STM32 dla początkujących (i nie tylko)

Przetwornik analogowo – cyfrowy

W tym artykule zajmiemy się przetwarzaniem sygnału analogowego na postać cyfrową. W mikrokontrolerach STM32F służą do tego przetworniki analogowo – cyfrowe wbudowane w strukturę. Opisy i przykłady będą dotyczyły sposobu uruchomienia konwersji oraz różnych trybów pracy przetwornika.

Przetwornik – blok o nazwie ADC – zamienia poziom napięcia wejściowego na równoważną reprezentację binarną. O tym jak wierne jest cyfrowe odwzorowanie sygnału analogowego decydują parametry przetwornika. W mikrokontrolerze STM32F103RB zamontowanym na Panelu Edukacyjnym przetwornik ma następujące parametry:

- Liczba przetworników pracujących niezależnie: 2.
- Rozdzielczość 12-bitowa (możliwość konwersji mierzonego napięcia analogowego na wartość binarną z zakresu 0…4095).
- Minimalny czas trwania konwersji 1 μs.
- 16 niezależnych kanałów pomiarowych (oraz zawarte wewnątrz struktury kontrolera kanały dodatkowe np. czujnik temperatury).
- Rozdzielone zasilania przetworników (w przypadku STM32F103RB będące także napięciami referencyjnymi) od zasilania pozostałych cyfrowych układów kontrolera. Zasilanie przetworników napięcie z przedziału 2,4…3,6 V. Podany na wejście pomiarowe sygnał nie może przekroczyć poziomu napięcia zasilania (referencyjnego).
- Różne tryby pracy i wyzwalania pomiaru.
- Możliwość pracy przetworników w trybie analogowego watchdoga (po przekroczeniu określonego poziomu napięcia na wybranym wejściu może nastąpić przerwanie i programowy restart mikrokontrolera).

Przetworniki mogą pracować niezależnie, jednak należy pamiętać, że każdy z 16 kanałów pomiarowych (wejść) jest wspólny dla obu przetworników.

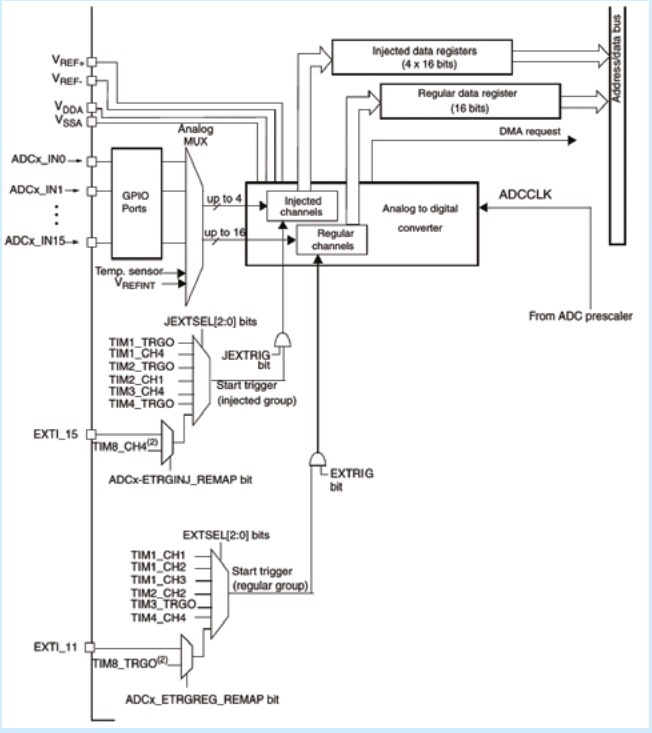
Tryby pracy

Przetworniki zawarte w strukturze STM32F103RB mogą pracować w różnych trybach pracy. Oznacza to zmianę sposobu wykonywania konwersji i jej wyzwalania.

1. **Pomiar jednorazowy lub ciągły.** W drugim przypadku przetwornik po zainicjowaniu działa automatycznie i nieprzerwanie dokonuje konwersji napięć z wcześniej przyporządkowanych kanałów.
2. **Praca z pojedynczym kanałem lub grupą.** W drugim wypadku po zakończeniu konwersji przetwornik automatycznie rozpoczyna konwersję napięcia kolejnego kanału z wcześniej przyporządkowanej grupy.
3. **Praca w trybie *regular* (podstawowej) lub *injected* (wstrzykiwanej).** Konwersje wykonywane w trybie *injected* mają wyższy priorytet niż *regular*. Można zaprogramować pomiar w dwóch grupach, do których zostaną przydzielone różne kanały przetwornika. Wyzwolenie pomiaru *injected* będzie przerywało trwające właśnie pomiary z grupy *regular*. Dodatkowo, wyniki konwersji grupy *injected* umieszczane są w 4 oddzielnych rejestrach danych. Natomiast wyniki kolejnych konwersji grupy *regular* umieszczane są we wspólnym rejestrze danych. Może to grozić nadpisaniem poprzedniego wyniku, nieodczytanego jeszcze przez program przez nowy.
4. **Wyzwalanie pomiarów programowe lub sprzętowe.** W pierwszym przypadku start konwersji następuje po wykonaniu przez główny program komendy sterującej. W drugim przypadku konwersja zostaje uruchomiona przez zdarzenie generowane przez stan licznika lub podanie wysokiego poziomu napięcia na port EXTI kontrolera.

Budowa przetwornika A/C

Na **rysunku 1** pokazano schemat blokowy przetwornika A/C. Rysunek jest uproszczoną wersją schematu z dokumentacji technicznej kontrolera. Konwersja analogowej wartości napięcia na postać cyfrową jest wykonywana w bloku *analog to digital converter*. Zależnie od ustawionego trybu pracy, wynik jest umieszczany albo we



Rysunek 1. Uproszczony schemat blokowy przetwornika A/C mikrokontrolera STM32F103RB

wspólnym dla wszystkich kanałów rejestrze danych *regular data register*, albo w jednym z 4 rejestrów *injected data registers*. Do rejestrów danych ma dostęp oprogramowanie kontrolera.

Napięcie do konwersji wybierane jest przez analogowy multiplekser spośród jednego z 16 kanałów *ADCx_IN0…15* lub z dodatkowego wewnętrznego źródła np. z sensora temperatury.

Sprzętowe wyzwalanie konwersji odbywa się oddzielnie dla grup *regular* i *injected*. Źródłem sygnału wyzwolenia konwersji mogą być TIMER-y lub stan wysoki podany na wejścia portów *EXTI*. Dokładny opis źródeł wyzwalania dla każdego konwertera i trybu pomiaru można znaleźć w dokumentacji w rozdziale *Conversion on external trigger*.

Pokazane na rysunku rozdzielone wyprowadzenia zasilania (Vdda, Vssa) i (Vref+, Vref-) dla kontrolera STM32F103RB w obudowie z 64 wyprowadzeniami są wewnętrznie odpowiednio połączone.

Funkcje biblioteczne konwertera ADC

Zmian ustawień konwerterów ADC dokonuje się poprzez rejestry sterujące. Opis rejestrów i ich bitów znajduje się w dokumentacji technicznej w rozdziale *ADC registers*. Zamiast bezpośrednio operować na rejestrach można także posłużyć się funkcjami z biblioteki *STM32F10x Standard Peripherals Firmware Library*.

Po otwarciu pliku *stm32f10x_stdperiph_lib_um.chm* i wpisaniu w polu wyszukiwarki skrótu „ADC" i wybraniu ADC_Exported_Functions wyświetlone zostaną funkcje obsługujące przetwornik. Funkcje pozwalają przeprowadzić auto kalibrację przetwornika, ustawić tryb pracy, zainicjować konwersję i odczytać wynik z rejestru danych. W opisach procedur i przykładach będę korzystał właśnie z tych funkcji.

Czas konwersji

W przetwornikach STM32F103 minimalny czas konwersji wynosi 1 <m>s, co oznacza maksymalną szybkość próbkowania 1 MS/s. Sam przetwornik może być taktowany zegarem o częstotliwości nieprzekraczającej 14 MHz. Każda konwersja wymaga minimum 14 cykli zegarowych.

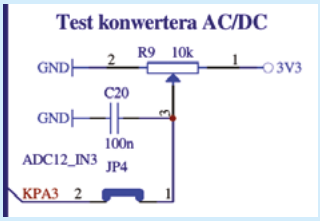
Dla osiągnięcia maksymalnej szybkości należy odpowiednio ustawić wewnętrzne zegary kontrolera. Przede wszystkim magistrala danych APB2 powinna być taktowana zegarem o częstotliwości 56 MHz. Do zmiany tego ustawienia można wykorzystać plik *system_stm32f10x.c* i zmodyfikować go.

W plikach przykładów ustawienie częstotliwości wewnętrznego zegara systemowego wygląda następująco:

```
/* #define SYSCLK_FREQ_HSE    HSE_VALUE */
/* #define SYSCLK_FREQ_24MHz  24000000 */
 #define SYSCLK_FREQ_36MHz  36000000
/* #define SYSCLK_FREQ_48MHz  48000000 */
/* #define SYSCLK_FREQ_56MHz  56000000 */
/* #define SYSCLK_FREQ_72MHz  72000000 */
```

Po zmianie plik powinien wyglądać następująco:

```
 /* #define SYSCLK_FREQ_HSE    HSE_VALUE */
/* #define SYSCLK_FREQ_24MHz  24000000 */
/* #define SYSCLK_FREQ_36MHz  36000000
/* #define SYSCLK_FREQ_48MHz  48000000 */
 #define SYSCLK_FREQ_56MHz  56000000 */
/* #define SYSCLK_FREQ_72MHz  72000000 */
```



Rysunek 2. Fragment schematu z elementami dołączonymi do wejścia analogowego IN3

Funkcja dołączająca zegar magistrali APB2 do przetwornika *RCC_ADCCLKConfig(RCC_PCLK2_Div4)* dzieli go przez cztery: 56 MHz/4=14 MHz. Ponieważ do konwersji potrzeba minimum 14 cykli zegara uzyskana zostanie maksymalna szybkość konwersji równa 1 MHz.

Przykładowe procedury i ustawienia Panelu Edukacyjnego

Podane w dalszej części artykułu przykłady łatwo można przetestować przy pomocy Panelu Edukacyjnego. Jako pojedyncze wejście analogowe wykorzystano kanał IN3 będący portem PA3. Do uzyskania testowego napięcia o zmiennym poziomie służy zamontowany na płytce Panelu potencjometr R9. Fragment schematu z elementami dołączonymi do wejścia analogowego IN3 pokazany został na **rysunku 2**.

Do wyświetlenia wyników konwersji zastosowano 2-liniowy wyświetlacz LCD, który należy dołączyć do złącza J5. W niektórych przykładach dodatkowymi elementami sygnalizacyjnymi mogą być niektóre z diod D1…D8. Jeżeli jako źródło mierzonego napięcia będzie wykorzystywany potencjometr zamontowany na płytce, należy założyć zworę na złączu JP4.

Przy pisaniu programów demonstracyjnych wykorzystano przykłady dostępne w sekcji *STM32F10x_StdPeriph_Examples* biblioteki *STM32F10x Standard Peripherals Firmware Library*. Przykłady pokazują typowe sposoby wykorzystania przetwornika ADC. Mogą być łatwo zmieniane i adoptowane do potrzeb użytkownika.

Programy demonstracyjne przystosowane są do natychmiastowego uruchomienia w pakiecie KEIL5. Wszystkie przykłady należy umieścić w podkatalogu wewnątrz biblioteki: *\Project\STM32F10x_StdPeriph_Template\Podkatalog_przykładu*. Źródła czyli pliki .c i .h powinno dać się bez problemu dostosować do innych typów pakietów programistycznych współpracujących z biblioteką *STM32F10x_StdPeriph_Examples* biblioteki *STM32F10x Standard Peripherals Firmware Library*.

Przetwornik A/C w trybie konwersji ciągłej

W tym przykładzie po zainicjowaniu przetwornik dokonuje ciągłych pomiarów napięcia na wejściu IN3. Pracuje w trybie *regular* co oznacza, że wyniki kolejnych pomiarów umieszczane są w tym samym rejestrze *regular data register*. Procedura inicjując przetwornik A/C może wyglądać jak na **listingu 1**.

Najpierw port PA3 inicjowany jest do pracy jako wejście analogowe kanału 3 przetwornika. Podłączane są impulsy zegarowe do portu i przetwornika. Następnie ADC1 ustawiany jest w trybie pracy ciągłej. Wynik konwersji będzie wyrównywany do prawej, co znaczy, że będzie umieszczany w 12 mniej znaczących bitach 16 bitowego rejestru danych. Po włączeniu przetwornika powinna zostać przeprowadzona

Listing 1. Procedura inicjująca przetwornik ADC1 (tryb pracy ciągłej)

```
void ADC_Inicjacja(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
ADC_InitTypeDef   ADC_InitStructure;
//włączenie sygnału zegarowego dla GPIO pracującego jako wejście
//przetwornika i dla przetwornika ADC1
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO |
                           RCC_APB2Periph_ADC1, ENABLE);
//port PA.03 będzie pracował jako wejście analogowe kanału 3
//przetwornika ADC1
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
//włączenie zegara taktującego przetwornik ADC1
//ADCCLK = PCLK2/4 częstotliwość taktowania = ¼
//częstotliwości wewnętrznej magistrali mikrokontrolera APB2
    RCC_ADCCLKConfig(RCC_PCLK2_Div4);
//konfiguracja przetwornika ADC1
//jeden przetwornik pracujący niezależnie
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
//pomiar jednego kanału bez opcji skanowania pozostałych kanałów
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
//pomiar w trybie ciągłym
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
//bez zewnętrznego wyzwalania pomiaru
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
//dane wyrównane do prawej, znaczących 12 najmłodszych bitów konwersji
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
//jeden kanał
    ADC_InitStructure.ADC_NbrOfChannel = 1;
//inicjacja przetwornika
    ADC_Init(ADC1, &ADC_InitStructure);
//grupa regularna, kanał 3, czas przetwarzania 28,5 cykli
    ADC_RegularChannelConfig(ADC1, ADC_Channel_3, 1, ADC_SampleTime_28Cycles5);
//włączenie ADC1
    ADC_Cmd(ADC1, ENABLE);
//zerowanie rejestrów kalibracyjnych
    ADC_ResetCalibration(ADC1);
//oczekiwanie na zakończenie zerowania
    while(ADC_GetResetCalibrationStatus(ADC1));
//start kalibracji ADC1
    ADC_StartCalibration(ADC1);
//oczekiwanie na zakończenie kalibracji ADC1
    while(ADC_GetCalibrationStatus(ADC1));
//start przetwarzania
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}
```

Listing 2. Procedura inicjująca przetwornik ADC1 (tryb pracy *injected*)

```
void ADC_Inicjacja(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
ADC_InitTypeDef   ADC_InitStructure;
//włączenie sygnału zegarowego dla GPIO pracującego
//jako wejście przetwornika i dla przetwornika ADC1
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO |
                           RCC_APB2Periph_ADC1, ENABLE);
//port PA.03 będzie pracował jako wejście analogowe kanału 3
//przetwornika ADC1
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
//włączenie zegara taktującego przetwornik ADC1
//ADCCLK = PCLK2/4 częstotliwość taktowania = 1/4
//częstotliwości wewnętrznej magistrali mikrokontrolera APB2
    RCC_ADCCLKConfig(RCC_PCLK2_Div4);
//konfiguracja przetwornika ADC1
//jeden przetwornik pracujący niezależnie
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
//pomiar jednego kanału bez opcji skanowania pozostałych kanałów
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
//wyłącz pomiar w trybie ciągłym
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
//zewnętrzne wyzwalanie pomiaru przez linię EXTI 15
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigInjecConv_Ext_IT15_TIM8_CC4;
//dane wyrównane do prawej, znaczących 12 najmłodszych bitów konwersji
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
//jeden kanał
    ADC_InitStructure.ADC_NbrOfChannel = 1;
//inicjacja przetwornika
    ADC_Init(ADC1, &ADC_InitStructure);
/* Enable ADC1 external trigger conversion */
    ADC_ExternalTrigConvCmd(ADC1, ENABLE);
//sekwencer ustawiony do pomiaru tylko 1 kanału
    ADC_InjectedSequencerLengthConfig(ADC1, 1);
//konfiguracja kanału 3 w trybie injection
    ADC_InjectedChannelConfig(ADC1, ADC_Channel_3, 1, ADC_SampleTime_28Cycles5);
//start konwersji w trybie injection będzie wyzwalany zboczem narastającym podawanym na wejście
EXTI 15
    ADC_ExternalTrigInjectedConvConfig(ADC1, ADC_ExternalTrigInjecConv_Ext_IT15_TIM8_CC4);
//start trybu injection
    ADC_ExternalTrigInjectedConvCmd(ADC1, ENABLE);
//włączenie ADC1
    ADC_Cmd(ADC1, ENABLE);
//zerowanie rejestrów kalibracyjnych
    ADC_ResetCalibration(ADC1);
//oczekiwanie na zakończenie zerowania
    while(ADC_GetResetCalibrationStatus(ADC1));
//start kalibracji ADC1
    ADC_StartCalibration(ADC1);
//oczekiwanie na zakończenie kalibracji ADC1
    while(ADC_GetCalibrationStatus(ADC1));
}
```

Lisitng 3. Procedura inicjująca przetwornik ADC1 (tryb wyzwalany synchronicznie za pomocą Timera 2)

```
void ADC_Inicjacja(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
ADC_InitTypeDef   ADC_InitStructure;

//włączenie sygnału zegarowego dla GPIO pracującego jako
//wejście przetwornika i dla przetwornika ADC1
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO |
                           RCC_APB2Periph_ADC1, ENABLE);
//port PA.03 będzie pracował jako wejście analogowe kanału 3
//przetwornika ADC1
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
//włączenie zegara taktującego przetwornik ADC1
// ADCCLK = PCLK2/4 częstotliwość taktowania = 1/4 częstotliwości
//wewnętrznej magistrali  //mikrokontrolera APB2
    RCC_ADCCLKConfig(RCC_PCLK2_Div4);
//konfiguracja przetwornika ADC1
//jeden przetwornik pracujący niezależnie
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
//pomiar jednego kanału bez opcji skanowania pozostałych kanałów
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
//wyłączenie pomiaru w trybie ciągłym
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
//wyzwalanie z TIM2 CC2
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_T2_CC2;
//dane wyrównane do prawej, znaczących 12 najmłodszych bitów konwersji
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
//jeden kanał
    ADC_InitStructure.ADC_NbrOfChannel = 1;
//inicjacja przetwornika
    ADC_Init(ADC1, &ADC_InitStructure);
/* Enable ADC1 external trigger conversion */
    ADC_ExternalTrigConvCmd(ADC1, ENABLE);
//grupa regularna, kanał 3, czas przetwarzania 28,5 cykli
    ADC_RegularChannelConfig(ADC1, ADC_Channel_3, 1, ADC_SampleTime_28Cycles5);
//włączenie ADC1
    ADC_Cmd(ADC1, ENABLE);
//zerowanie rejestrów kalibracyjnych
    ADC_ResetCalibration(ADC1);
//oczekiwanie na zakończenie zerowania
    while(ADC_GetResetCalibrationStatus(ADC1));
//start kalibracji ADC1
    ADC_StartCalibration(ADC1);
//oczekiwanie na zakończenie kalibracji ADC1
    while(ADC_GetCalibrationStatus(ADC1));
}
```

procedura auto kalibracji. Na koniec komendą *ADC_SoftwareStartConvCmd(ADC1, ENABLE)* włączany zostaje proces ciągłej konwersji.

Odczyt wyniku konwersji rozkazem *ADC_GetConversionValue(ADC1)* odbywa się w nieskończonej pętli głównej procedury main(). Na wyświetlaczu będzie wyświetlany wynik ciągłej konwersji. Przy założeniu, że napięcie zasilania (referencyjne) wynosi 3,3 V wynik zamieniany jest na mV i wyświetlany.

Przetwornik A/C wyzwalany zboczem narastającym wejścia EXTI15

W drugim przykładzie kolejny pomiar jest wyzwalany zboczem narastającym impulsu podawanego na wejście EXTI15. Wejście to połączone jest z portem PB15. Jak można zauważyć na rys. 1, to wejście inicjuje pomiar w trybie *injected*. Procedurę inicjowania przetwornika A/C w trybie *injected* przedstawiono na **listingu 2**.

Jak poprzednio (list. 1) jako pierwszy jest inicjowany port PA3 i są włączane zegary do portu i przetwornika. Następnie ADC1 inicjowany jest do pracy w trybie wyzwalania pomiaru przez zewnętrzne wejście EXTI15. Wynik konwersji będzie wyrównywany do prawej. Przetwornik ustawiany jest do pracy w trybie *injected* w grupie z jednym wejściem IN3 przyporządkowanym do *injected channel 1*. Po włączeniu przetwornika przeprowadzana jest jego kalibracja. Jeżeli teraz na wejście portu PB15 podany zostanie impuls, jego przednie zbocze wyzwoli kolejną konwersję przetwornika. Dioda D8 będzie sygnalizować stan napięcia na wejściu PB15 o ile zostanie założona zwora JP6 15-16.

Wynik konwersji odczytywany jest w pętli głównej procedury main() rozkazem *ADC_GetInjectedConversionValue(ADC1, ADC_InjectedChannel_1)*. Tak jak poprzednio wynik konwersji wyświetlany jest w mV.

Przetwornik A/C wyzwalany przez Timer 2

Trzeci przykład służy do pokazania jak do wyzwalania pomiaru ADC1 można wykorzystać Timer 2. Takie rozwiązanie pozwala dokonać serii pomiarów w sposób całkowicie automatyczny. W omawianym przykładzie można ustawić czas pomiaru w zakresie od 1 do 65565 sekund. Wykorzystując do wyzwalania pomiaru TIMER2 i jego kanał 2, ADC1 musi pracować w trybie *regular*. Procedurę inicjowania pokazano na **listingu 3**.

Procedura jest bardzo podobna do tej z list. 2. Różnica polega na wskazaniu jako źródła wyzwalania pomiaru licznika TIMER2 i jego kanału 2: *ADC_ExternalTrigConv_T2_CC2*. Ponieważ konwerter będzie pracował w trybie *regular* nie ma potrzeby ustawiania sekwencera trybu *injected*.

W głównym programie *main()* po wywołaniu procedury inicjacji przetwornika ADC następuje inicjacja TIMER2. Będzie on pracował jako PWM z impulsami generowanymi przez kanał CC2 i wyprowadzonymi na porcie PA1. Po uruchomieniu licznika procedurą *Start_TIMER2()* rozpoczyna się automatyczne wyzwalanie ADC1 co 6 sekund i zapis wyniku konwersji do rejestru danych.

Użyte w przykładzie przerwanie TIMER2 służy jedynie do zwiększenia licznika kolejnego pomiaru *numer_konwersji*. Stan licznika i wartość ostatniego pomiaru wyświetlane są na wyświetlaczu. Jeżeli połączy się przewodem wyjście PA1 (złącze J6) z którąś z diod sygnalizacyjnych (złącze JP6) świecenie diody będzie wskazywać moment wykonywania kolejnego pomiaru.

Ryszard Szymaniak, EP