

Podstawy systemów mikroprocesorowych

Wykład nr 6

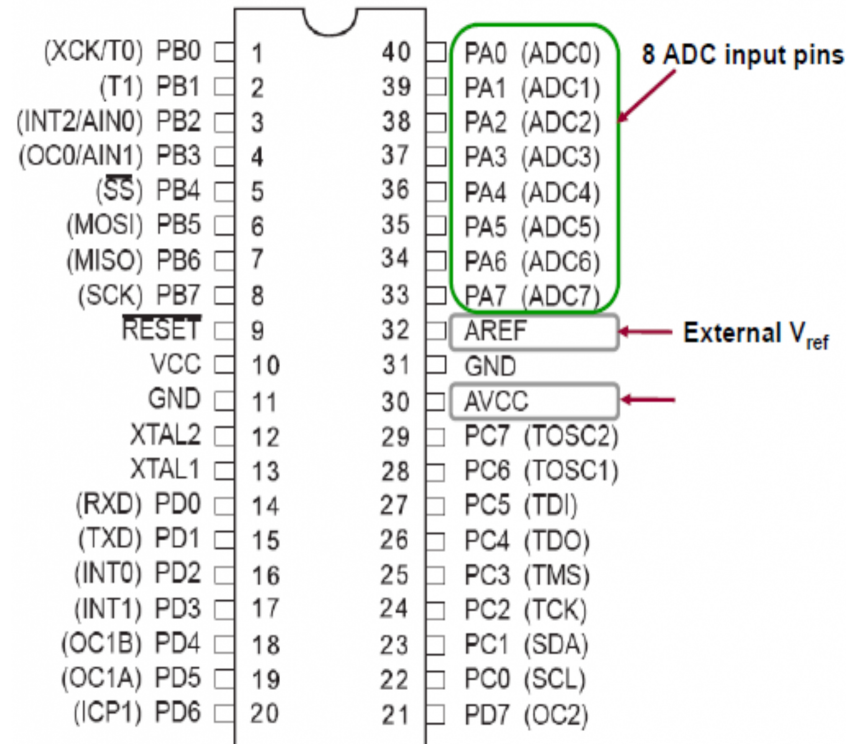
dr inż. Dariusz Tefelski

na podstawie wykładu dr hab. Piotra Fronczaka

http://fizyka.if.pw.edu.pl/~labe/index.php/Wyklad_PSM

Przetwornik ADC

- rozdzielczość 10 bitów
- 8 kanałów wejściowych (8 źródeł sygnałów)
- próbkowanie sekwencyjne (jeden kanał na raz)
- dla standardowego napięcia odniesienia $V_{ref} = 5V$
 - kwant pomiaru: $5(V)/1024 = 4.88mV$.
 - dokładność $\pm 2 \text{ LSB} = \pm 9.76mV$
- Inne możliwe źródła napięć odniesienia
 - źródło wewnętrzne 2,56V
 - napięcie podane na nóżkę AREF
- Tryby pomiarów
 - ciągły pomiar + generowanie przerwania ADC
 - na żądanie



Przetwornik ADC

Wynik konwersji zapisany jest w dwóch rejestrach ADCH i ADCL

Wyrównanie wyniku konwersji do prawej (bit ADLAR = 0)

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	

Wyrównanie wyniku konwersji do lewej (bit ADLAR = 1)

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	

Jeśli zadowala nas rozdzielczość 8-bitowa, to wystarczy odczytać ADCH.

Przetwornik ADC - konfiguracja

Krok 1: Konfiguracja rejestrów ADMUX i ADCSRA.

Który kanał ADC? _____

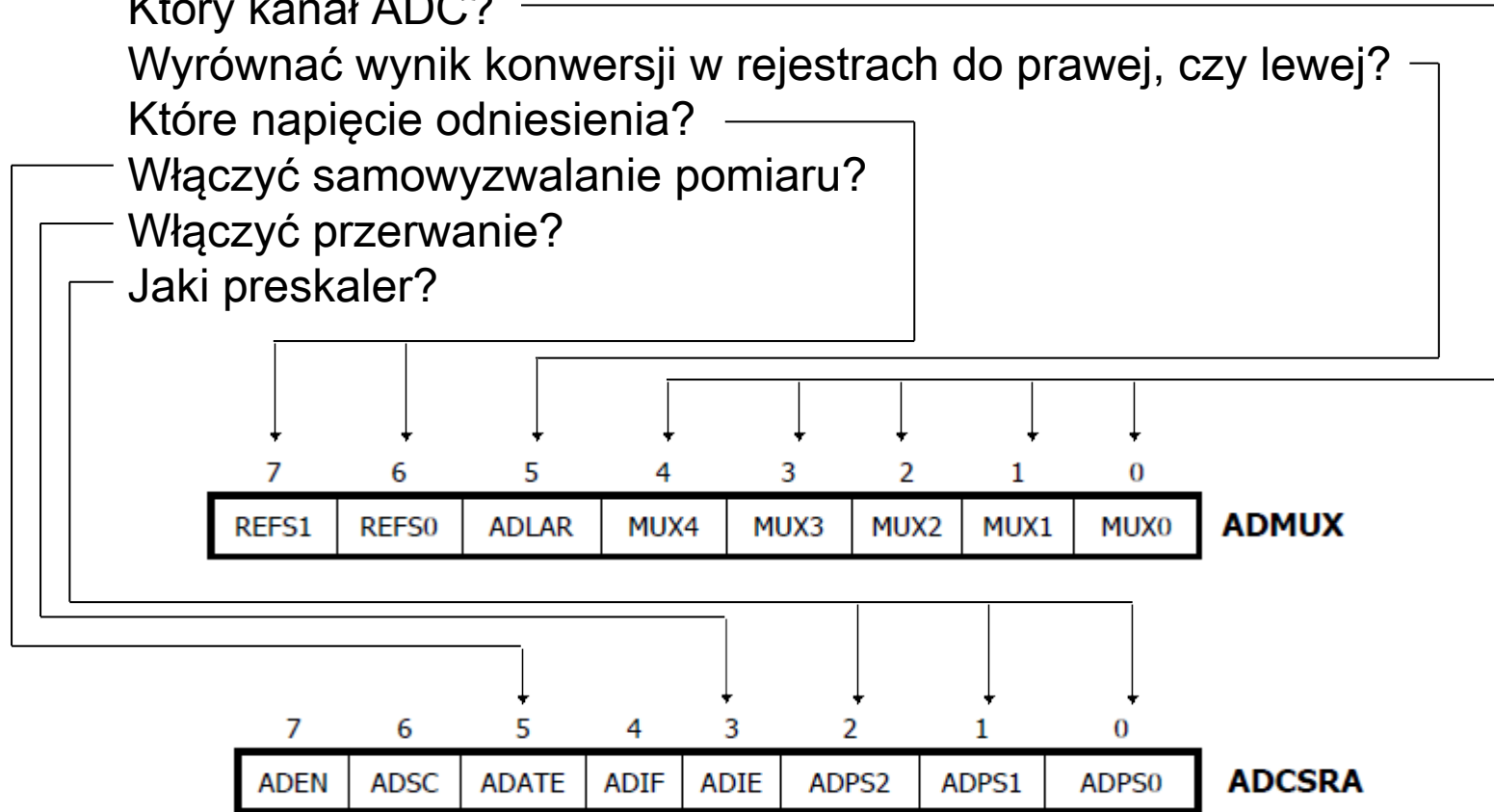
Wyrównać wynik konwersji w rejestrach do prawej, czy lewej? _____

Które napięcie odniesienia? _____

Włączyć samowyzwalanie pomiaru? _____

Włączyć przerwanie? _____

Jaki preskaler? _____



Przetwornik ADC

Krok 2: Uruchomienie konwersji ADC

Wpisz 1 do flagi ADSC rejestru ADCSRA.

Krok 3: Odczytanie wyniku

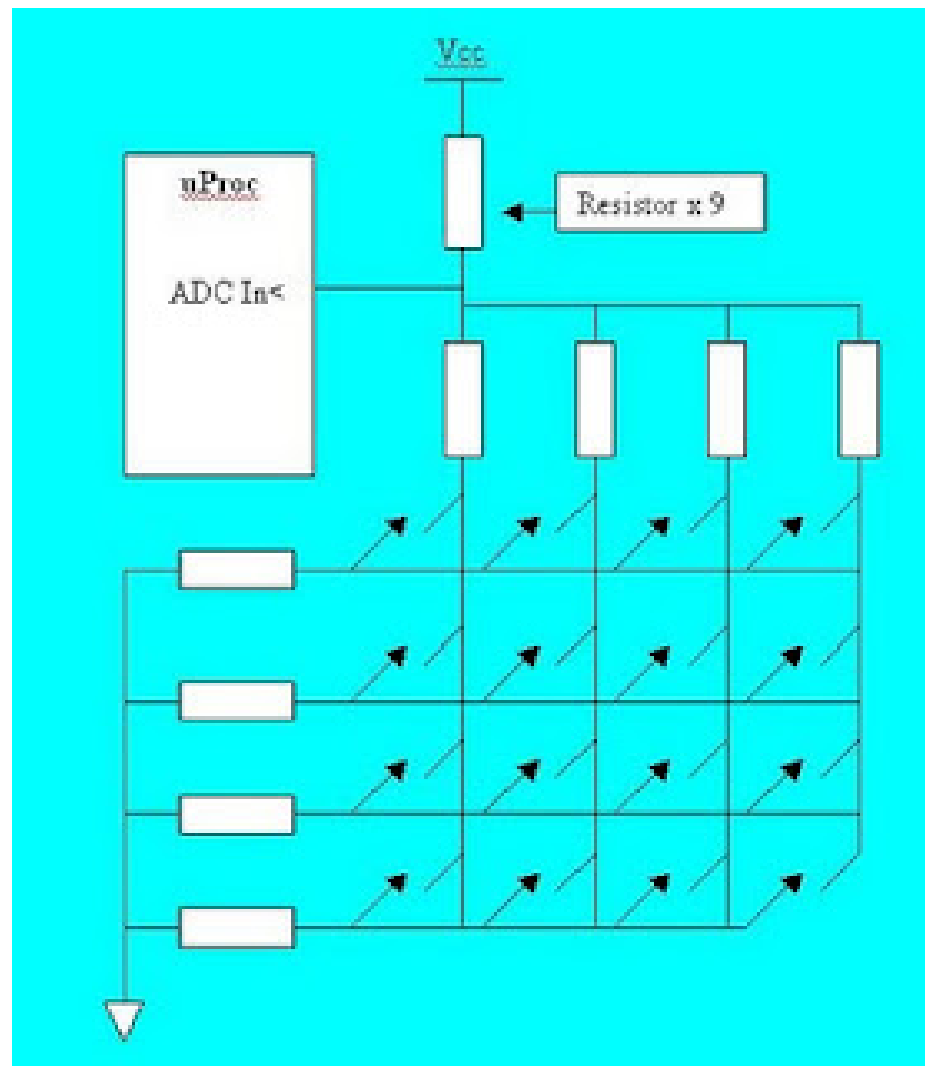
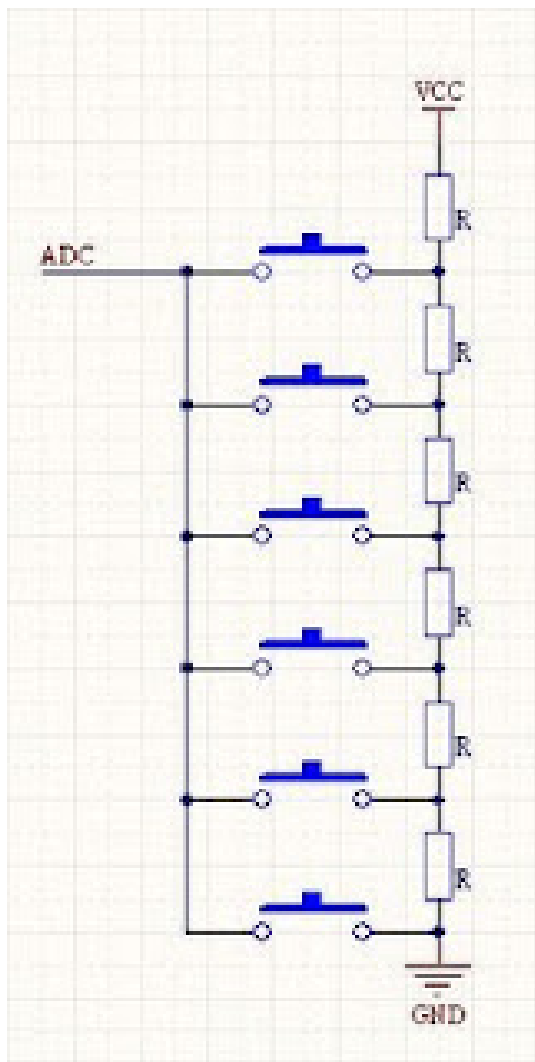
Czekaj, aż flaga ADSC wyzeruje się.

Odczytaj wynik z rejestrów ADCL i ADCH.

```
#include<avr/io.h>
int main (void){
    unsigned char result;
    ADMUX = 0b01100000; // REFS1:0 = 01 -> AVCC as reference,
                        // ADLAR = 1 -> wyrównanie do lewej
                        // MUX4:0 = 00000 -> kanał ADC0
    ADCSRA = 0b10000001; // ADEN = 1: włączamy przetwornik
                        // ADSC = 1: start konwersji
                        // ADIF = 0: tryb ręczny
                        // ADIE = 0: bez przerwań
                        // ASPS2:0 = 001: prescaler = 2

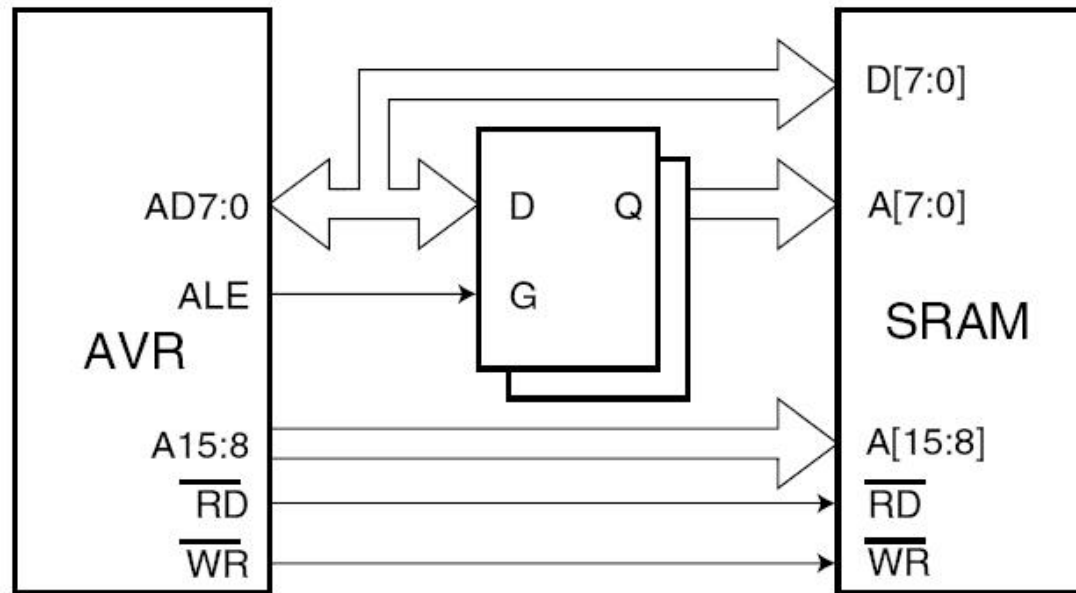
    while(1){
        ADCSRA |= (1 << ADSC); // krok 2
        while (ADCSRA & (1 << ADSC)); // krok 3
        result = ADCH; // odczytujemy wynik
    }
}
```

Przetwornik ADC - przykład

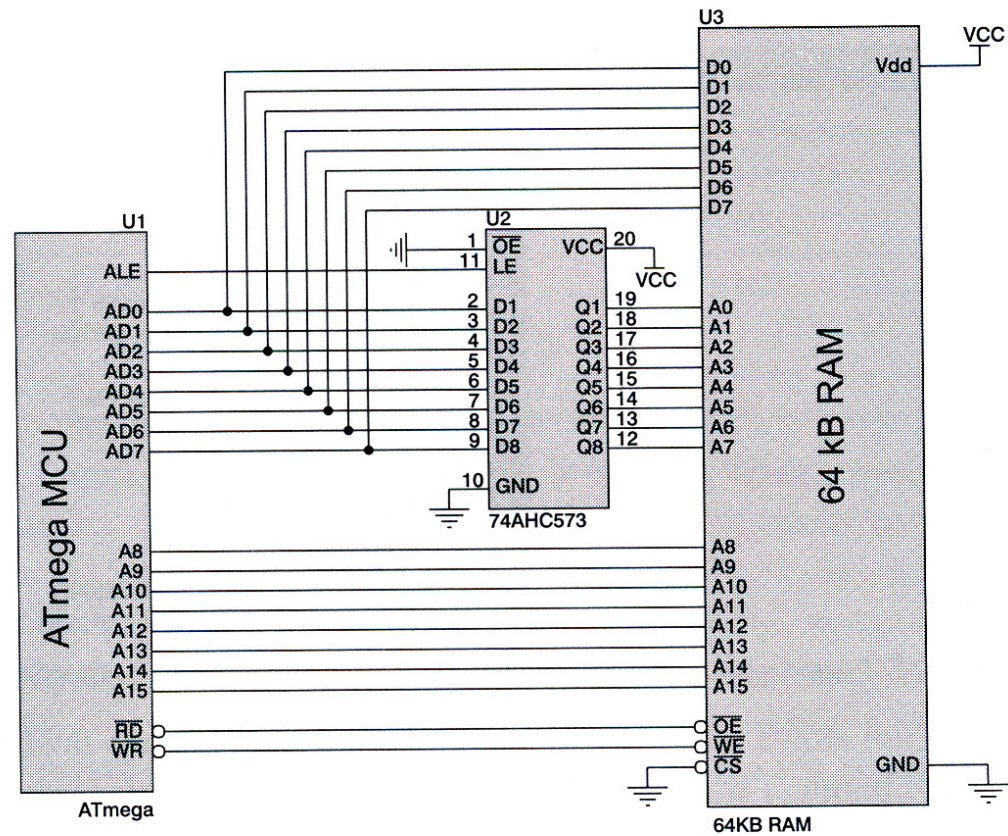


Pamięci równoległe

- n linii (bitów) danych
- m linii (bitów) adresowych
- Linie sterujące:
 - RD – żądanie odczytu (ang. *read strobe*)
 - WR – żądanie zapisu (ang. *write strobe*)
 - ALE – *Address Latch Enable* – niezbędna linia do współdzielenia linii adresowych z liniami danych

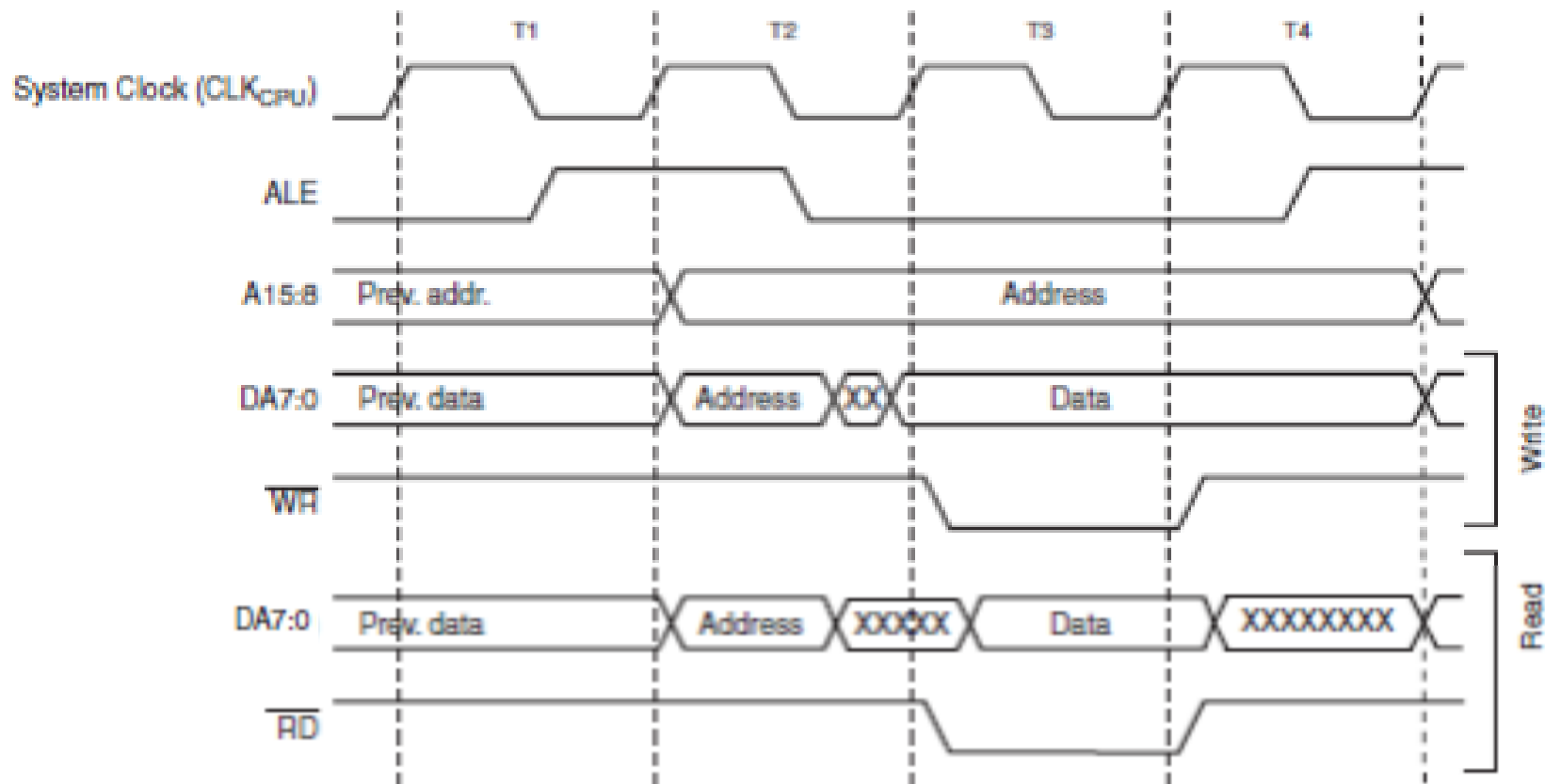


Pamięci równoległe



Rys. 1.16. Sposób podłączenia zewnętrznej pamięci RAM

Pamięci równoległe



Źródła sygnału RESET

ATmega16 posiada pięć źródeł sygnału reset (omówimy cztery, piąty wykorzystywany jest w interfejsie JTAG).

Power-on RESET

Układ ten odpowiada za resetowanie mikrokontrolera w sytuacji przywrócenia napięcia zasilającego. Układ utrzymuje sygnał RESET w stanie aktywnym przez określony odcinek czasu (poprzez fusebity).

Zewnętrzny sygnał RESET

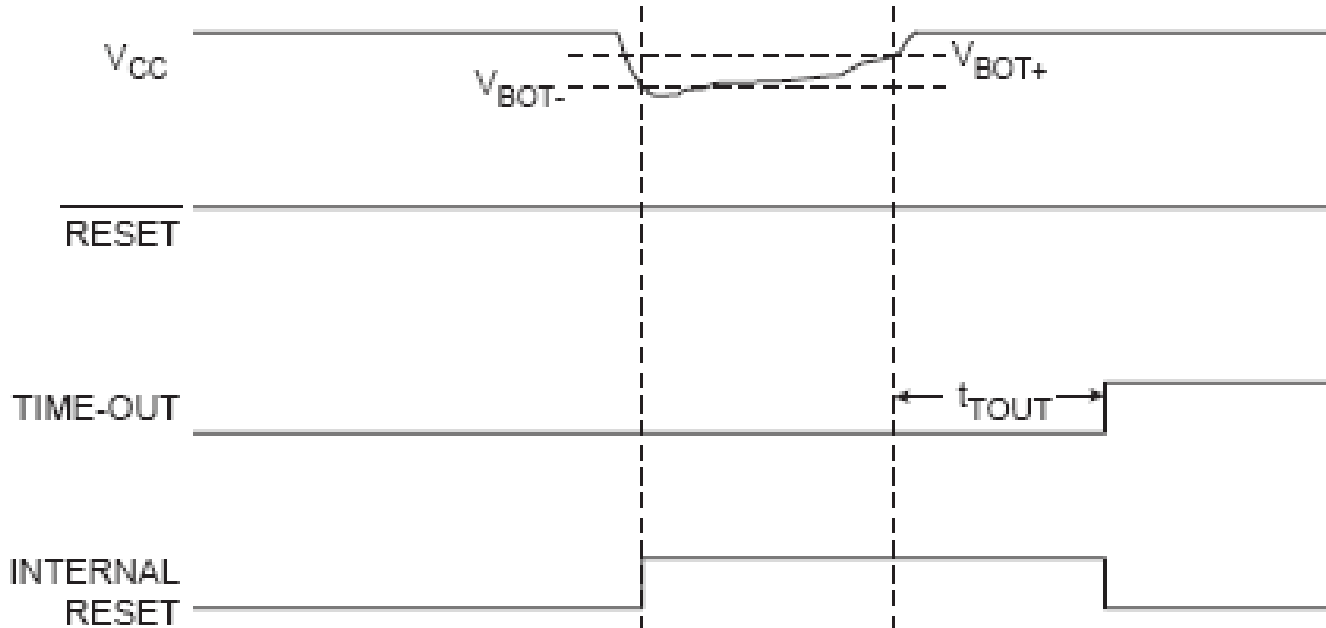
Mikrokontroler jest resetowany kiedy na wejściu RESET (zanegowane) jest niski poziom dłużej niż minimalna długość impulsu. Możemy programowo zresetować mikrokontroler podłączając jeden z pinów IO do wejścia RESET.

Źródła sygnału RESET

Reset przy obniżonym napięciu zasilania (Brown-out detector)

Układ ten monitoruje napięcie zasilające. Jeśli napięcie spadnie poniżej pewnego ustalonego poziomu (poprzez fusebity), układ generuje sygnał RESET, aż do czasu powrotu napięcia do prawidłowego zakresu.

Uwaga: obniżenie napięcia może skutkować np.uszkodzeniem pamięci EEPROM.



Źródła sygnału RESET

Układ Watchdog

- Niezależnie działający podsystem procesora, posiadający własny niezależny zegar.
- Po skonfigurowaniu i uaktywnieniu działa jak licznik. Po zliczeniu do określonej wartości generuje sygnał RESET.
- Aby temu zapobiec, program musi co pewien czas wykonać instrukcję zerującą licznik.
- Wykorzystanie układu Watchdog krytyczne w układach działających bez ingerencji człowieka.
- Układ po zadziałaniu może generować również przerwanie (możemy je wykorzystać do obsługi niewłaściwego działania programu).

Zarządzanie poborem energii

Aby zredukować pobór energii można wyłączyć część podsystemów mikrokontrolera.

Tryb uśpienia wybiera się przy pomocy bitów SM0-SM2 rejestru SMCR.

6 trybów uśpienia:

Idle Mode : wyłączany rdzeń (brak generowania sygnału taktowania CPU i FLASH), działają układy peryferyjne SPI, USART, ADC, TWI, liczniki, Watchdog i system przerwań. CPU jest wybudzany przez przerwanie zewnętrzne, przerwanie licznika lub przerwanie USART.

ADC Noise Reduction Mode : Wszystkie podsystemy działają normalnie. Wyłączany jest rdzeń procesora, pamięć FLASH i układy portów IO, co zmniejsza zakłócenia, ułatwiając pomiar ADC.

Zarządzanie poborem energii

Power-down Mode : Wyłącza większość podsystemów procesora i zewnętrzny oscylator. Działają tylko BOD, Watchdog, TWI i przerwania zewnętrzne.

Power-save Mode : działają tylko timery.

Standby Mode : identyczny jak Power-down, tyle że działa oscylator (przyśpiesza to wybudzenie – z kilku tysięcy do kilku taktów zegara).

Extended Standby Mode : identyczny jak Power-save, tyle że działa oscylator (przyśpiesza to wybudzenie – z kilku tysięcy do kilku taktów zegara).

Średnie zużycie prądu
(zegar 8MHz, $V_{cc}=5V$)

Active	12 mA
Idle	6 mA
Power-down	15 uA
Power-save	12 uA
Standby	140 uA

Zarządzanie poborem energii

- Ilość zużywanej przez układy cyfrowe energii zależy od częstotliwości ich przełączania, dlatego należy wybrać możliwie niską częstotliwość zegara (m.in. poprzez fusebity).
- Stan portów IO powinien być ustalony. Wymusić stan wejść przyłączając je do masy lub zasilania. Pływanie potencjału na wejściu powoduje ciągłe przełączanie stanu i zwiększony pobór prądu.
- Wyłączyć przetwornik ADC, gdy nie jest używany.
- Wyłączyć komparator analogowy.

Fusebity – bity konfiguracyjne

Niektóre parametry dotyczące konfiguracji fizycznej pracy mikrokontrolerów, można ustawiać nie w sposób programowy (bezpośrednio z kodu programu), lecz w trakcie procesu programowania.

- Sposób generowania sygnału zegarowego (zewn., wewn., rezonator kwarcowy).
- Częstotliwość taktowania (domyślnie wewn. zegar taktujący jest dzielony przez 8 – uwaga na pomyłki)
- Odblokowanie interfejsów ISP i JTAG (do programowania) – uwaga na pomyłki
- Poziom brown-out detektora
- Włączenie watchdoga
- Zabezpieczenie pamięci EEPROM podczas programowania
- Wydzielenie obszaru pamięci na bootloader

Bootloader

Mikrokontrolery serii ATMega wyposażone zostały w instrukcje pozwalające programować pamięć Flash z poziomu aplikacji. Dzięki temu istnieje możliwość zmiany oprogramowania w mikrokontrolerze poprzez dowolny, wybrany interfejs.

Gdy używamy bootloadera, pamięć procesora jest podzielona na dwie części Read-While-Write (RWW) oraz No Read-While-Write (NRWW). W przypadku kiedy nie korzystamy z samoprogramowania takiego podziału nie ma.

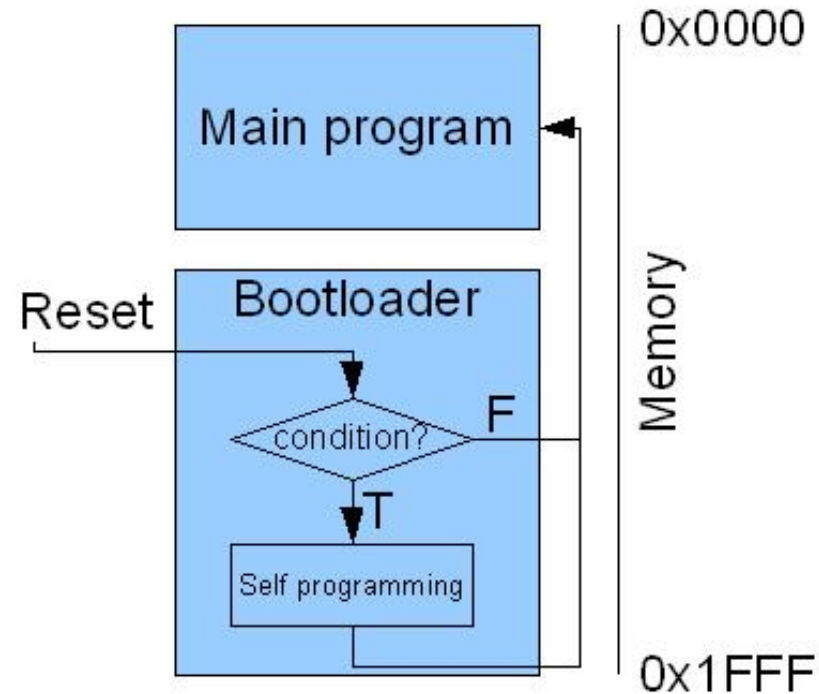
Podczas zapisu RWW praca procesora nie zostaje zatrzymana i możliwe jest działanie programu pod warunkiem, że znajduje się on w NRWW. W czasie ładowania programu do RWW **nie można** odwoływać się do komórek pamięci znajdującej się w tej sekcji, ponieważ może to spowodować zawieszenie układu i błąd przy zapisie.

Zapis obszaru NRWW wiąże się z zatrzymaniem pracy CPU.

Read-While-Write (RWW)	\$0000 Koniec RWW
No Read-While-Write (NRWW)	Początek NRWW Koniec pamięci

Bootloader

- Za pomocą fusebitów możemy przesunąć adres startowy programu z 0x0000 do sekcji bootloadera.
- Po włączeniu zasilania, program bootloadera może oczekiwać na sygnał z dowolnego interfejsu, np. USART.
- Gdy dane zaczną napływać, bootloader wgra je w odpowiedniej kolejności do obszaru RWW.
- Po zakończeniu zapisywania danych, nastąpi skok pod adres 0x0000 i rozpocznie działanie normalny program.



Możemy również skonfigurować mikrokontroler tak, by przenieść adresy wektorów przerwań w obszar NRWW. Przeniesienie przerwań daje nam pewność, że nie odbędzie się skok do RWW (co może się stać, gdy nie zablokujemy przerwań) podczas jej programowania, co może spowodować błąd zapisu.

Zabezpieczanie kodu programu

- Wśród bitów konfiguracyjnych są tzw. lock bity.
- Ich funkcją jest ochrona pamięci mikrokontrolera przed możliwością jej odczytania przy pomocy programatora.
- Przy próbie odczytu zwracane będą adresy komórek (a nie wartości).
- Aby skasować lockbity, należy skasować całą pamięć FLASH i EEPROM.

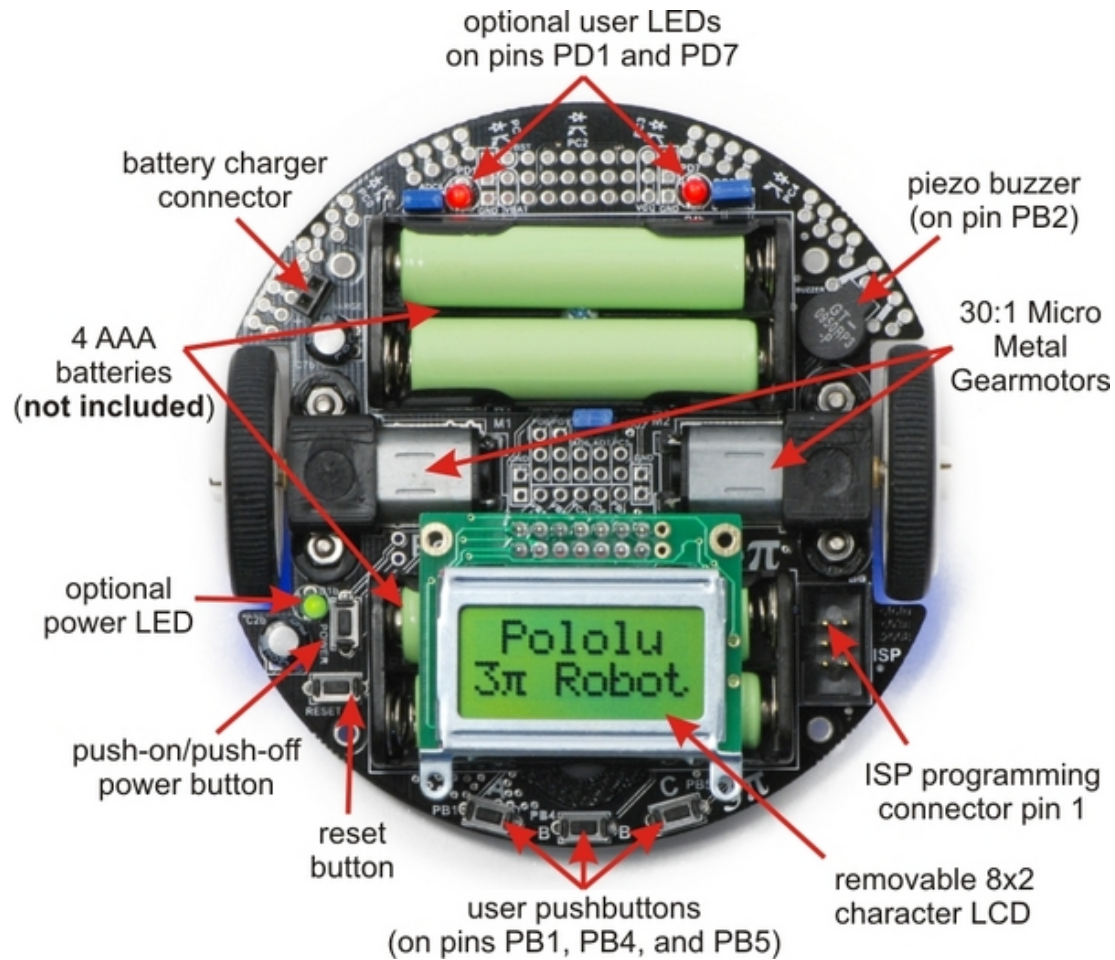
Inwazyjne metody łamania zabezpieczeń:

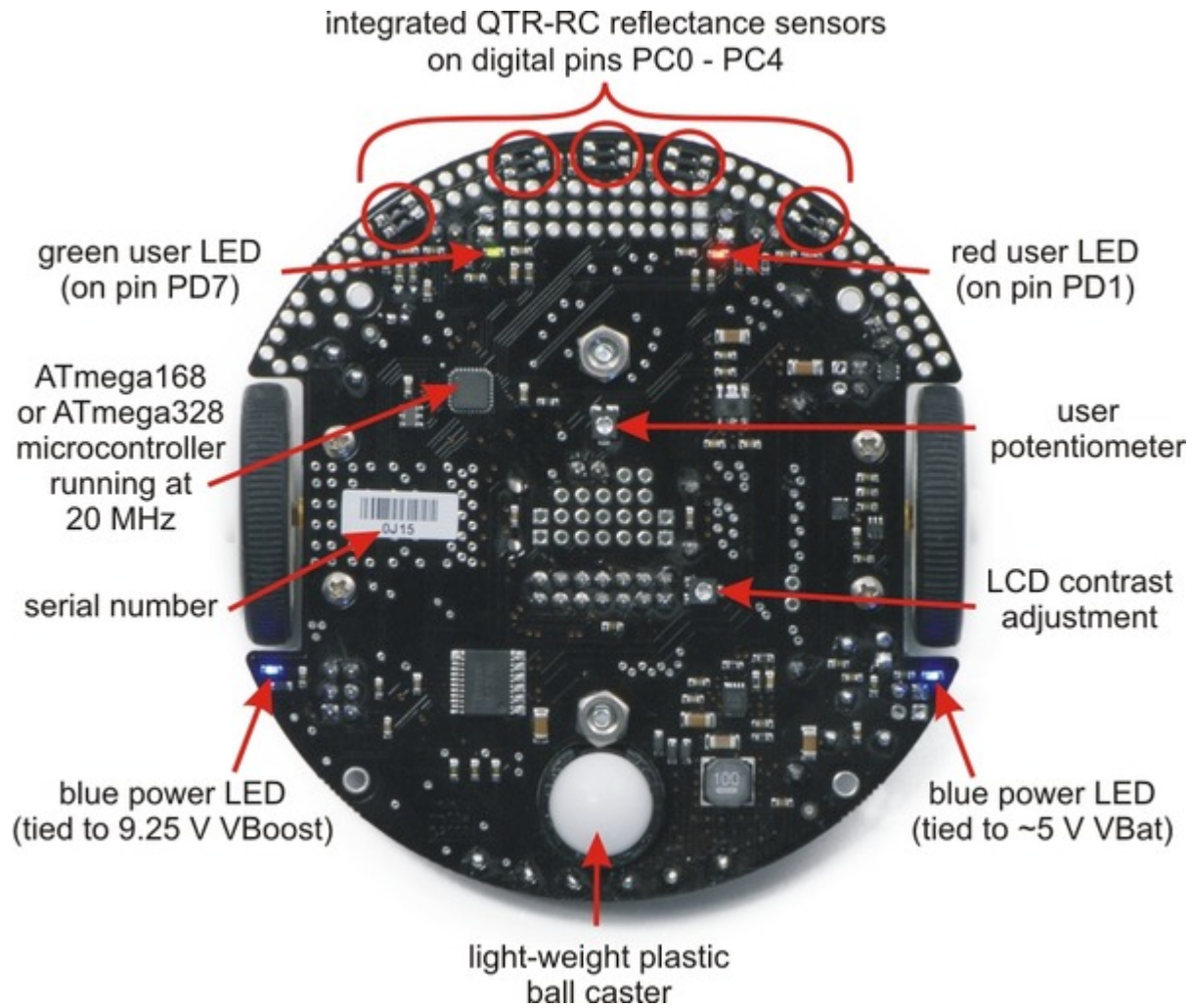
- odsłonięcie struktury układu scalonego i odczytanie danych za pomocą mikrosond
- przeprogramowanie bitów zabezpieczających (naświetlanie UV)

Nieinwazyjne metody łamania zabezpieczeń:

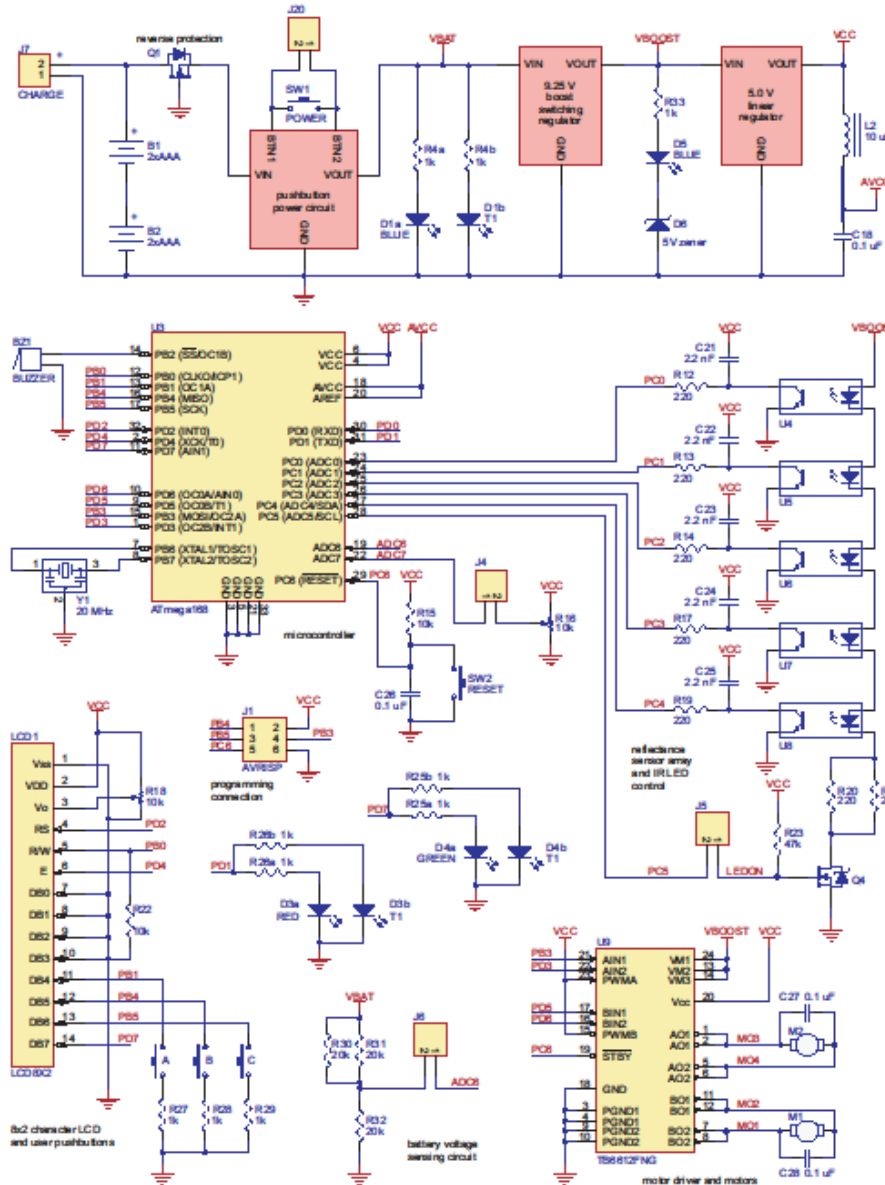
- Typowe – luki w oprogramowaniu
- Eavesdropping – monitorowanie z wysoką rozdzielczością sygnałów analogowych (np. poboru prądu przez procesor).
- Każda instrukcja przełączając inne bramki logiczne zostawia swój odcisk palca w postaci zakłóceń na nóżkach układu.

Przykład to jest ostatni...

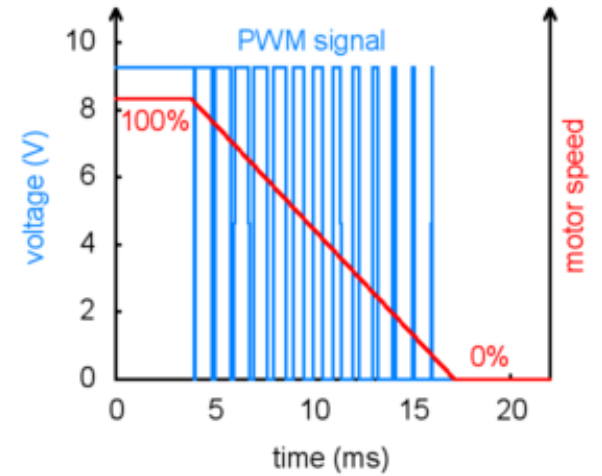
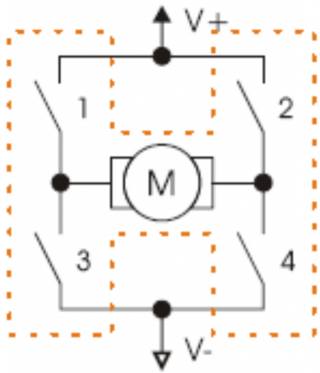




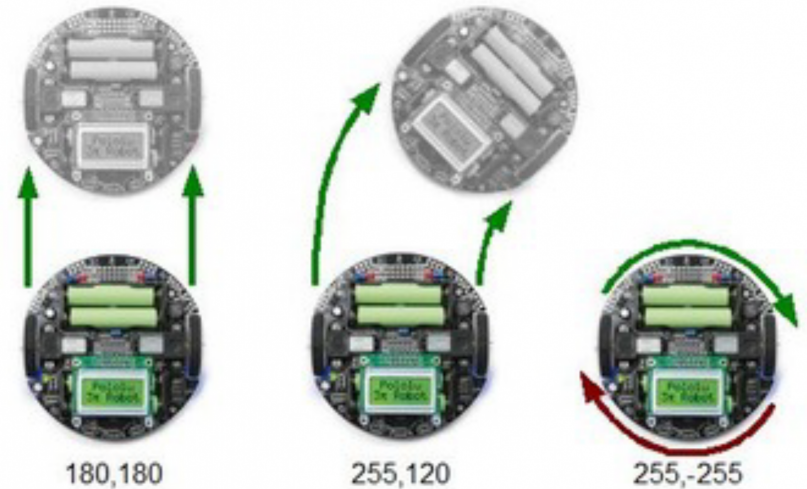
Pololu 3pi Robot Simplified Schematic Diagram



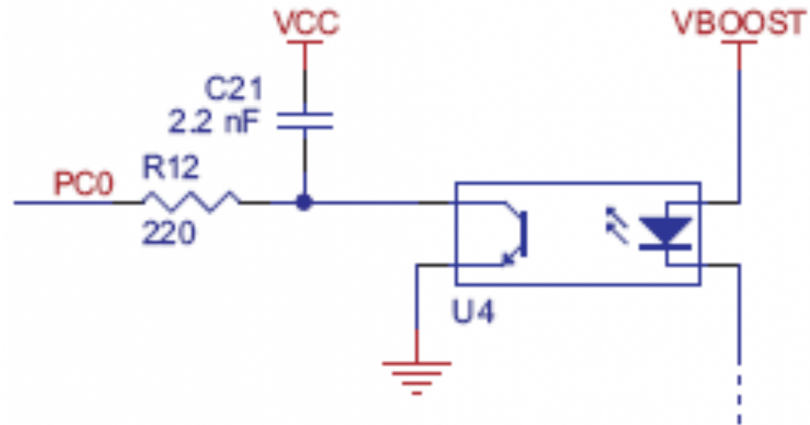
Silniki



PD5	PD6	1	2	3	4	M1
0	0	off	off	off	off	off (coast)
0	1	off	on	on	off	forward
1	0	on	off	off	on	reverse
1	1	off	off	on	on	off (brake)

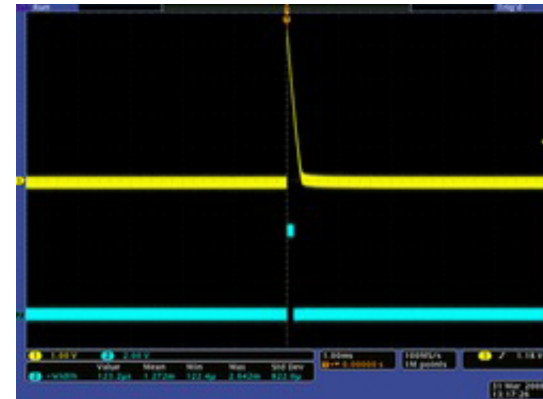
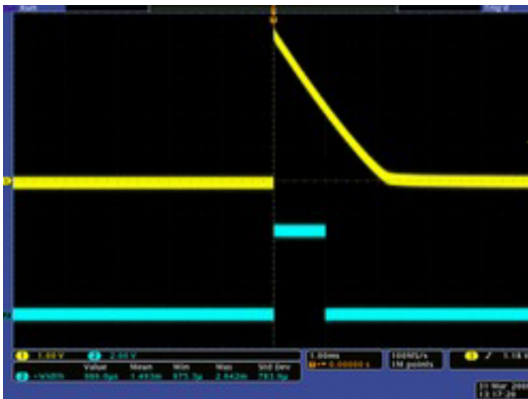
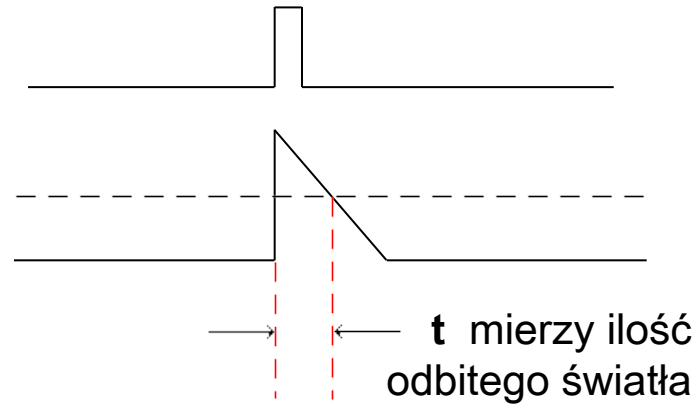


Sensory



1. Wyjście: 10 μ s impuls na PC0

2. Wejście: mierzymy czas



DEMO