC, interní reference a teplotní čidlo

- V bloku ADC v CubeMX dále aktivujte vstupy Temperature Sensor Channel a Vrefint Channel. Vyberte také rovnou GPIO vstupy pro tlačítka (PC0 = S2, PC1 = S1). U obou vstupů tlačítek aktivujte pull-up.
- Callback AD převodníku je volaný po každém dokončení převodu, všechny tři kanály se neustále opakují. Pro účely rozlišení výsledků si do callbacku definujte statickou proměnnou **`channel`**. Pro kanál 0 zůstane kód exponenciální kumulace hodnoty trimru, kanál 1 představuje surovou hodnotu teploty (**`raw_temp`**), kanál 2 surovou hodnotu napěťové reference (**`raw_volt`**). Pro tyto nové hodnoty definujte příslušné statické volatilní proměnné na úrovni modulu.
- Dokončení převodu sekvence všech tří kanálů lze detekovat pomocí testování **`EOSEQ`** flagu. Je-li nastavený, proměnnou **`channel`** vynulujte, v opačném případě ji inkrementujte:

```
if (__HAL_ADC_GET_FLAG(hadc, ADC_FLAG_EOS)) channel = 0;
else channel++;
```

- Postupy výpočtu skutečného napětí AVCC (tj. AREF) a vnitřní teploty ve °C lze nalézt ve STM32SnippetsF0:

```
/* Temperature sensor calibration value address */
#define TEMP110_CAL_ADDR ((uint16_t*) ((uint32_t) 0x1FFFF7C2))
#define TEMP30_CAL_ADDR  ((uint16_t*) ((uint32_t) 0x1FFFF7B8))

/* Internal voltage reference calibration value address */
#define VREFINT_CAL_ADDR ((uint16_t*) ((uint32_t) 0x1FFFF7BA))

uint32_t voltage = 330 * (*VREFINT_CAL_ADDR) / raw_volt;

int32_t temperature = (raw_temp - (int32_t)(*TEMP30_CAL_ADDR));
temperature = temperature * (int32_t)(110 - 30);
temperature = temperature / (int32_t)(*TEMP110_CAL_ADDR - *TEMP30_CAL_ADDR);
temperature = temperature + 30;
```

- Kód nekonečné smyčky v main() řešte jako jednoduchý stavový automat, např. se stavy:

```
static enum { SHOW_POT, SHOW_VOLT, SHOW_TEMP } state = SHOW_POT;
```

- Podle stavu **`state`** zobrazujte na displeji příslušnou hodnotu. V případě stisku daného tlačítka (čtení pomocí **`HAL_GPIO_ReadPin()`**) změňte stav a nastavte statickou proměnnou udávající aktuální čas. Po uplynutí 1s od uloženého času přejděte zpět do výchozího stavu.
- Postup je podobný jako ve cvičení *ISR, tlačítka se zákmity, časovač SysTick*. SysTick není třeba explicitně inicializovat (řeší HAL knihovny, výchozí perioda 1ms), jeho hodnota je dostupná pomocí volání funkce **`HAL_GetTick()`**.