Laboratorium 6 MARM Przetwarzanie sygnałów analogowych. Przetworniki A/C oraz C/A.

Lucjan Bryndza Politechnika Warszawska

12 grudnia 2019

Celem laboratorium jest zapoznanie z przetwarzaniem sygnałów analogowych za pomocą mikrokontrolerów rodziny STM32, oraz obsługą zintegrowanych przetworników **analogowo - cyfrowych** oraz **cyfrowo - analogowych**.

1. Zadania do wykonania na ćwiczeniach

Ćwiczenia związane z przetwornikiem **A/C** należy zrealizować z zastosowaniem płytki **STM32F411E-DISCO**. Natomiast ćwiczenia z przetwornikiem **C/A** należy zrealizować z wykorzystaniem zestawu **STM32F469I-DISCO**, ponieważ mikrokontroler *STM32F411* nie posiada zintegrowanego przetwornika C/A. Po wykonaniu ćwiczenia należy opracować sprawozdanie, którego sposób opracowania opisano w rozdziale 3.

1.1. Pomiary wartości skutecznej z wykorzystaniem przetwornika A/C

Do realizacji tego ćwiczenia należy wykorzystać zestaw z procesorem STM32F411. Napisz program który:

- Umożliwi pomiar wartości skutecznej napięcia (RMS) w zakresie do $3.3V_{pp}$ podanej na wejście przetwornika A/C
- Jako poziom zerowy przyjmij wartość połowy napięcia V_{dd}
- Jako częstotliwość próbkowania przyjmij częstotliwość **48kHz**
- Do wyznaczania częstotliwości próbkowania przetwornika A/C należy wykorzystać wyzwalanie sprzętowe z wybranego układu czasowo-licznikowego zgodnie z dokumentacją mikrokontrolera [2]
- Jako wejście sygnału należy wybrać odpowiedni port GPIO zgodnie z dokumentacją mikrokontrolera [3]
- Program powinien wyświetlać zmierzoną wartość skuteczną (RMS) za pomocą funkcji *dbprintf*.
- Program powinien za pomocą diod **LD3-LD6** wizualizować poziom napięcia skutecznego (RMS) (Linijka świetlna)

Po opracowaniu programu i sprawdzeniu poprawności działania:

- Na wybrane wejście analogowe podłącz generator sygnałowy
- Wykonaj po 3 pomiary wartości skutecznej napięcia dla przebiegu trójkątnego oraz sinusoidalnego o częstotliwości **1kHz** w zakresie od $100mV_{pp}$ do $3V_{pp}$ z krokiem $250mV_{pp}$
- Zacznij od $100mV_{pp}$ a po każdych 3 pomiarach zwiększaj wartość napięcia o $250mV_{pp}$ aż do osiągnięcia wartości $3V_{pp}$

- Aby wyznaczyć błąd pomiaru zmierz i zapisz wartości napięcia skutecznego korzystając z multimetru wzorcowego z funkcją *TrueRMS*

Uwaga: W generatorze sygnałowym poziom offsetu dla sygnału należy ustawić na połowę wartości napięcia referencyjnego.

1.2. Generowanie sygnału za pomocą przetwornika C/A

Korzystając z zestawu STM32F469I napisz program który z pomocą wbudowanego przetwornika C/A wygeneruje sygnał sinusoidalny o nastepujących parametrach:

- Amplituda: $3V_{pp}$
- Częstotliwość sygnału: 1kHz
- Częstotliwość próbkowania: 48kHz

Przy pisaniu oprogramowania należy przyjąć następujące założenia:

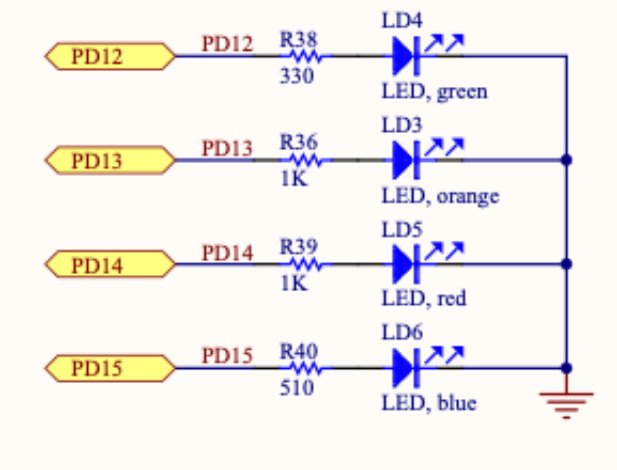
- Sygnał generowany za pomocą przetwornika **DAC1**
- Tablica funkcji sinus umieszczona w pamięci FLASH
- Taktowanie przetwornika C/A za pomocą sprzętowego układu czasowo-licznikowego
- Do przesyłania próbek z pamięci FLASH do przetwornika należy wykorzystać kanał DMA

Po napisaniu oprogramowania należy podłączyć oscyloskop, a następnie obejrzeć i zapisać oscylogramy na wyjściu przetwornika.

2. Materiały pomocnicze

2.1. Diody LED w zestawie STM32F411E-DISCO

Zestaw ewaluacyjny posiada 4 diody LED podłączone do portu **PD** do pinów 12...15.



2.2. Konfiguracja pobranie oraz uruchomienie projektu

Aby pobrać repozytorium możemy się posłużyć komendą, którą należy uruchomić w wierszu polecenia *GIT Bash*:

```
git clone --recursive -b pub/pw/lab6 https://boff.pl/cgit/
  ↪ public/isixsamples/
```

W kolejnym kroku należy zmienić katalog na *isixsamples* oraz pobrać konfigurację dla środowiska *VSCode*:

```
cd isixsamples
curl http://bryndza.boff.pl/downloads/prv/vscodecfg.zip --
  ↪ output vscodecfg.zip
```

Pobrany plik należy rozpakować bezpośrednio do katalogu *isixsamples*.

```
unzip vscodecfg.zip
```

Konfiguracja projektu dla zestawu STM32F411 discovery odbywa się za pomocą polecenia:

```
python waf configure --debug --board=stm32f411e_disco
```

Kompilacje projektu możemy wykonać za pomocą polecenia:

```
python waf
```

Natomiast zaprogramowanie zestawu odbywa się za pomocą polecenia:

```
python waf program
```

Projekt możemy również konfigurować oraz uruchamiać bezpośrednio z *Visual Studio Code*. Najpierw należy w pliku *.vscode/task.json* zmienić argumenty dla polecenia *waf configure*. Następnie za pomocą polecenia **CTRL+P** otwieramy wiersz polecenia i wpisujemy *task waf configure* w kolejny kroku należy zbudować projekt za pomocą polecenia *task waf*, a w kolejnym kroku zaprogramować poleceniem *task waf program*.

Debugowanie programu następuje po wciśnięciu klawisza **F5** wcześniej jednak należy w pliku *.vscode/launch.json* ustawić odpowiednio ścieżkę do uruchamianego programu.

Opis skrótów klawiaturowych dla środowiska **VSCode** możemy znaleźć tutaj: [6].

2.3. Pomiary wartości RMS

Informację na temat pomiarów wielkości skutecznych w domenie cyfrowej z wykorzystaniem przetworników A/C możemy znaleźć w nocie katalogowej firmy *Analog Devices* [4]

2.4. Konfiguracja kontrolera DMA dla przetwornika C/A

Do przesyłania zawartości tablicy do rejestru **DR** przetwornika C/A możemy wykorzystać kontroler **DMA1 strumień 5** oraz **kanał 7**. Konfiguracje kontrolera DMA możemy zrealizować z wykorzystaniem następującego kodu:

```
/*## Configuration of NVIC
  ↪ ##################################################*/
 /* Configure NVIC to enable DMA interruptions */
      isix::set_irq_priority(DMA1_Stream5_IRQn, {1, 7);
      isix::request_irq(DMA1_Stream5_IRQn);
 /*## Configuration of DMA
   ↪ ##################################################*/

 /* Enable the peripheral clock of DMA */
 LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_DMA1);
 /* Configure the DMA transfer */
 /* - DMA transfer in circular mode to have an unlimited DAC
   ↪ signal */
 /* generation. */
 /* - DMA transfer to DAC without address increment. */
 /* - DMA transfer from memory with address increment. */
 /* - DMA transfer to DAC by half-word to match with DAC
   ↪ resolution 12 bits */
 /* - DMA transfer from memory by half-word to match with DAC
   ↪ data */
 /* buffer variable type: half-word.
                                  */
```

```
LL_DMA_SetChannelSelection(DMA1, LL_DMA_STREAM_5,
   ↪ LL_DMA_CHANNEL_7);
LL_DMA_ConfigTransfer(DMA1,
                      LL_DMA_STREAM_5,
                      LL_DMA_DIRECTION_MEMORY_TO_PERIPH |
                      LL_DMA_MODE_CIRCULAR |
                      LL_DMA_PERIPH_NOINCREMENT |
                      LL_DMA_MEMORY_INCREMENT |
                      LL_DMA_PDATAALIGN_HALFWORD |
                      LL_DMA_MDATAALIGN_HALFWORD |
                      LL_DMA_PRIORITY_HIGH );


/* Set DMA transfer addresses of source and destination */
LL_DMA_ConfigAddresses(DMA1,
                       LL_DMA_STREAM_5,
                       (uint32_t)&WaveformSine_12bits_32samples,
                       LL_DAC_DMA_GetRegAddr(DAC1, LL_DAC_CHANNEL_1,
                          ↪
                          ↪ LL_DAC_DMA_REG_DATA_12BITS_RIGHT_ALIGNED
                          ↪ ),
                       LL_DMA_DIRECTION_MEMORY_TO_PERIPH);

/* Set DMA transfer size */
LL_DMA_SetDataLength(DMA1,
                     LL_DMA_STREAM_5,
                     WAVEFORM_SAMPLES_SIZE);

/* Enable DMA transfer interruption: transfer error */
LL_DMA_EnableIT_TE(DMA1,
                   LL_DMA_STREAM_5);

/* Note: In this example, the only DMA interruption activated is
   ↪  */
/* tranfer error. */
/* If needed, DMA interruptions of half of transfer */
/* and transfer complete can be activated. */
/* Refer to DMA examples. */

/*## Activation of DMA
   ↪ ######################################################*/
/* Enable the DMA transfer */
```

```
LL_DMA_EnableStream(DMA1, LL_DMA_STREAM_5);
```

Jedynym przypadkiem, gdy będziemy potrzebować przerwania od kontrolera DMA będzie sytuacja wystąpienia błędu transmisji. Procedura obsługi przerwania od kanału DMA przedstawia się w sposób następujący:

```
extern "C" {
	void dma1_stream5_isr_vector()
	{
		/* Check whether DMA transfer error caused the DMA
		   ↪ interruption */
		 if(LL_DMA_IsActiveFlag_TE7(DMA1) == 1)
		{
		/* Clear flag DMA transfer error */
			LL_DMA_ClearFlag_TE7(DMA1);

			/* Call interruption treatment function */
			DacDmaTransferError_Callback();
		 }
	}
}
```

W momencie wystąpienia błędu transmisji kontroler DMA zgłosi przerwanie i ustawi flagę **TE** informującą o wystąpieniu błędu w wybranym kanale. Korzystając z odpowiedniej funkcji zwrotnej w przypadku wystąpienia błędu możemy podjąć kroki zaradcze.

2.5. Podstawowe API systemu ISIX do ćwiczenia

Podstawową funkcją diagnostyczną służącą do wypisywania danych na domyślny port diagnostyczny (terminal) jest funkcja:

```
#define dbprintf(fmt, ...) fnd::tiny_printf("%s:%d|" fmt "\r\n",
   ↪ _isix_dbglog_extract_basename(__FILE__),__LINE__,##
   ↪ __VA_ARGS__)
```

Funkcja przyjmuje identyczny zestaw argumentów jak standardowa funkcja **printf** pozbawiona jest jedynie formatowania i wyświetlania liczb zmiennoprzecinkowych.

Do utworzenia nowego wątku systemu operacyjnego możemy wykorzystać funkcję:

```
ostask_t isix::task_create(task_func_ptr_t task_func, void *
   ↪ func_param, unsigned long stack_depth, osprio_t priority,
   ↪ unsigned long flags=0);
```

Jako pierwszy argument funkcja przyjmuje funkcję, która będzie wykorzystana do realizacji wątku. Jako argument *func_param* możemy przekazać argument przekazany funkcji realizującej wątek. Parametr *stack_depth* określa wielkość stosu dla danego wątku. Do realizacji zadania powinien wystarczyć stos o wielkości 2kB. Jako parametr *priority* należy przekazać priorytet wątku, przy czym 0 oznacza priorytet najwyższy.

W systemie ISIX wektory przerwań mają inne nazwy niż w przypadku przykładów z biblioteki **LL**. W przypadku układów przerwań AC i CA nazwy wektorów przerwań są następujące:

```
ISR_VECTOR(adc_isr_vector);
ISR_VECTOR(tim6_dac_isr_vector);
```

Należy pamiętać, że w przypadku deklaracji wektorów w języku C++ należy użyć konstrukcji **extern "C"**, a funkcje wektorów przerwań powinny być zadeklarowane jako **void fun(void)**.

3. Instrukcja opracowania sprawozdania

- Opracować wyniki pomiarów napięcia skutecznego zmierzone za pomocą przetwornika A/C
- Określić względny oraz bezwzględny błąd pomiarowy
- Określić niepewność wzorcowania
- Umieścić oscylogramy z sygnału wygenerowanego za pomocą przetwornika C/A. Dlaczego sygnał na wyjściu odbiega od sinusoidalnego?
- Jaki układ analogowy należy zastosować na wejściu przetwornika C/A gdybyśmy chcieli opracować woltomierz True RMS, a których brakło podczas realizacji ćwiczenia?
- Jaki układ analogowy należy zastosować na wyjściu przetwornika C/A jeśli chcielibyśmy opracować układ generatora?
- Opracować wnioski na temat możliwości analogowych mikrokontrolerów rodziny STM32

Literatura

[1] *STM32F411E-DISCOVERY* kit user manual

[2] *STM32F411* reference manual

[3] *STM32F411* datasheet

[4] RMS Calculation for Energy Meter Applications Using the ADE7756

[5] *ISIX-RTOS* API examples

[6] *Visual Studio Code* keyboards shortcuts