JIEE Časopis priemyselnej elektrotechniky
Journal of Industrial Electrical Engineering
ISSN 2454-0900
www.jiee.eu

Time measuring of STM32F446RE microcontroller's A/D conversion

[1]*Patrik JACKO,* [2]*Dobroslav KOVÁČ*

[1,2] Department of Theoretical and Industrial Electrical Engineering, Faculty of Electrical Engineering and Informatics, Technical University of Košice,
Slovak Republic

[1] patrik.jacko.2@tuke.sk, [2] dobroslav.kovac@tuke.sk

***Abstract* — Analog to digital conversion is nowadays very important electronic component because currently all devices operate in digital form. Time of conversion and precision are keys of successful conversions from analog to digital form. We know a lot of various A/D converters, but this paper is focused to Successive-approximation converter, because this model is used in the STM32F446RE microcontrollers. This microcontroller is fast, for that primary objective of this topic is measuring time of conversion one and 10 000 samples.**

***Keywords* — microcontroller, STM32F4, AD converters, successive-approximation converter, time of AD conversion**

I. Introduction

Nowadays there are a lot of various fast electronic devices for controlling other devices, regulation, controlling of motors or other devices, or for analog to digital conversions. We know some microcontrollers such as ATMega64 of Atmel Corporation, or some microcontrollers of Texas Instruments. The best devices for it is a digital signal processor, because it is a very fast microcontroller. Unfortunately, DSP microcontroller of Texas Instruments is expensive, therefore I would like tell you something about a cheaper variant of DSP – microcontroller STM32F446RE which includes DSP set too. In this paper I write something about AD converters of STM32F446RE and there is shown rate test of 12-bit AD converter.

II. ANALOG TO DIGITAL CONVERTERS

Analog to digital converter is an electronic device which is used for converting continuously analog value to the discrete digital form consist of logic levels – high level is logic 1 and low level is presented as logic 0. Each conversion is made in three steps - sampling, quantization and coding. The first step is sampling. The analog signal is first sampled periodically, it means that the AD converter retrieves a sequence of pulses whose amplitude corresponds to the analog signal in close, the time moments. These samples are converted to the digital value by quantization and coding. If the AD converter has more quantization number, AD converter is more accurate [1]. Quantization levels determine the number of bits the converter. For example 12-bit converter, 10-bit converter, etc. 2-bit converter has 4 various quantization levels (22 = 4): 00, 01, 10, and 11. It means that the voltage input range of 0 to 10 volts is divided into 4 values. The first is 0-2,5 V, the second is 2,5-5 V, the third 5-7,5V and the last value is 7,5 to 10 V. Because there is more converting error (2,5V) we can use more bits converter. 3-bit converter has 8 quantization levels (000, 001, 010, 011, 100, 101, 110, 111), 4-bit has 16 quantization levels, etc. Let's say something about 12-bit converter. This converter has maximal value 4096. Unfortunately, 12-bit is only theoretical bit value, but practically it is 11-bit or 10-bit converter because last bits can be various value when there is more conversions. The bits that have always the same value we named effective bites [2].

III. SUCCESSIVE-APPROXIMATION CONVERTER

Successive-approximation converter starts the conversion with MSB (most-significant bit) and ends conversion with LSB (less-significant bit). At the beginning of the transmission cycle is approximation register (SAR) sets on output value 10000000. This value is equal with output to DA converter and it is feedback too. The value of feedback is compared with the comparator with input voltage Vin. If Vin is greater than feedback value, MSB is set to logic 1, otherwise it returns a 0 [3].

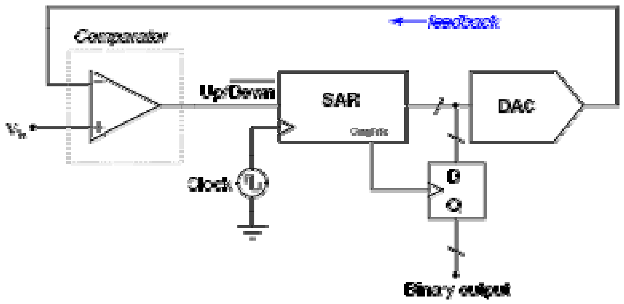


Fig. 1 Successive-approximation converter

AD converters of the STM32F446RE. Microcontroller STM32F446RE is high-performance system based on the ARM family with the Cortex-M4 processor. This microcontroller includes a full set of DSP (digital signal processing) instruction. We can use it for a lot of different applications such as controlling motors, regulations or controlling other devices where the rate is a primary factor. STM32F446RE includes three 12-bit successive-approximation AD converters. They can operate at 12-bit, 10-bit, 8-bit or 6-bit converters and they have up to 19 multiplexed channels allowing it to measure signals from 16 external sources. STM32F446RE's AD converters can be performed in single, continuous, scan or discontinuous mode. It means, that single mode makes only one conversion. Continuous mode starts a new conversion as soon as it finishes one. Scan mode is used for scanning a group of channels and discontinuous converts a short sequence of n conversion, where n ≤ 8.

Microcontroller STM32F446RE is fast device because its operate frequency is up to 180 MHz and it uses fast AHB BUS MATRIX 7S8M bus. This frequency is divided with prescaler to low-speed bus APB1 with 45 MHz and high-speed bus APB2 with 90 MHz. AD converters are connected to the APB2 bus, it means that AD converters operate with 90 MHz [4].

The rate of conversion is very important information. How much samples AD converter makes at some time. We did a test of conversion speed with using 12-bit AD converter in STM32F446RE. We tested time of 10 000 conversions in "online" and "offline" mode. Online mode means that the AD converter makes 10 000 samples, saves them to the memory and after it sent the last sample to the UART line. Offline mode means that microcontroller sent each sample after end of conversion to the UART line. Offline mode is longer than online mode.

IV. CONFIGURATION OF AD CONVERTER

The first step of time measuring AD conversion is correctly configuration of AD converter. Microcontroller STM32F446RE includes three AD converters. There is configured ADC1.

```
60 void adc_single_config()
61 {
62   //ADC1
63   GPIOA->MODER |= GPIO_MODER_MODER1;        // GPIOA pin PA1 as analog
64
65   RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;       // enable clock to the ADC1
66   ADC1->CR2 &= ~ADC_CR2_CONT;               // enabled single mode
67   ADC1->CR2 &= ~ADC_CR2_ALIGN;              //right align
68   ADC1->CR1 |= ADC_CR1_EOCIE;               //interrupt to the end of conversion
69
70   ADC1->CR1 &= ~ADC_CR1_RES;                //12-bit mod
71   ADC1->SMPR2 |= ADC_SMPR2_SMP1;             // 480 cycles
72   ADC1->SQR1 &= ~ADC_SQR1_L;                // one conversion
73   ADC1->SQR3 = 1;                           // channel 1 ADC1_IN1 (PA1)
74   ADC1->CR2 |= ADC_CR2_ADON;                //ennable adc
75
76   NVIC_EnableIRQ(ADC_IRQn);                 //interrupt
77   NVIC_SetPriority(ADC_IRQn,5);
78
79 }
80
```

Fig. 2 Configuration of AD1 converter

In the Fig.2 shows ADC configuration. ADC1 is applied on GPIOA (General-Purpose Input/Output) on the pin 1. The MODER register has available two bits in mode configuration. If there is configured *00*, GPIO pin is configured to input pin, *01* present output pin, *10* is an alternative model for USART, SPI e.g. and combination *11* presents analog mode. *RCC_MODER_MODER1* means *11* bit combination. The second line enabled clock for ADC1 where is used APB2 bus operate to 90MHz.

As there was mentioned, A/D converters can operate in four modes: *Single mode, Continuous mode, Scan mode and Discontinuous mode.* We use single mode, because we need only one sample. It is configured in 66 line *ADC1->CR2 &= ~ADC_CR2_CONT.* For this configuration we use Control Register 2 (CR2) where we set one bit. If this bit is set to logic 0, ADC is configured to the single mode. If it is set to logic 1, converter operates continuously. Right align in the next line presents data alignment. We have configured it to right alignment. The next line of the code says about enabling interrupt of end conversion. After each conversion is set EOC (end of conversion) flag.

Resolution of AD converter is a very important setting. This configuration is in the CR1 register, on 24 and 25 bits concretely. There are four possibilities for resolution: 12 bit AD converter (*00* set bit), 10 bit mode – *01*, 8 bit mode-*10* and 6 bit mode with configuration *11*. The fast of AD conversion is dependent on the resolution and the number of samples which is configured in the next line. Registers SMPR is divided to the SMPR1 and SMPR2 registers. These registers include 3-bits for setting cycles numbers. The command *ADC1->SMPR2 |= ADC_SMPR2_SMP1* presents AD converter 1 references to the SMPR2 register (Sample time Register 2). The right side of the command presents bit, which is configured to the 480 samples and it is set to the channel 1. If we want to set the ADC applied to the channel 5 with 3 cycles, the command will be form: ADC1->SMPR2 |= *ADC_SMPR2_SMP5_0*. There are 8 numbers of cycles: 3, 15, 28, 56, 84, 112, 144 and 480.

Registers SQR1 and SQR3, in the next line, are used for selecting number of conversions and select ADC channel. In this test is selected one conversion and it is applied to the ADC channel 1 (Channel is selected in the SQR3 register).

The last configuration command is enable AD converter with command *ADC1->CR2 |= ADC_CR2_ADON;* There are also two other commands for enabling interrupts and for setting the interrupt priority. [4]

In the next figure shows the source code of time counting AD conversion.


```
60 void adc_single_config()
61 {
62   //ADC1
63   GPIOA->MODER |= GPIO_MODER_MODER1;        // GPIOA pin PA1 as analog
64
65   RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;       // enable clock to the ADC1
66   ADC1->CR2 &= ~ADC_CR2_CONT;               // enabled single mode
67   ADC1->CR2 &= ~ADC_CR2_ALIGN;              //right align
68   ADC1->CR1 |= ADC_CR1_EOCIE;               //interrupt to the end of conversion
69
70   ADC1->CR1 &= ~ADC_CR1_RES;                //12-bit mod
71   ADC1->SMPR2 |= ADC_SMPR2_SMP1;             // 480 cycles
72   ADC1->SQR1 &= ~ADC_SQR1_L;                // one conversion
73   ADC1->SQR3 = 1;                           // channel 1 ADC1_IN1 (PA1)
74   ADC1->CR2 |= ADC_CR2_ADON;                //ennable adc
75
76   NVIC_EnableIRQ(ADC_IRQn);                 //interrupt
77   NVIC_SetPriority(ADC_IRQn,5);
78
79 }
80
```


Fig. 3 Time counting of the AD conversion

If the user pushes the button connected to the GPIOA on the pin 0 which external interrupt EXTI0, timer 2 starts the counting and ADC makes 10000 samples. It is started by command *ADC1->CR2 |=ADC_CR2_SWSTART.* After the conversion each of the samples is stored in the DR register (data register). These values are saved to the integer array *pole[]*. This buffer includes 10 000 values. The *uint32_t sum=sum+pole[i]* command makes the sum of the all samples. If the AD converter makes the last conversion, *i=9999*, program makes an average of all the values and save it to the variable *average.* Concurrently is stopped timer TIM2 and actual counting value of the CNT (counter register) is storage to the variable *time*. If the both variables are completed, functions *sendString* and *sendNumber* sent the text and values of AD average and time to the terminal via USART2.

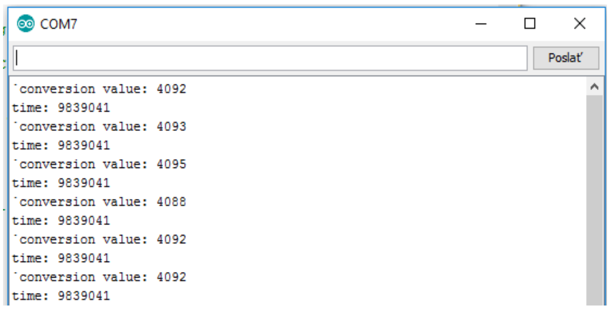
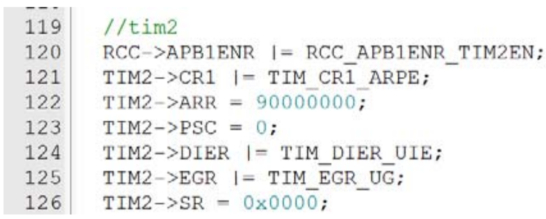


Fig. 4 Results of AD conversion and counting the time with 12-bit converter and 480 cycles of sample conversions

As there is shown, 10 000 conversions were 9839041 clock cycles. It means, that one conversion took 983 clock cycles when an AD converter is configured on the 12-bit resolution and it make 480 sample cycles. If there was only 3 sample cycles, time would be lower – 55 pulses on the one sample.

There is possible calculating the conversion time of the actual value of the timer.

For time calculating we must know timer parameters. In this program was used Timer 2. This timer is connected to the APB1 bus. It means that timer operates to frequency 90MHz. There were mentioned that APB1 has 45MHZ and APB2 90MHZ. For all timers are these frequency values multiple 2x. STM32F446RE includes 14 timers, but only 5 timers operate to the 180MHz. The other timers operate to the 90MHz.



```
//tim2
RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;
TIM2->CR1 |= TIM_CR1_ARPE;
TIM2->ARR = 90000000;
TIM2->PSC = 0;
TIM2->DIER |= TIM_DIER_UIE;
TIM2->EGR |= TIM_EGR_UG;
TIM2->SR = 0x0000;
```

Fig. 5 Timer 2 configuration

The first step presents enable clock for TIM2. The timer is set to Auto-reload preload. It means, that if timer counts to the maximum value, it will be reset and it starts counting yet. This configuration is in the control register 1 (CR1). The ARR (Auto-reload register) register served for setting maximal counting value. The maximal value of this register is 4 294 967 296. It is 32-bit timer. The PSC register is prescaler. This value is set to zero. The DIER register DMA/Interrupt Enable Register. It served for update interrupt. The EGR (Event Generation Register) is set for re-initialize the counter and generates an update of the registers. The last line with a SR register (Status Register) resets all flags. [4]

$$time = \frac{time\_prescaler + 1}{time\_freqency} * timer\_actual\_value; \Rightarrow time = \frac{0+1}{90000000} * 9839041 = 0{,}109 \quad (1)$$

The time of full conversion is 0,109 sec, by using (1). When we want time of one conversion we have to divide 0,109s by 10000, because there is 10000 conversions. The final result is up to 10,932 μs for one conversion. This measurement was applied at the “online” conversions. Offline conversions take much longer, because there is long sending time to the USART2. The same measuring in offline mode takes approximately 5 sec for full conversion – 10000 samples.

The online time can be shorter when we use only 3 ADC cycles defined in SMPR2 register. This method is not precise such as 480 cycles, but it is faster.

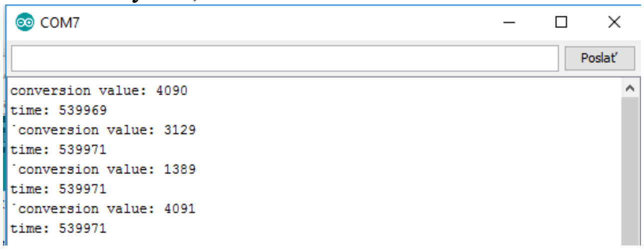


Fig. 6 Results of AD conversion and counting the time with 12-bit converter and 3 cycles of sample conversions

Using 3 sample cycles the time is much shorter than 480 cycles. The microcontroller makes 10 000 samples for 539971 clock cycles, it means that is equal 5,999 ms. For a one sample time is divided 10000. The time one sample is 599 ns.

Of course, conversion time can be reduced yet, when we use 10-bit, 8-bit or 6-bit AD converter.

V. CONCLUSION

There is a lot of various possible for measuring the time of conversion, but this method is one of the reliable methods. For a comparison of speed measurements that we find “online” mode is much faster than “offline” method. One conversion of AD converter takes only 599 ns what is great speed. Microcontroller STM32F446RE is fast and very reliable device not only for AD conversions, but for a lot of other applications too.

ACKNOWLEDGMENT

We support research activities in Slovakia / Project is co-financed from EU funds. This paper was developed within the Project "Centre of Excellence of the Integrated Research & Exploitation the Advanced Materials and Technologies in the Automotive Electronics ", ITMS 26220120055

This paper has been prepared under support of Slovak grant project FEI-2015-26.

REFERENCES

[1] SPŠ Martin: Analógové vstupy a výstupy-A/D, D/A prevodníky, online: <http://www.pit.6f.sk/wp-content/uploads/2014/04/27_Prevodn%C3%ADky-AD-DA2.pdf>

[2] LabJack: What does 12- or 16-bit resolution mean?, online: <https://labjack.com/support/faq/what-does-12-or-16-bit-resolution-mean>

[3] Žilinská univerzita – KTEBI, A/D prevodník. Digitalizácie EKG signálu, online: <fel.utc.sk/ktebi/new/predmety/SSL/material/cv_3.pdf

[4] ST, RM0390 Reference manual, online: <www.st.com/resource/en/reference_manual/dm00135183.pdf>

[5] Guzan M., Špaldonová D., Hodulíková A., Tomčíková I., Gladyr A.: *Boundary Surface and Load Plane of the Ternary Memory*, In: Electromechanical and energy saving systems. Vol. 15, no. 3 (2011), p. 163-167. - ISSN 2072–2052

[6] Dziak,J. : *Linear circuit simulation using MATLAB and modeling of nonlinear elements*, In: SCYR 2014 Proceeding from Comference: 20.5.2014: Herľany, S. 70 - 71, Košice : Technická univerzita v Košiciach, 2014 /978-80-553-1714-4/.

[7] Kováčová I.,: *EMC of power DC electrical drives* - 2005. In: Journal of Electrical Engineering. Vol. 5, no. 1 (2005), p. 61-66. - ISSN 1582-4594

[8] Vince T., Molnár J., Bučko R.: *Real-time regulation systems based on internet - optimization algorithm* - 2010. In: Transactions of KMOSU. Vol. 4, no. 3 (2010), p. 150-153. - ISSN 2072-8263

[9] Bereš M.: *Vstupné a výstupné charakteristiky znižovaco- zvyšovacích impulzových DC-DC meničov* - 2016. In: Electrical Engineering and Informatics 7 : proceedings of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. - Košice : FEI TU, 2016 S. 793-796. - ISBN 978-80-553-2599-6

[10] Schweiner D.: *A parallel connection of the DCDC converters and the current-sharing methods* - 2017. In: SCYR 2017. - Košice : TU, 2017 S. 14-17. - ISBN 978-80-553-3162-1