OBSŁUGA PRZETWORNIKA ADC NA MIKROKONTROLERZE ATMEGA8

Opracowanie zawiera treści różnych publikacji takich jak:
książki, datasheety, strony internetowe

Cezary Klimasz
Kraków 2008

Spis treści

1. **Wprowadzenie .....** str. 3
2. **Podstawowe wiadomości .....** str. 3
3. **Przykład obsługi przetwornika w języku C .....** str. 7
4. **Podsumowanie .....** str. 9
5. **Bibliografia .....** str. 9

1. Wprowadzenie

Co jest pośrednikiem powiędzy sygnałem analogowym a sygnałem cyfrowym ? Oczywiście przetwornik. Dzięki zastosowaniu **ADC** (*Analog-to-Digital Converter*) lub **DAC** (*Digital-to-Analog Converter*) możliwe jest wykorzystanie techniki analogowej oraz cyfrowej na raz. Stwarza to olbrzymie możliwości w wielu dyscyplinach elektroniki, dlatego istnieje wiele rodzajów przetworników realizujących przetwarzanie sygnałów. Podstawowymi parametrami takich przetworników jest rozdzielczość czyli zdolność przetwornika do wiernego odtworzenia sygnału oraz szybkość jego działania.

Celem niniejszego opracowania jest wykorzystanie bloku przetwornika AD wbudowanego w mikrokontroler **ATmega8**. Uwaga zostanie zwrócona na opis katalogowy mikrokontrolera oraz praktycznie wykorzystanie w systemie pomiarowym (pomiar natężenia pola magnetycznego czujnikami *Halla*).

2. Podstawowe wiadomości

MikrokontroleryATmega8 posiadają w zależności od obudowy:

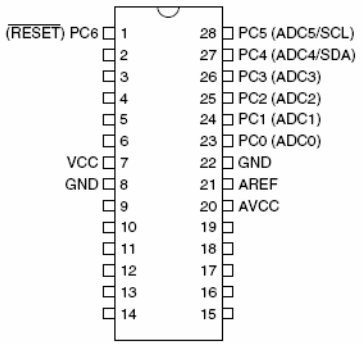
- **8 kanałowy przetwornik ADC** dla układów w obudowach **SMD**
- **6 kanałowy przetwornik ADC** dla układów produkowanychw obudowach **DIP**.

Wspólne cechy przetworników w ATmega8 to:

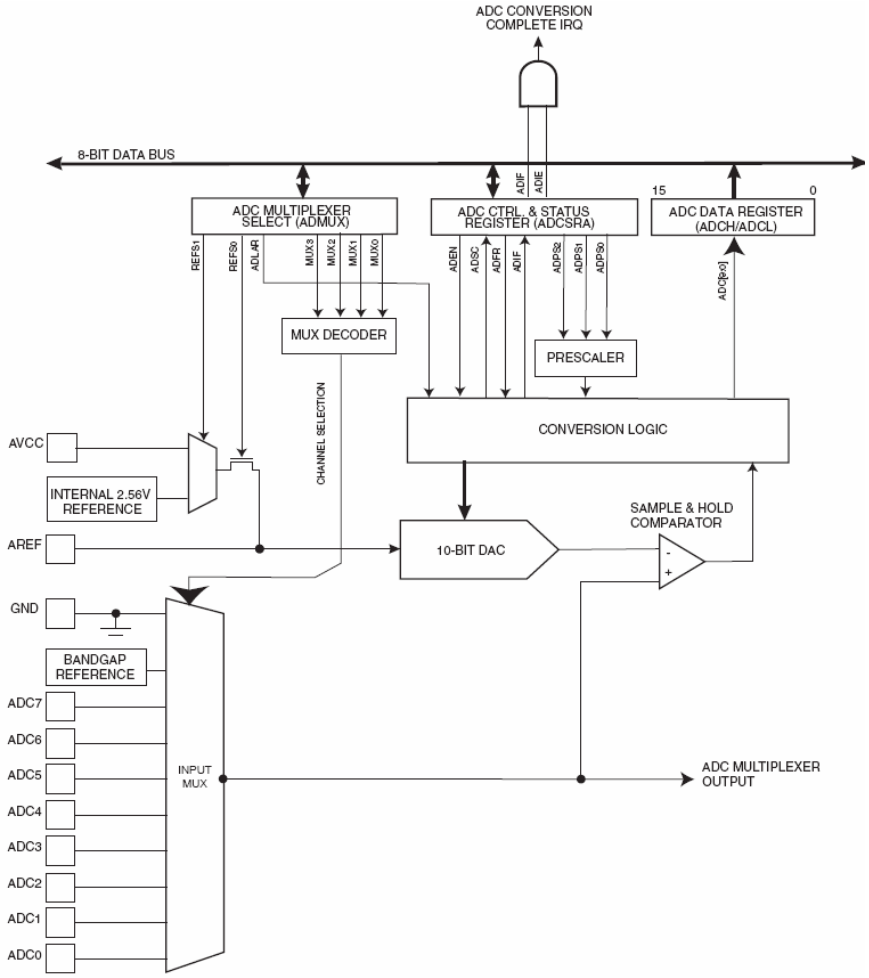
- rozdzielczość **10 bitów** (1024 próbek),
- **0.5 LSB** nieliniowości charakterystyki przetwarzania,
- **±2 LSB** całkowitej dokładności,
- od **13-260µs** czasu konwersji,
- 6 multipleksowanych pojedyńczych wejść,
- 2 wspólnie multipleksowane wejścia (obudowy SMD),
- zakres pomiaru ADC od 0 do V_{cc}
- przerwanie od zakończenia konwersji na postać cyfrową.

10-bitowy przetwornik ADC podłączony jest do 8-kanałowego analogowego multipleksera. Wejścia tego multipleksera podłączone są do Portu C mikrokontrolera. Przetwornik teoretycznie posiada odseparowany obwód zasilania. Niestety w mikrokontrolerze ATmega8 dokumentacja wprowadza w błąd czytelnika, gdyż de facto zasilanie mikrokontrolera oraz zasilanie przetwornika nie są odseparowane (błąd podczas produkcji – inne mikrokontrolery AVR mają odseparowane zasilania).

Wewnętrzne napięcie odniesienia dla przetwornika wynosi 2.56V lub **AV_{cc}** podane do mikrokontrolera. Napięcie odniesienia może być zewnętrznie odsprzężony poprzez podpięcie kondensatora do pinu **AREF**. Poniżej widoczny jest mikrokontroler ATmega8 w obudowie DIP z opisanymi pinami znaczącymi dla ADC.



Poniżej widoczny schemat blokowy przetwornika ADC.



Przetwornik konwertuje wejściowe napięcie analogowe na 10-bitową cyfrową wartość. Minimalną wartość reprezentuje poziom masy zaś maksymalną napięcie AREF minus 1 najmniej znaczący bit. Opcjonalnie, AV_{cc} oraz wewnętrzne źródło odniesienia 2.56V mogą być podłączone do pinu AREF poprzez zapis w rejestrze **ADMUX** bitu **REFSn**.

Analogowe wejście wybierane jest poprzez zapis bitów **MUX** w rejestrze **ADMUX**. Przetwornik aktywowany jest poprzez ustawienie bitu **ADEN** w rejestrze **ADCSRA**. Ustawienie napięcia odniesienia czy kanału wejściowego nie będzie miało skutku dopóki ADEN nie zostanie ustawiony. Przetwornik nie pobiera energii jeśli bit ADEN jest wyczyszczony, dlatego zaleca się wyłączać przetwornik ADC przez przejściem do trybu pracy *sleep mode*.

ADC generuje 10-bitowy wynik, który prezentowany jest w rejestrach **ADCH** i **ADCL**. Domyślnie wynik prezentowany jest ze skorygowaniem do prawej, można to zmienić ustawiając korygowanie do lewej poprzez ustawienie bitu **ADLAR** w rejestrze **ADMUX**. Jeśli wynik korygowany jest do lewej i potrzebna jest dokładnoś wyższa niż 8 bitów, wystarczający jest odczyt jedynie z rejestru ADCH. Z kolei, ADCL musi być odczytane pierwsze, następnie ADCH aby mieć pewność, że oba wyniki pochodzą z tej samej konwersji. Kiedy ADCL jest odczytywany ADC blokuje dostęp do rejestrów danyh. Oznaza to, że kiedy ADCL jest odczytywane oraz konwersja zakończyła się przed oczytem ADCH, żaden rejest nie jest modyfikowany oraz żaden wynik nie jest tracony. Kiedy odczytywany jest rejestr ADCH, ADC aktywuje dostęp do rejestrów danych.

Przetwornik ADC posiada własna przerwania, które wyzwalane są kiedy przetwarzanie do postaci cyfrowej kończy się. Kiedy przetwornik korzysta z rejestrów danych zabronione jest odczytywanie rejestrów ADCH oraz ADCL.

Konwersja rozpoczyna się w momencie zapisu logicznej jedynki do bitu ADSC. Bit ten pozostaje w stanie wysokim tak długo jak długo trwa konwersja i jest czyszczony sprzętowo kiedy konwersja kończy się. Jeśli podczas konwersji zostanie wybrany inny kanał danych, ADC skończy bieżącą konwersję przed zmianą kanału.

Domyślnie aby uzyskać maksymalną rozdzielczość pomiaru wejściowy zegar powinien taktować z częstotliwością pomiędzy 50kHz a 200kHz. Jeśli potrzebujemy mniej niż 10-bitów rozdzielczości a częstsze próbkowanie możemy zwiększyć częstotliwość taktowania powyżej 200kHz. Blok przetwornika zawiera w sobie preskaler, który generuje sygnał zegarowy dla ADC ponad 100kHz. Preskaler ustawiany jest poprzez wybór bitu ADPS w rejestrze ADCSRA. Preskaler zaczyna zliczać od momentu kiedy ADC zostaje załączony – ustawienie bitu ADEN w rejestrze ADCSRA. Zliczanie trwa tak długo jak długo ustawiony jest bit ADEN.

Po konwersji pomiaru (bit ADIF ustawiany na wysoki), wynik zostaje zapisany w rejestrach ADCL oraz ADCH. Zależność wiążąca wynik z pomiarem:

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

Wartość *0x000* reprezentuje poziom masy, zaś *0x3FF* reprezentuje wartość napięcia odniesienia minus 1 LSB.

Opis poszczególnych rejestrów ADC

ADC Multiplexer Selection
ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bity 7:6 – REFS1:0 – Bit wyboru napięcia referencyjnego
Bit ten odpowiada za wybór odpowiedniego napięcia co pokazuje poniższa tabela.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Bit 5 – ADLAR: ADC Left Adjust Result
Bit ten odpowiada za odpowiednie prezentowanie wyniku konwersji. Zapis bitu jako logicznej jedynki powoduje ustawienie korygowania do lewej. W innym wypadku do prawej.

Bity 3:0 – MUX3:0: Bit wyboru kanału analogowego
Wartość zapisana w tych bitach wybiera które z analogowych wejść podłączone jest aktualnie do przetwornika.

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7

ADC Control and Status
ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – ADEN: ADC Enable
Zapis tego bitu jako jeden aktywuje przetwornik. Zapis logicznego zera wyłącza ADC. Wyłączenia ADC podczas konwersji powoduje zakończenie tej konwersji.
Bit 6 – ADSC: ADC Start Conversion
W trybie pojedynczej konwersji, zapis tego bit jako jeden powoduje start konwersji. W trybie Free Running zapis tego bitu jako jeden powoduje start pierwszej konwersji. ADSC jest tak długo ustawione na jeden dopóki konwersja trwa.
Bit 5 – ADFR: ADC Free Running Select
Ustawienie tego bitu jako jeden powoduje, że ADC działa w trybie Free Running.
Bit 4 – ADIF: ADC Interrupt Flag
Bit ten jest ustawiany kiedy konwersja kończy się i rejestry danych są uaktualnione. Przerwanie od zakończenia konwersji jest aktywne kiedy bit ADIE oraz I-bit w rejstrze mikrokontrolera SREG są ustawione.
Bit 3 – ADIE: ADC Interrupt Enable
Kiedy ten bit jest ustawiony wraz z I-bitem w rejestrze SREG wtedy aktywne jest przerwanie od zakończenia konwersji.
Bity 2:0 – ADPS2:0: Wybór preskalera
Bity te określają są za współczynnik podziału pomiędzy częstotliwości taktowania mikrokontrolera a wejściowym zegarem ADC.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADC Data Register – ADCL, ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Kiedy konwersja się kończy wynik zapiswany jest do tych obu rejestrów.

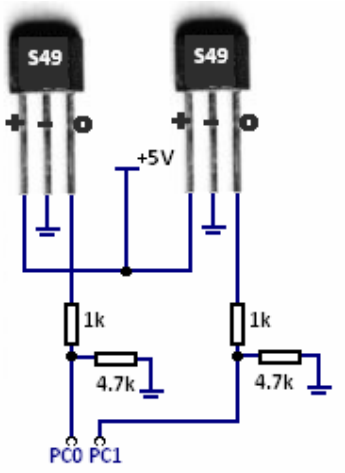
ADC9:0: Wynik konwersji
Bity te reprezentują wynik konwersji.

3. Przykład obsługi przetwornika w języku C

W układach pomiarowych czujniki zazwyczaj posiadają wyjścia analogowe, choć nie jest to już regułą. Chcąc zinterpretować otrzymany sygnał analogowy musimy podać go na przetwornika ADC. W naszym wypadku zakładamy użycie dwóch czujników Halla (czujniki pola magnetycznego) używanych w układach pozycjonowania czy pomiaru prędkości obrotowej. Dla nas nie istotne będzie zastosowanie tych czujników, ale przetworzenie ich wskazań do postaci cyfrowej.

Czujniki Halla *SS49* (prod. *Honeywell*) dają na wyjściu sygnał napięciowy proporcjonalny do przyłożonego pola magnetycznego. Nie są one jednak na tyle dokładne aby można było mierzyć bezpośrednio pole magnetyczne Ziemi. Czujniki zasilane są napięiem zakresu *4-10V*.

Poniżej widoczny schemat podpięcia czujników do mikrokontrolera.



Wartości rezystorów akurat w wypadku naszych czujników nie mają większego znaczenia. W doborze może być pomocna informacja o tym iż wewnętrzne źródło napięcia referencyjnego mikrokontrolera wynosi 2.56V[1]. Im niższe napięcie referencyjne tym dokładniejszy pomiar możemy przeprowadzić, gdyż zawsze wartość wskazana przez ADC - *0x000* jest to wartość poziomu masy, zaś wartość *0x3FF* (*1024*) jest wartością napięcia referencyjnego minus 1LSB.

Jeśli chodzi o realizację programową obsługi przetwornika to środowiskiem w którym to wykonamy będzie WinAVR (*AVR-GCC*). Realizacja programowa zakłada użycie dwóch kanałów przetwornika oraz wykorzystanie przerwania od zakończenia przetwarzania AD. Konfiguracja sprzętowa: *AREF +5V*, *AVCC +5V*, *PC0* – sensor1, *PC1*- sensor2.

```
#include <avr/io.h>
#include <avr/signal.h>
#include <avr/interrupt.h>

#define VREF 5                                          //wartosc napiecia referencyjnego

volatile float result;                                  //obl. wartosc napiecia
volatile uint8_t counter;                               //licznik co ktory pomiar pobierac
volatile float measure_sensor1, measure_sensor2;        //wyniki dla kazdego z czujnikow

//przerwanie od zakonczenia przetwarzania przez ADC
SIGNAL(SIG_ADC)
{
result=(float)(ADCL | (ADCH<<8))/1024*VREF;             //obl. wartosci napiecia na wejsciu
counter++;                                              //zwiekszanie licznika

if (counter==50) {                                      //jesli co 50 pomiar
    if (bit_is_set(ADMUX,MUX0))                         //w zaleznosci od ust. kanalu
        {
            measure_sensor1=result;                     //pobranie pomiaru
            ADMUX&=~_BV(MUX0);                          //przelaczenie kanalu
        }
        else
        {
            measure_sensor2=result;                     //pobranie pomiaru
            ADMUX=_BV(MUX0);                            //przelaczenie kanalu
        }

        counter=0;                                      //wyzerowanie licznika
    }

} // SIGNAL(SIG_ADC)

//funkcja inicjuje przetwornik ADC
void init_adc(void)
{
//ADEN - aktywowanie ADC,               ADIE - aktywowanie przerwania
//ADFR - Free Runing (pomiar caly czas), ADSC - start pomiaru
//ADPSX - ustawienie preskalera
ADCSR = _BV(ADEN)|_BV(ADIE)|_BV(ADFR)|_BV(ADSC)|_BV(ADPS0)|_BV(ADPS1)|_BV(ADPS2);
} //init_adc


int main(void)
{
init_adc();                                             //aktywowanie ADC
sei();                                                  //zwolnienie przerwan globalnych
    for(;;)                                             //petla nieskonczona
    {}
return 0;
} //main
```

[1] *1LSB=2.56/1024=0.0025*

Kluczową funkcją działania przetwornika jest odpowiednia jego inicjalizacja. Jako, że będziemy korzystać z niego oczywiście aktywujemy go: *ADEN*, pozwalamy na przerwanie od zakończenia przetwarzania *ADIE* oraz ustawiamy tryb Free Running – *ADFR*. Tryb ten pozwala na realizację pomiarów non-stop. Bit *ADSC* uruchamia pomiary, zaś bity *ADSPX* odpowiedzialne są za ustawienie odpowiedniego preskalera i związanej z tym częstotliwości pracy przetwornika.

Ważną operacją jest również multipleksowanie kanałow przetwornika. Jak widać zmiana kanałów (zmiana bitów w rejestrze *ADMUX*) odbywa się co 50 przerwań. Ogólnie założono, że pod uwagę będzie brany co 50 wynik przetwarzania. Oczywiśie można to uskutecznić licząc średnią arytmetyczną z 50 pomiarów ale w wypadku programu jakim jest powyższy program testowy nie miałoby to i tak głębszego sensu.

Jak widać obsługa programowa przetwornika jest prosta i możliwości które oferuje ADC są duże, może poza rozdzielczością (10 bitów) oraz wartością minimalnego napięcia referencyjnego (*2V* przy zasilaniu mikrokontrolera *5V*). W wypadku napięć innych niż *0-5V* należy stosować dzielniki napięcia, ew. wzmacniacze w przypadku sygnałów zbyt małych.

4. Podsumowanie

W niniejszym opracowaniu przedstawiono jedynie podstawowe informacje na temat przetwornika oraz sposóbu przeprowadzenia pomiaru. Jednakże opis dokumentacji powinien pozwolić na realizację trudnych projektów, w których kluczową rolę pełnić będzie przetwornik ADC.

5. Bibliografia

[1] ATMEL. *ATmega8 datasheet 07/07*. 2007