ATmega 128

Wojciech Glądała
Tomasz Kopeć
Łukasz Przepióra
Tomasz Tokarski
Piotr Zych

CECHY

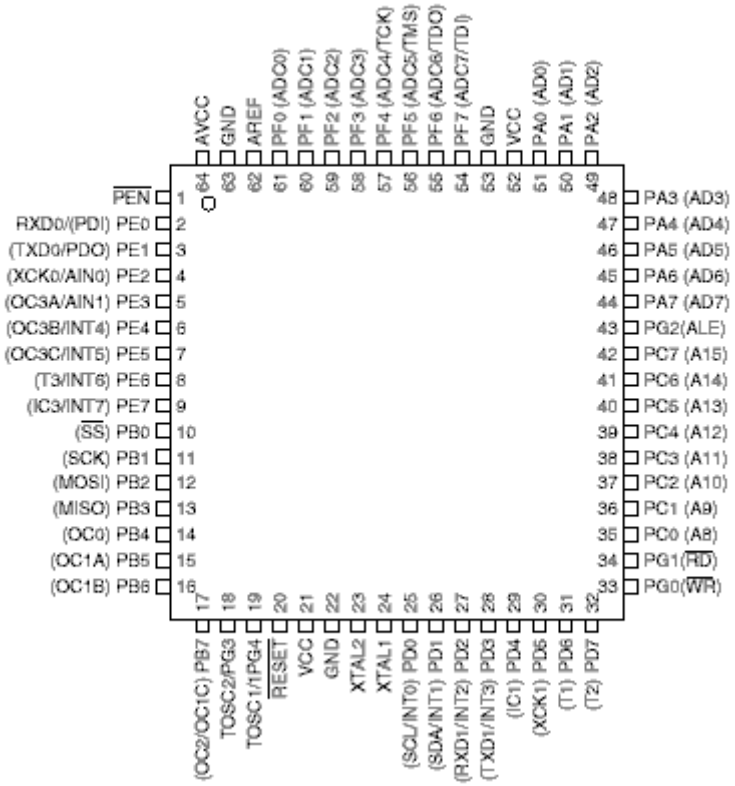
- Wysoka wydajność, niski pobór mocy AVR 8 bitowy mikrokontroler
- Zaawansowana architektura RISC
  - 133 rozkazy – większość wykonywana w czasie jednego cyklu maszynowego
  - 32 x 8 rejestry ogólno dostępne + rejestr kontroli zewnętrznej
  - statyczne operacje
  - do 16 MIPS przy 16MHz
  - wbudowany dwu cyklowy mnożnik
- trwała pamięć programu i danych
  - 128KB wewnętrznej reprogramowalnej Flash o wytrzymałości 1000 cykli zapisy/wymazania
  - opcjonalna sekcja programu ładującego z niezależnymi bitami blokującymi, wewnętrzne programowanie oraz operacje odczytu podczas zapisu
  - 4KB EEPROM o wytrzymałości 100 000 cykli zapisu/wymazania
  - 4KB wewnętrznego SRAM
  - do 64KB opcjonalnej zewnętrznej przestrzeni pamięci
  - blokada programowania dla bezpieczeństwa programów
  - interfejs SPI dla systemowego programowania
- interfejs JTAG
  - skanowanie graniczne w poszukiwaniu zgodności ze standardem JTAG
  - rozległe wbudowane wsparcie debugera
  - programowanie Flash, EEPROM, bezpieczników i bitów blokujących przez interfejs JTAG
- cechy peryferyjne
  - dwa 8 bitowe liczniki/stopery z oddzielnymi prescalerami i trybami porównywania
  - dwa rozszerzone 16 bitowe liczniki/stopery z oddzielnymi prescalerami, trybami porównywania i trybami schwytania
  - licznik czasu rzeczywistego z oddzielnym oscylatorem
  - dwa 8 bitowe kanały PWM
  - 6 kanałów PWM z programowalną rozdzielczością od 2 do 16 bitów
  - wyjściowy modulator porównujący
  - 8 kanałowy 10 bitowy ACD
    - 8 sygnałów zakończenia kanału
    - 7 różnych kanałów
    - 2 różne kanały z programowalnym przyrostem 1x, 10x lub 200x
  - bajtowo zorientowany dwuprzewodowy szeregowy interfejs
  - podwójny programowalny szeregowy USART
  - interfejs szeregowy SPI podrzędny/nadrzędny
  - programowalny licznik Watchdog'a z wbudowanym oscylatorem
  - wbudowany analogowy komparator
- specjalne cechy mikrokontrolera
  - reset przy włączaniu zasilania, programowalny detektor obniżonego napięcia sieciowego
  - wewnętrzny wzorcowy oscylator RC
  - wewnętrzne i zewnętrzne źródła przerwań

- sześć trybów snu: Idyl, redukcja szumów ADC, oszczędność mocy, obniżony pobór mocy, oczekiwanie i rozszerzone oczekiwanie
- częstotliwość zegara wybierana przez oprogramowanie
- tryb kompatybilności z ATmega103 wybierany poprzez bezpiecznik
- globalne zablokowanie podciągania

- I/O i packages
  - 53 programowalnych linii I/O
  - 64- TQFP
- napięcia pracy
  - 2.7 – 5.5 dla ATmega128L
  - 4.5 –5.5 dla ATmega128
- klasa szybkości
  - 0-8 MHz ATmega128L
  - 0 – 16 MHz ATmega128

KONFIGURACJA NÓŻEK

Figure 1. Pinout ATmega128

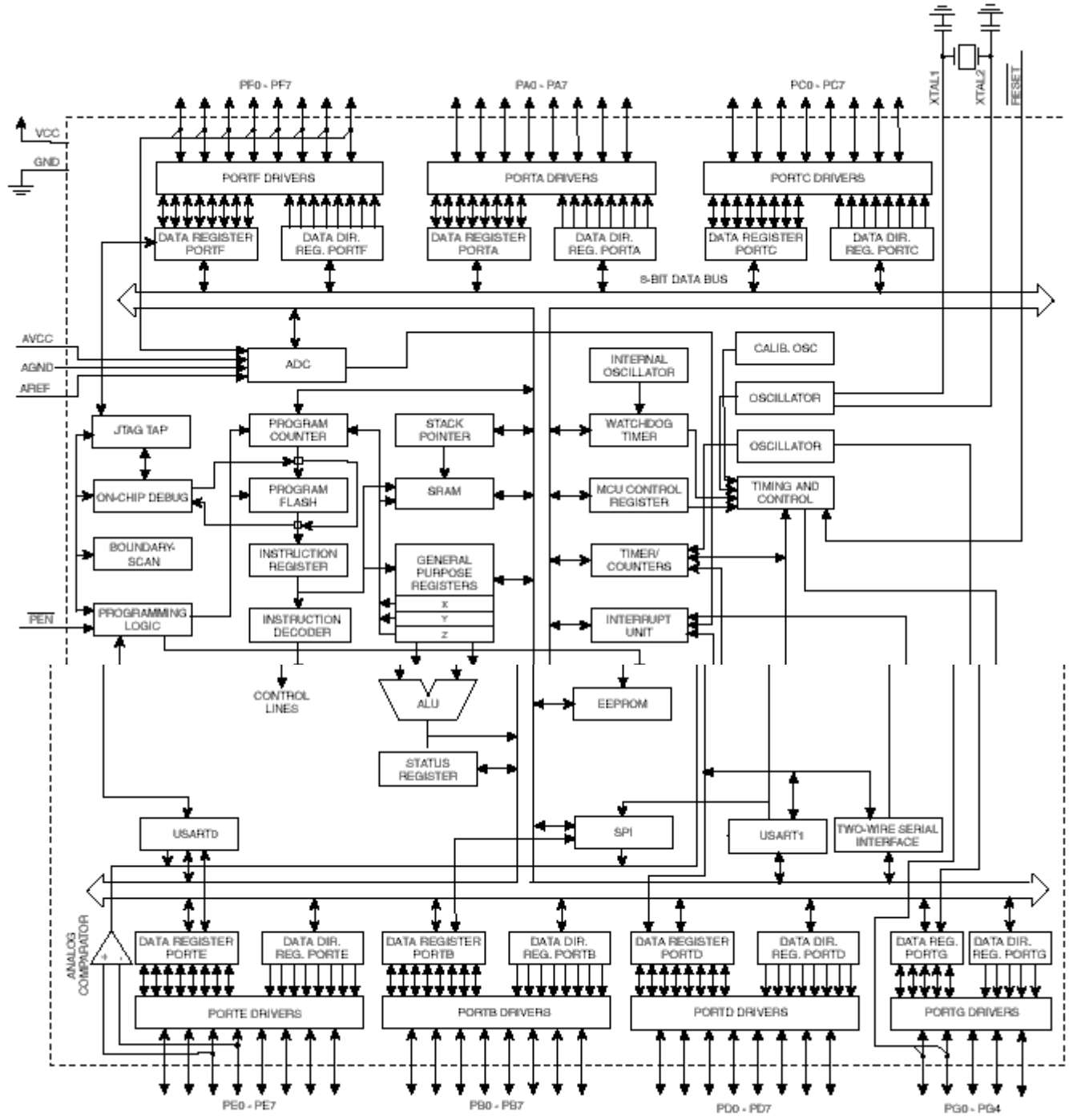


KRÓTKI OPIS

ATmega128 jest 8 bitowym mikrokontrolerem o niskim poborze mocy opartym na architekturze AVR z rozszerzonym RISC. Poprzez wykonywanie instrukcji w jednym cyklu zegarowym, ATmega128 osiąga rezultat w okolicy 1MIPS na MHz, co pozwala poprzez projektowanie systemu zoptymalizować pobór mocy z jednoczesnym przyspieszeniem działania

- **Diagram blokowy**

Figure 2. Block Diagram



Rdzeń AVR zawiera bogaty zestaw instrukcji z 32 rejestrami dostępnymi do różnych celów. Wszystkie 32 rejestry są bezpośrednio podłączone do Jednostki Arytmetyczno Logicznej (ALU), co pozwala na jednoczesny dostęp do dwóch niezależnych rejestrów w jednej instrukcji jednego taktu maszynowego. Taka architektura jest bardzie efektywna dla kodu obsługi oraz pozwala nawet na dziesięciokrotnie szybszą pracę w porównaniu z konwencjonalnymi mikrokontrolerami CISC'owymi.

Atmega128 zapewnia następujące elementy: 128K bajtów wewnętrznej programowalnej pamięci typu FLASH z dopasowaniem odczytu podczas zapisu, 4K bajty EEPROM, 4K bajty SRAM, 53 linie wejścia/wyjścia, 32 robocze rejestry, Licznik rzeczywistego czasu(RTC), cztery liczniki z trybami porównywania i PWM, dwa USART, bajtów zorientowany dwu-kablowy równoległy interfejs, 8 kanałów, 10 bitowy ADC z opcjonalnym różnicowym progiem wejściowym z programowalnym przyrostem, programowalny licznik Watchdog z wewnętrznym oscylatorem, SPI równoległy port, IEEE standard 1149,1 ze zgodnym JTAG testowym interfejsem, również używanym przy dostępie do wbudowanego systemu debugowania i z sześcioma programowalnymi trybami oszczędności mocy. Tryb nieaktywny (Idle) zatrzymuje CPU pozwalając SRAM, licznikom, portom SPI i systemowi przerwań funkcjonować. Tryb obniżonego poboru mocy (Power-Down) zachowuje zawartość rejestrów, ale zatrzymuje Oscylator, uniemożliwiając wykonywanie innych funkcji aż do następnego przerwania lub resetu sprzętu. Tryb oszczędzający moc (Power save) asynchroniczny zegar nie przestaje działać podczas, gdy reszta sprzętu jest nieaktywna. Redukcja zakłóceń ADC zatrzymuje CPU i wszystkie wyjściowo-wejściowe moduły z wyjątkiem asynchronicznego zegara i ADC w celu zmniejszenia hałasów podczas konwersji ADC. W trybie oczekiwania (Standby) zarówno oscylator jak i asynchroniczny licznik działają.

Urządzenie to zostało wyprodukowane przy użyciu Atmelowskiej najnowszej wysoko upakowanej trwałej technologii tworzenia pamięci. Wbudowany ISP FLASH pozwala na reprogramowanie w systemie pamięci mikrokontrolera poprzez szeregowy interfejs SPI, poprzez tradycyjne trwałe programowanie pamięci, lub poprzez wbudowany program ładujący działający na rdzeniu AVR. Program ładujący może korzystać ze wszystkich tych interfejsów w celu załadowania programu do pamięci FLASH. Oprogramowanie w ładowalnej części Flash będzie działało podczas aktualizacji części programów tej pamięci zapewniającej prawdziwe operacje odczytu podczas zapisu. Poprzez połączenie 8-bitowego RISC'owego CPU z wewnętrznym systemowym samo-programowalnej pamięci FLASH na jednej płytce, Atmelowski Atmega128 jest mikrokontrolerem, który zapewnia łatwą dopasowalność wraz z efektywnymi rozwiązaniami wiążącymi kontrole aplikacji.

Atmega128 AVR jest zaopatrzony w pełną gamę programów wspomagających rozwój: kompilatory C, assembler makro, debuger, emulatory itp.

DOPASOWANIE ATMEGA103 DO ATMEGA128

ATmega128 jest kompleksowym mikrokontrolerem gdzie liczba umiejscowień I/O została zastąpiona 64-oma lokalizacjami I/O zarezerwowanymi dla zbioru rozkazów AVR. W celu zapewnienia kompatybilności z ATmega103, wszystkie I/O lokalizacje obecne w ATmega103 są w tym samym miejscu w ATmega128. Większość dodatkowych lokalizacji I/O zastała dodana w rozszerzonej przestrzeni adresowej od $60 do $FF. Do tych miejsc pamięci można się odwoływać tylko poprzez instrukcje LD/LDS/LDD oraz ST/STS/STD, a nie prze rozkazy IN i OUT. Ta wewnętrzna realokacja przestrzeni pamięci RAM może nadal być problemem dla użytkowników ATmega103. Również zwiększona liczba wektorów przerwań może sprawiać problem dla kodów operujących na bezwzględnym adresowaniu. W celu rozwiązania tego programu tryb dopasowania ATmega103 może zostać wybrany poprzez programowanie bezpiecznika M103C. W tym trybie żadna z funkcji w rozszerzonej przestrzeni adresowej jest niedostępna, jak i wewnętrzna pamięć RAM jest zlokalizowana jak w ATmega103. Również, rozszerzony wektor przerwań jest usunięty.

100% wyjść z ATmega128 jest kompatybilne z wyjściami z ATmega128 i może zastępować ATmega103 na płytkach drukowanych.

TRYBY KOMPATYBILNOŚCI ATMEGA103

Poprzez programowanie bezpiecznika M103C ATmega128 jest kompatybilna z ATmega103 zarówno w stosunku do RAM jak i nóżek I/O, wektorów przerwań. Jednakże, niektóre z tych rzeczy w ATmega103 nie są dostępne w tym trybie dopasowania, są to:

- jeden USART zamiast dwóch, tylko w trybie asynchronicznym. Tylko 8 najmniej znaczących bitów w rejestrze szybkości transferu jest dostępne
- jeden 16-bitowy licznik z dwoma rejestrami porównawczymi, zamiast dwóch liczników z trzema rejestrami
- dwu-drutowy równoległy interfejs nie jest obsłużone
- port C jest tylko wyjściowy
- port G służy różnym/(alternatywnym) funkcjom – nie jest to całościowo port I/O
- port F służy tylko jako bitowy wejściowy port jako dodatek do analogowego wejścia ADC
- program ładujący nie jest obsłużony
- Nie można dostosować częstotliwość wewnętrznego RP oscylatora
- Rozszerzony interfejs pamięci nie może zwolnić żadnych adresów dla I/O, jak również skonfigurować różnych stanów oczekiwania dla różnych sekcji rozszerzonego adresu pamięci

W dodatku, jest tu jeszcze parę mniej znacznych różnic, aby zapewnić lepszą kompatybilności z ATmega103:

- tylko EXTRF i PORF istnieją w MCUCSR
- sekwencje czasowe nie wymagają dla Watchdog'a zmiany czasu oczekiwania
- Zewnętrzne nóżki przerwań 0-3 służą tylko jako stopnie przerwań
- USATR nie ma buforu FIFO, więc do przekroczenia liczby danych dochodzi wcześniej

Nieużywane bity I/O w ATmega103 powinny być 0, aby zapewnić takie same operacje w ATmega128.

OPIS NÓŻEK

- VCC – zasilanie
- GND – masa
- Port A (PA0-PA7) – 8-bitowy dwukierunkowy port I/O z wewnętrznymi rezystorami podciągającymi (dla każdego bitu). Bufory wyjściowy mają symetryczne charakterystyki sterowników zarówno z wysokim opadającym poziomem i dopasowaniem źródła. Na wyjściu jest stan niski i pobierają prąd jeżeli rezystory podciągające są aktywowane. Wyjścia z portu po wykonaniu resetu są w trzecim stanie nawet jeżeli zegar nie działa.

  Dodatkowe informacje strona 67.

- Port B (PB0-PB7) – opis jak wyżej. Dodatkowe informacje na stronie 68
- Port C(PC0-PC7) – opis jak wyżej. Dodatkowe informacje na stronie 71. W trybie dopasowania do ATmega103, jest to wyłącznie port wyjściowy i wyjścia nie są w trzecim stanie dla aktywnego resetu.
- Port D (PD0-PD7) – jak wyżej. Dodatkowe informacje na stronie 72
- Port E (PE0-PE7) - jak wyżej. Dodatkowe informacje na stronie 78.
- Port F (PF0-PF7) – służy jako analogowe wejście dla A/D konwertera. Jeśli nie pracuje w tym trybie działa jak powyższe porty. Jeżeli jest załączony interfejs JTAG rezystory podciągające będą uaktywnione dla PF7(TDI), PF5(TMS) i PF4(TCK) nawet jeżeli dla sygnału reset.

W trybie zgodności z Tamega103 jest to tylko port wejściowy.

- Port G (PG0-PG7) – od powyższych portów różni się tym tylko, że jest 5-bitowy
  W trybie dopasowania do ATmega103 wyjścia te służą tylko jako sygnały do zewnętrznej pamięci zarówno jak wejścia do 32 kHz oscylatora. Wyjścia te są inicjalizowane asynchronicznie PG0 = 1, PG1 = 1, PG2 = 0 kiedy zostanie wykonana operacja reset, nawet przy niedziałającym zegarze. Pg3 i PG 4 są wejściami do oscylatora.
- RESET – wejście. Dla sygnału niskiego trwającego dłużej niż minimalny takt cyklu zostanie wygenerowany rozkaz restartu nawet przy niedziałającym zegarze, Długość minimalnego impulsu jest podana w tabeli na stronie 46 w tabeli 19. Krótsze impulsy nie gwarantują wygenerowania restartu.
- XTAL1 – ujemne wejście do wzmacniacza oscylatora , oraz do wewnętrznego zegara.
- XTAL2 – ujemne wyjście ze wzmacniacza oscylatora
- AVCC – zasób napięcia dla portu F i A/D konwertera. Powinien być podłączony do zewnętrznego napięcia, nawet jeżeli ADC nie jest używany. Jeżeli ADC jest używane powinien być podłączony do napięcia przez filtr dolnoprzepustowy.
- AREF – jest to analogowy odpowiednik dla wejścia A/D konwerter.
- PEN - wejście umożliwiające programowanie dla SPI – tryb szeregowego programowania . Przyciskając to wyjście podczas resetu w trybie uruchamiania, urządzenie przejdzie do trybu SPI. W czasie normalnych operacji wejście to nie ma żadnych funkcji.

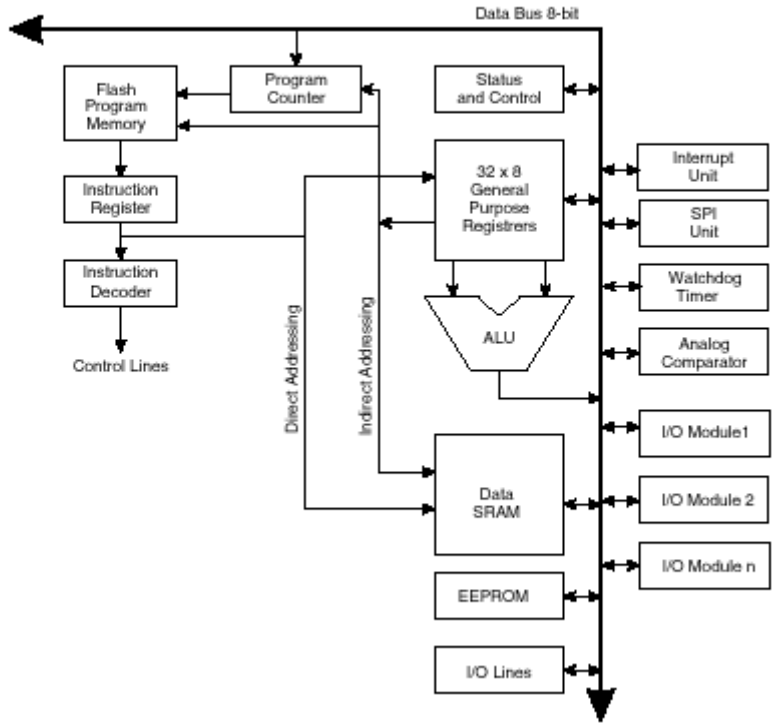
PRZYKŁADY KODU

Ten arkusz danych zawiera proste przykłady kodu, które w prosty sposób obrazują jak wykorzystywać różne części urządzenia. Kody te zakładają, że specyficzne nagłówki zostały dołączone przed kompilacją. Należy być świadomym tego, że nie wszystkie sprzedawane kompilatory C zawierając definicje bitów w plikach nagłówkowych i obsługa procedur przerwań w C jest zależna od kompilatora. W celu uzyskania dodatkowych informacji proszę przejrzeć dokumentację kompilatora C.

JEDNOSTKA CENTRALNA

- **Architektura:**

Figure 3. Block Diagram of the AVR Architecture



W celu zwiększenia wydajności i równoległości, AVR korzysta ze architektury Harvard – z oddzielnymi pamięciami i magistralami dla programów i danych. Instrukcje w programach są wykonywane w sposób jednopotokowy. W czasie wykonywania jednej instrukcji, następna jest wstępnie pobierana z pamięci programu. W ten sposób umożliwia wykonywanie instrukcji w każdym cyklu maszynowym. Program jest zapisany w wewnętrznej systemowej reprogramowalnej pamięci Flash.

Zestaw rejestrów szybkiego dostępu składa się z 32 8-bitowych ogólnych rejestrów roboczych o czasie dostępu równemu jednemu cyklowi maszynowemu. To pozwala na jednocyklowe operacje na Jednostce Arytmetyczno Logicznej (ALU). W typowych operacjach na ALU dwa operandy są pobierane z rejestrów, wykonywana jest operacja i jej rezultat jest zapisywany do jednego ze zestawu rejestrów w jednym takcie maszynowym.

Sześć z tych 32 rejestrów może być używanych jako 16-bitowe wskaźniki z adresami pośrednimi do przestrzeni danych – co umożliwia efektywne operowanie na adresach. Jeden z tych wskaźników adresów może również być używany jako wskaźnik na adres w tablicy przeglądowej pamięci Flash. Te dodatkowe funkcje dotyczą szesnastobitowych rejestrów X, Y i Z, które zostaną opisane później.

ALU wspiera arytmetycznie i logiczne operacje pomiędzy rejestrami lub pomiędzy stałą i rejestrem. Operacje na pojedynczych rejestrach również mogą być wykonywane w ALU. Po operacji arytmetycznej, rejestr stanu jest aktualizowany by odzwierciedlać rezultat operacji.

Przepływ programu jest możliwy dzięki warunkowym i bezwarunkowym rozkazom skoku (jump) i przywołania (call) zdolnych do bezpośredniego zaadresowania całej przestrzeni adresowej. Większość rozkazów AVR ma 16-bitowy format słowa. Każdy adres pamięci programu zawiera 16-to lub 32-bitowy rozkaz.

Pamięć programowa Flash jest podzielona na dwie sekcje: Program ładujący i Program użytkowy. Obydwie sekcje mają wydzielone bity dla ochrony zapisu lub zapisu i odczytu. Instrukcja SMP, która zapisuje do części użytkowej pamięci Flash musi rezydować w sekcji Programu ładującego.

Podczas przerwań i wywołań programu standardowego adres powrotu z Licznika Program(PC) jest odkładany na stos. Stos jest alokowany w pamięci SRAM i jego rozmiar jest limitowany przez rozmiar tej pamięci. i jej użytkowanie. Wszystkie programy użytkowe muszą inicjalizować wskaźnik stosu (SP) podczas procedury RESET (zanim przerwania lub inne programy zostaną wykonane). Wskaźnik stosu jest dostępny zarówno do odczytu jak i zapisu w przestrzeni I/O. Dane w SRAM mogą być łatwo dostępne poprzez pięć różnych trybów adresowania wspieranych przez architekturę AVR..

Przestrzeń pamięci w architekturze AVR jest linearna i regularna mapa pamięci.

Elastyczny moduł przerwań ma swoje rejestry kontrolne w przestrzeni I/O z dodatkowymi globalnymi bitami przerwań w rejestrze stanu. Wszystkie przerwania mają osobny wektor przerwania w tablicy wektorów przerwań. Przerwania mają priorytety zgodnie z rozmieszczeniem ich w tablicy wektorów przerwań. Im niższy adres wektora przerwań tym wyższy jego priorytet.

Przestrzeń adresowa I/O zawiera 64 adresów, które mogą być dostępne bezpośrednio lub jako lokacje przestrzeni danych następujące po tych określających zestaw rejestrów, $20 - $4F. Dodatkowo ATmega128 ma rozszerzoną przestrzeń I/O od $60-$FF w SRAM, gdzie dostęp mają tylko rozkazy: ST/STS/STD i LD/LDS.LDD.

- **Jednostka Arytmetyczno Logiczna – ALU:**

Wysokiej klasy ALU pracuje z bezpośrednim połączeniem ze wszystkimi 32 uniwersalnymi roboczymi rejestrami. Podczas jednego cyklu maszynowego wykonywane są operacje arytmetyczne pomiędzy tymi rejestrami lub między rejestrem i stałą. Operacje wykonywane przez ALU można podzielić na trzy główne kategorie: arytmetyczne, logiczne i bitowe. Niektóre implementacje tej architektury zapewniają również mnożenie ze znakiem i format ułamkowy. Dokładny opis w spisie rozkazów.

- **Rejestr Stanu (SR)**

Rejestr stanu zawiera informacje o rezultacie ostatniej operacji arytmetycznej. Ta informacja może zostać wykorzystana poprzez różne programy do wykonania operacji warunkowych. Należy zwrócić uwagę, że rejestr stanu jest aktualizowany po każdej operacji na ALU. TO w wielu przypadkach sprawia, że niepotrzebne jest używanie wyspecjalizowanych instrukcji porównywania, co wpływa na przyspieszenie działania kodu i bardziej zwięzły kod.

Rejestr stanu nie jest automatycznie zapamiętywany kiedy procesor przechodzi do obsługi przerwania i ustawiany podczas powrotu z przerwania. To musi zostać obsłużone przez oprogramowanie.

Rejestr stanu – SREG – jest zdefiniowany jako:

Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
InVal	0	0	0	0	0	0	0	0

R/W – odczyt/zapis
InVal – wartość inicjująca

a. *Bit 7 – I: Globalne przerwanie*

Bit ten musi być ustawiony na 1 dla przerwań. Indywidualna kontrola przerwań jest wtedy wykonywana w oddzielnym rejestrze kontrolnym. Jeżeli globalny rejestr przerwań jest wyzerowany, żadne z przerwań nie może zostać obsłużone niezależnie od indywidualnych ustawień przerwań. I-bit jest wyzerowany sprzętowo po zajściu przerwania i jest ustawiany przez rozkaz RETI, aby można było obsługiwać następne przerwania. I-bit może zostać również programowo wyzerowany poprzez rozkazy SEI i CLI.

b. *Bit 6 – T: odpis pamięci*

Rozkazy kopiowania bitu BLD i BST korzystają z bitu T jako źródło lub bit docelowy dla zmienianego bitu. Rozkaz BST kopiuje bit z rejestru do bitu T, natomiast rozkaz BLD kopiuje bit T do rejestru w zestawie rejestrów.

c. *Bit 5 – H: flaga połowicznego przeniesienia*

Flaga ta wskazuje przeniesienie w niektórych operacjach arytmetycznych. Flaga ta jest wykorzystywana przy arytmetyce BCD.

d. *Bit 4 – S: bit znaku S= N + V (+ exclusive or)*

Bit ten jest zawsze alternatywą pomiędzy zanegowanym N i dwójkowym uzupełnieniem flagi V.

e. *Bit 3 V: dwójkowe dopełnienie*

Flaga ta wspiera dwójkową arytmetykę .

f. *Bit 2 N: flaga zaprzeczenia*

Flaga ta wskazuje na ujemny rezultat w operacjach logicznych lub arytmetycznych.

g. *Bit 1 Z: flaga Zero*

Flaga ta wskazuje zerowy rezultat w operacjach arytmetycznych i logicznych.

h. *Bit 0 – C: flaga przeniesienia*

Flaga przeniesienia wskazuje na przeniesienie podczas operacji logiczej lub arytmetycznej.

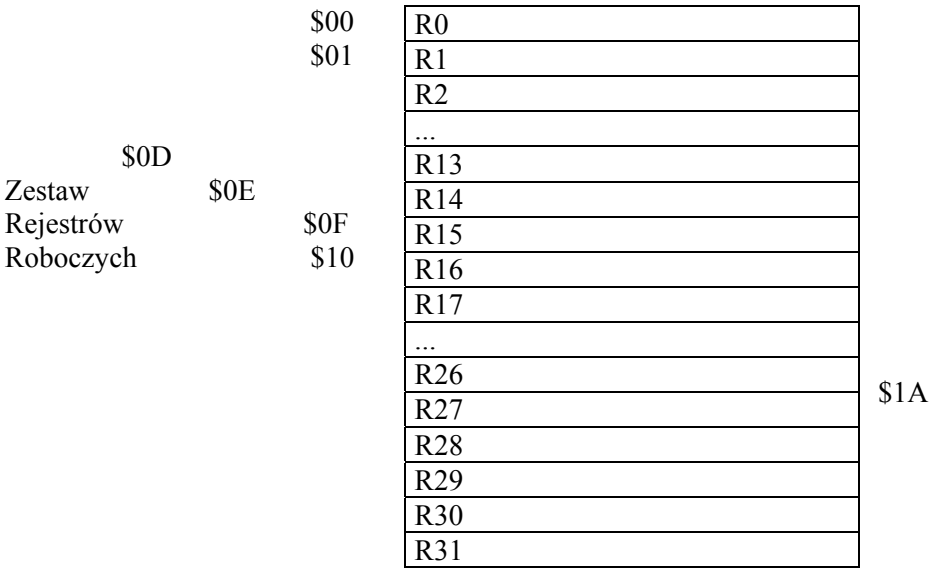
- **Ogólny zestaw rejestrów**

Zestaw rejestrów jest zoptymalizowany dla AVR zwiększonego zestawu rozkazów RISC. W celu osiągnięcia wymaganej szybkości i dopasowalności następujące I/O schematy są wspierane przez zestaw rejestrów:

a. jeden 8-bitowy wyjściowy argument i jeden 8-bitowy wejściowy wynik
b. dwa 8-bitowe wyjściowe argumenty i jeden 8-bitowy wejściowy wynik
c. dwa 8-bitowe wyjściowe argumenty i jeden 16-bitowy wejściowy wynik
d. jeden 16-bitowy wyjściowy argument i jeden 16-bitowy wejściowy rezultat.

Rysunek 4 pokazuje strukturę 32 roboczych rejestrów w jednostce centralnej.

7 0 Adres



X – rejestr młodszych bitów
X – rejestr starszych bitów
Y – rejestr młodszych bitów
Y – rejestr starszych bitów
Z – rejestr młodszych bitów
Z – rejestr starszych bitów

Większość rozkazów operujących na tych rejestrach ma bezpośredni dostęp do wszystkich rejestrów i większość to rozkazy wykonywane w jednym cyklu maszynowym.

Jak pokazano na 4 rysunku, każdy z rejestrów ma również adres w pamięci, który umieszcza zestaw rejestrów na pierwszych 32 bajtach przestrzeni danych. Chociaż nie są fizycznie zaimplementowane jako lokacje SRAM, taka organizacja pamięci zapewnia łatwy dostęp do rejestrów. X, Y i Z rejestry mogą być ustawione jako wskaźnik do innych rejestrów w zbiorze.

Rejestry R26..R31 mają parę dodatkowych funkcji. Rejestry te są 16-bitowymi wskaźnikami pośrednimi na adres przestrzeni danych. I tak w skład rejestru X wchodzą komórki R26 i R27, Y R28 i R29 , Z R30 i R31, gdzie pierwsza komórka określa młodszy bajt.

W różnych trybach adresacji rejestry te pełnią funkcje stałego przemieszczenia, automatycznej inkrementacji i dekrementacji (szczegóły dalej).

- **Wskaźnik stosu**

Stos jest głównie używany do przechowywania tymczasowych danych, lokalnych zmiennych i adresów powrotów z przerwań lub wywołań podprogramów. Rejestr wskaźnika stosu zawsze pokazuje na górę stosu. Stos jest zaimplementowany rosnąco w pamięci od wyższych numerów do niższych. Wpływa to na to, że instrukcja PUSH zmniejsza wartość wskaźnika stosu.

Wskaźnik stosu wskazuje na dane w przestrzeni stosu w SRAM, gdzie są ulokowane stosy Przerwań i Wywołań podprogramów. Rozmiar stosu musi być zdefiniowany zanim może dojść do wykonania podprogramów lub przerwań. Wskaźnik stosu musi zostać ustawiony tak aby wskazywał ponad adres $60. Wskaźnik stosu jest dekrementowany o jeden kiedy dane są odkładane na stos przez instrukcję PUSH i

jest dekrementowany o dwa kiedy odkładany jest adres powrotu z przerwania lub podprogramu. Wskaźnik stosu jest inkrementowany o jeden kiedy dane pobierane są przez instrukcję POP a o dwa przy powrotach z podprogramów - RET i przerwań – RETI.

Wskaźnik stosu w AVR jest zaimplementowany jako dwa 8-bitowe rejestry w przestrzeni I/O. Numer aktualnie używanych bitów jest zależny od implementacji. Należy zauważyć, że przestrzeń danych w niektórych implementacjach architektury AVR jest tak mała, że tylko SPL (młodszy bajt) jest potrzebne, w tym wypadku SPH (starszy bajt) nie będzie obecny.

Bit	15	14	13	12	11	10	9	8
SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
	7	6	5	4	3	2	1	0
InVal	0	0	0	0	0		0	0	0
	0	0	0	0	0		0	0	0

InVal – wartości początkowe

Wszystkie bity są do zapisu i odczytu.

- **RAMPZ – RAM page Z Select register**

Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	RAMPZ0
R/W	R	R	R	R	R	R	R	R/W
InVal	0	0	0	0	0	0	0	0

a: bity 7..2 – Res: Zarezerwowane bity

To są bity zarezerwowane i zawsze będą miały wartość zero. Podczas zapisu do tych lokacji należy zapisywać zero w celu kompatybilności z przyszłymi wersjami sprzętu.

b: bit 1 – TAMPZ0: rozszerzony stronicowy Z-wskaźnik RAM

Rejestr RAMPZ jest normalnie używany do wybrania, które 64k strony RAM jest dostępne przez Z-wskaźnik. Ponieważ ATmega128 nie wspiera więcej niż 64k pamięci SRAM ten rejestr jest używany tylko do wybrania, która strona w pamięci programu jest dostępna podczas korzystania z instrukcji ELPM/SPM. Różne ustawienia tego bitu mają następujące znaczenie:

RAMPZ0 = 0 przez ELPM/SPM jest dostępny adres pamięci programu od $0000 - $7FFF

RAMPZ0 = 1 przez ELPM/SPM dostępna jest pamięć o adresie $8000-$FFFF.

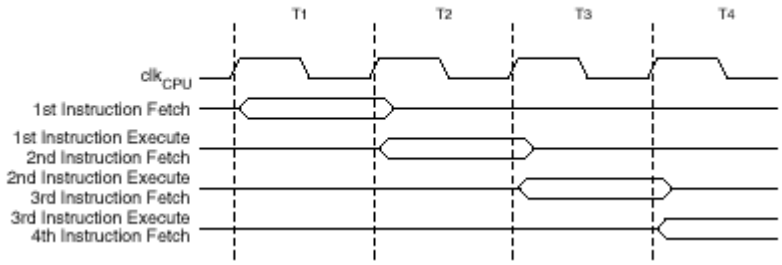
Ustawienie RAMPZ nie wpływa na LPM.

- **Przebiegi czasowe rozkazów**

Ta sekcja opisuje przebieg czasowy poszczególnych instrukcji. Jednostka centralna AVR jest napędzana przez zegar jednostki centralnej, bezpośrednio wygenerowany ze źródła zegara w układzie scalonym. Żaden wewnętrzny dzielnik zegara nie jest używany.

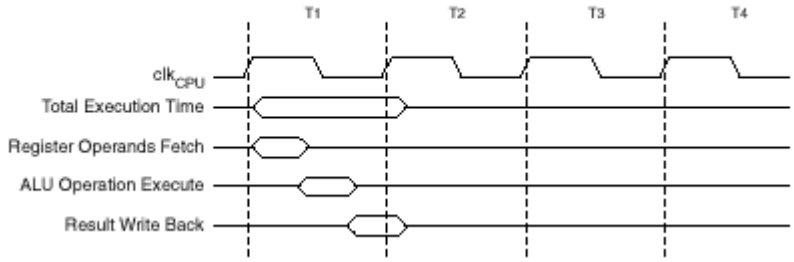
Rysunek 6 przedstawia równoległy rozkaz pobrania i wykonania. To jest podstawowy potokowy model osiągający do 1 MIPS na MHz.

Figure 6. The Parallel Instruction Fetches and Instruction Executions



Rysunek 7 przedstawia wewnętrzny model taktowania Zbioru rejestrów. W jednym cyklu maszynowym wykonywana jest operacja pobierająca operandy z dwóch rejestrów do ALU i jej wynik jest zapisywany w rejestrze przeznaczenia.

Figure 7. Single Cycle ALU Operation



- **RESET i obsługa przerwań**

AVR zapewnia wiele różnych źródeł przerwań. Przerwania i oddzielne wektory resetu mają oddzielny wektor programu w przestrzeni pamięci programu. Każde przerwanie ma przypisany bit przerwania, który musi być zapisany zgodnie logicznie z Globalnym bitem przerwań w rejestrze stanu w celu możliwości obsługiwania przerwań. Zależnie od wartości licznika programu (PC) przerwania mogą być automatycznie unieruchamiane kiedy bity blokujące program ładujący BLB02 lub BLB12 są zaprogramowane. To rozwiązanie ulepsza bezpieczeństwo oprogramowania. Szczegółowy opis w sekcji programowanie pamięci – 282.

Młodsze adresy w przestrzeni pamięci programu są domyślnie zdefiniowane jako wektory RESET'u i przerwań. Kompletna lista wektorów jest opisana w sekcji Przerwania – 54. Lista ta determinuje priorytet poszczególnych przerwań. Im niższy adres tym wyższy priorytet. RESET ma najwyższy priorytet, następnie INT0. Wektory przerwań mogą zostać przeniesione na początek ładowalnej sekcji pamięci Flash prze ustawienie bitu IVSEL w MCU rejestrze kontroli. Wektor RESET'u również może zostać przeniesiony do tej sekcji poprzez programowanie bezpiecznika BOOTRST. Dalsze informacje na stronie 269.

W przypadku wystąpienia przerwania, I-bit rejestru globalnych przerwań jest zerowany i wszystkie inne przerwania zostają zablokowane. Oprogramowanie użytkownika może wpisać do tego bitu logiczną jedynkę, aby odblokować zagnieżdżone przerwania. Wtedy wszystkie przerwania mogą przerywać procedurę obsługi przerwania. I-bit jest automatycznie ustawiany przy powrocie z przerwania - RETI.

Wyróżniamy dwa podstawowe rodzaje przerwań. Pierwszy typ jest uruchamiany poprzez zdarzenie, które ustawia flagę przerwań. Dla tych przerwań licznik programu (PC) jest ustawiany na aktualny wektor przerwania w celu wykonania procedury obsługi przerwania i sprzęt zeruje odpowiednią flagę przerwania. Flagi przerwań mogą być zerowane poprzez wpisywanie logicznej jedynki do bitu flagi. Jeśli zajdą wszystkie warunki przerwania podczas obsługi takiego samego przerwania zerującego bit przerwań, flaga zostanie ustawiona i zapamiętana do obsłużenia tego przerwania, lub jest ona wyzerowana przez oprogramowanie. Podobnie, jeżeli zostaną zgłoszone przerwania podczas, gdy bit globalnego przerwania jest wyzerowany, flagi tych przerwań zostaną zapamiętane do czasu ustawienia globalnego bitu przerwania i zostaną wykonane zgodnie z priorytetem.

Drugi typ przerwań będzie uruchamiany tak długo, jak warunki na zaistnienie przerwania będą obecne. Takie przerwanie nie muszą koniecznie mieć flag przerwań. Jeśli warunki powodujące przerwanie zakończą się zanim zostanie ono obsłużone, przerwanie to nie zostanie obsłużone.

Kiedy AVR wyjdzie z obsługi przerwania zawsze powróci do głównego programu i wykona następną instrukcję przed obsługą oczekującego przerwania.

Należy zauważyć, że rejestr stanu nie jest automatycznie odkładany na stos przy przejściu do obsługi przerwania, jak również nie jest z tego stosu pobierany po powrocie z obsługi przerwania. To musi zostać obsłużone programowo.

Korzystając z rozkazu CLI możemy natychmiastowo zamaskować przerwanie. Żadne przerwanie nie zostanie obsłużone po tym rozkazie, nawet jeżeli zostało zgłoszone równolegle z wykonywaniem tego rozkazu. Przykład pokazuje jak można wykorzystać ten rozkaż w celu uniknięcia przerwań przy zapisie do EEPROM.

Kod w asemblerze:

```
in  r16, SREG
cli
sbi EECR, EEMWE      //rozpoczęcie zapisu do EEPROM
sbi EECR, EEWE
out SREG, r16
```

Kod w C:

```
char cSREG;
cSREG  = SREG;
_CLI();
EECR |= (1<<EEMWE);  /* początek zapisu*/
EECR |= (1<<EEWE);
SREG = cSREG ;
```

Korzystając z rozkazu SEI można odmaskować zamaskowane przerwania. Instrukcja następująca po SEI będzie wykonana przed obsługą przerwania.

- **Czas odpowiedzi na przerwanie**

Minimalnym czasem oczekiwania na wykonanie przerwania jest czas czterech cykli maszynowych. Po czterech cyklach maszynowych adres programu wskazywany wektorem dla aktualnego obsługi przerwania jest wykonywany. W tym czasie licznik programu jest odkładany na stos. Zwyczajnie wektor jest skokiem do podprogramu obsługi przerwania, i skok ten trwa zazwyczaj trzy cykle maszynowe. Jeżeli dojdzie do zgłoszenia przerwania podczas wykonywania wielocyklowej operacji, operacja jest kończona

przed obsługą przerwania. Jeżeli zostanie zgłoszone przerwanie gdy MCU jest w trybie snu, odpowiedź na przerwanie jest wydłużona o cztery cykle maszynowe. To zwiększenie jest następstwem czasu potrzebnego na rozpoczęcie działania.

Powrót z procedury obsługi przerwania trwa cztery cykle maszynowe. W tym czasie ze stosu jest pobierany licznik programu, wskaźnik stosu jest zwiększany o dwa i jest ustawiany I-bit rejestru SREG.

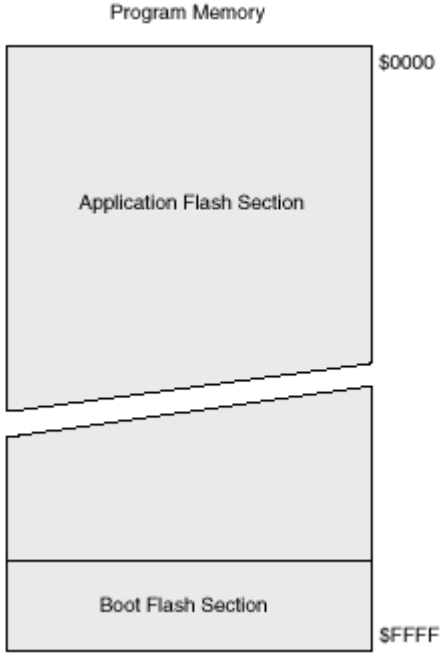
PAMIĘCI AVR ATMEGA128

Sekcja ta opisuje różne rodzaje pamięci w ATmega128. Architektura ATmega128 ma dwie główne przestrzenie pamięci: Pamięć Danych i Pamięć Programu. W dodatku, ATmega128 ma pamięć EEPROM służącą przechowywaniu danych. Wszystkie te przestrzenie pamięci są linearne i regularne.

- **Wewnętrzna reprogramowalna pamięć programu typu Flash.**

ATmega128 zawiera 128k bajtów reprogramowalnej pamięci Flash w strukturze półprzewodnikowej służącej do magazynowania pamięci. Odkąd wszystkie rozkazy są 16-to lub 32 bitowe, pamięć Flash jest zorganizowana jako 64K x 16. Dla bezpieczeństwa oprogramowania pamięć Flash jest podzielona na dwie sekcje: Inicjującą system i dla programów użytkowych.
Pamięć Flash ma trwałość ponad 1000 cykli zapisu i odczytu. Licznik programu w ATmega128 jest 16 bitowy, tak więc może zaadresować on 64k pamięci. Szczegóły dotyczące sekcji inicjującej system i związane z nim bity zabezpieczeń zostały opisane na stronie 269. Szczegółowy opis ładowania programu poprzez SPI przy użyciu JTAG interfejs'u znajduje się na stronie 282.
Stałe tablice mogą być alokowane przy użyciu całej przestrzeni adresowej pamięci programu.
Przebiegi czasowe dla instrukcji pobrania i wykonania znajdują się na stronie 13.

Figure 8. Program Memory Map



- **Pamięć danych SRAM**

ATmega128 wspiera dwa rodzaje konfiguracji pamięci SRAM:

Konfiguracja	Wewnętrzna pamięć SRAM	Zewnętrzna pamięć SRAM
tryb normalny	4096	Do 64k
Tryb kompatybilności z ATmega103	4000	Do 64k

ATmega128 jest złożonym mikrokontrolerem z większą ilością jednostek peryferyjnych niż może być obsłużona za pomocą 64 lokacji zarezerwowanych w kodzie operacyjnym dla rozkazów IN i OUT. Dla rozszerzonej przestrzeni pamięci I/O od $60 do $FF w SRAM, tylko rozkazy ST/STS/STD oraz LD/LDS/LDD mogą zostać użyte. Ta przestrzeń adresowa nie istnieje gdy jest włączony tryb kompatybilności z ATmega103.
W normalnym trybie pierwszych 4352 lokacji pamięci danych adresuje zarówno zestaw rejestrów jak i pamięć I/O, rozszerzoną pamięć I/O oraz wewnętrzną pamięć SRAM. Pierwsze 32 adresy odpowiadają zestawowi rejestrów, następne 64 adresy to pamięć I/O, potem 160 adresów to rozszerzona pamięć I/O i wreszcie 4096 adresów adresuje przestrzeń danych w SRAM.
W trybie kompatybilności z ATmega103, pierwsze 4096 adresów pamięci danych adresuje zarówno zestaw rejestrów jak i pamięć I/O, wewnętrzny SRAM. Pierwsze 32 adresy odnoszą się do zestawu rejestrów, następne 64 do pamięci I/O a pozostałe 4000 do SRAM.

Opcjonalna zewnętrzna pamięć danych SRAM może być dołączona do ATmega128. Ta pamięć będzie zajmowała pozostałą przestrzeń adresową – 64K. Przestrzeń ta rozpoczyna się po przestrzeni adresowej dla wewnętrznej pamięci SRAM. Zestaw rejestrów, I/O, rozszerzone I/O i wewnętrzna SRAM zajmują 4096 młodszych adresów w trybie kompatybilności z ATmega103, więc korzystając z 64KB zewnętrznej pamięci 61184 bajty zewnętrznej pamięci są dostępne w trybie normalnym a 61440 w trybie kompatybilności z ATmega103. Dalsze szczegóły w „Rozszerzonym interfejs'ie pamięci" na stronie 24.
Kiedy adresy dostępu do przestrzeni pamięci SRAM przekroczą adresy wewnętrznej pamięci, pamięć zewnętrzna jest dostępna dla tych samych rozkazów, dla których była dostępna pamięć wewnętrzna. Kiedy wewnętrzna pamięć jest dostępna, wyprowadzenia strobowe do odczytu i zapisu (PG0 – PG1) są nieaktywne podczas cyklu dostępu. Zewnętrzna pamięć SRAM jest dostępna poprzez ustawienie bity SRE w rejestrze MCUCR.
NA dostęp do zewnętrznej pamięci potrzebny jest dodatkowy cykl maszynowy porównujący bajt dostępu do wewnętrznej SRAM. TO znaczy, że rozkazy: LD, ST, LDS, STS, LDD, STD, PUSH i POP są dłuższe o jeden cykl maszynowy. Jeżeli stos jest umiejscowiony w zewnętrznej pamięci SRAM rozkazy powrotów i wejść do programów obsługi przerwań i podprogramów są o trzy cykle dłuższe ze względu na odkładanie i pobieranie dwubajtowej zawartości licznika rozkazów i adres zewnętrznej pamięci nie korzysta z wewnętrznej linii dostępu do pamięci. Kiedy zewnętrzny interfejs pamięci SRAM jest używany w trybie oczekiwania, jedno bajtowy zewnętrzny dostęp trwa jeden, trzy lub nawet o cztery dodatkowe cykle maszynowe dłużej odpowiednio dla jednego, trzech i czterech stanów oczekiwania. Zgłoszenia przerwań i odwołania do podprogramów i powroty z nich będą potrzebowały o pięć, siedem lub dziewięć cykli maszynowych więcej niż jest to przewidziane w zestawie rozkazów dla stanów oczekiwań.

Pięć trybów adresowania dla pamięci danych zawiera: bezpośredni, pośredni z przemieszczeniem, pośredni, pośredni z wcześniejszą dekrementacją i pośredni
z późniejszą inkrementacją. W zestawie rejestrów, rejestry R26 – R31 występują w roli pośrednich wskaźników do pamięci.

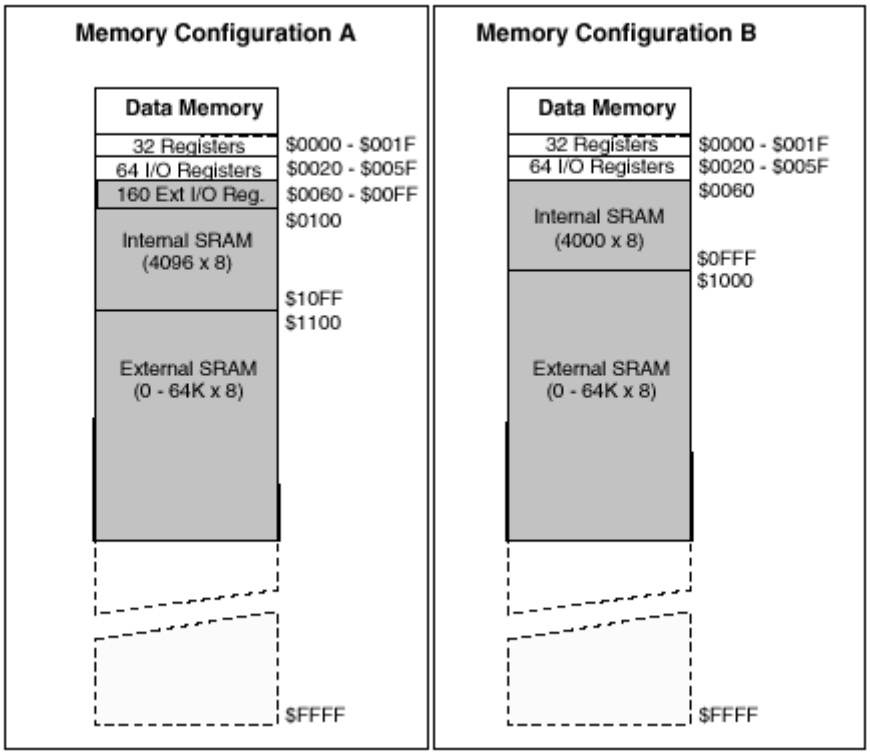
Adresowanie bezpośrednie osiąga cała przestrzeń adresową.

Adresowanie bezpośrednie z przemieszczeniem osiąga 63 adresy od bazy podanej przez rejestr Y lub Z.

Korzystając z pośredniego adresowania z wcześniejszą dekrementacją lub późniejszą inkrementacją, rejestry adresu X, Y, Z są dekrementowane lub inkrementowane.

32 ogólno dostępne rejestry robocze, 64 rejestrów I/O, i 4096 bajtów wewnętrznej przestrzeni danych SRAM w ATmega128 jest dostępne przez wszystkie tryby adresowania.

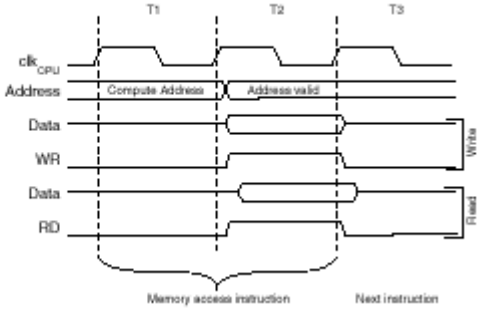
Figure 9. Data Memory Map



Czas dostępu do pamięci danych.

Sekcja ta opisuje ogólny pojęcie czasu dostępu do pamięci. Dostęp do wewnętrznej pamięci SRAM trwa dwa cykle maszynowe jak opisano na rysunku 10.

Figure 10. On-chip Data SRAM Access Cycles



EEPROM pamięć danych

ATmega128 zawiera 4k bajty pamięci danych EEPROM. Jest ona zorganizowana jako oddzielna przestrzeń pamięci, w której oddzielny bajt może być zapisywany lub odczytywany. EEPROM wytrzymuje ponad 100000 cykli zapisu i wymazania. Dostęp pomiędzy EEPROM i jednostką centralną jest opisany poprzez wyszczególnienie Rejestrów adresów EEPROM, Rejestrów danych EEPROM i rejestru kontroli EEPROM.

Szczegółowy opis ładowania pamięci poprzez SPI znajduje się na stronie 296 i 301.

EEPROM dostęp do zapisu i odczytu

Rejestry dostępu w EEP{ROM są dostępne w przestrzeni I/O.

Czas dostępu w przypadku zapisu jest podany w tabeli 2. Samowyzwalające się funkcje, jednakże pozwalają oprogramowaniu użytkownika wykrywać kiedy następny bajt może zostać zapisany. Jeśli program użytkowy zawiera instrukcje zapisując do EEPROM niektóre środki ostrożności muszą zostać zachowane. W silnie filtrowanych zasobach mocy, napięcie V może powoli opadać lub narastać podczas wyłączania lub załączania. To powoduje, że urządzenie pracuje przy niższym zasilaniu niż minimalne potrzebne do utrzymania odpowiedniej częstotliwości zegara. Szczegóły w „Zapobieganiu uszkodzeniom EEPROM" na stronie 23.

W celu uniknięcia niezamierzonych zapisów do EEPROM musi być przestrzegana specjalistyczna procedura zapisu. Szczegółowy opis w rejestrach kontroli EEPROM.

Podczas odczytu z EEPROM jednostka centralna jest zatrzymywana na cztery cykle maszynowe zanim następny rozkaz zostanie wykonany. Kiedy EEPROM jest zapisywany jednostka centralna jest zatrzymywana na dwa cykle maszynowe przed wykonaniem następnej instrukcji.

Rejestr Adresu EEPROM – EEARH, EEARL

-	-	-	-	EEAR11	EEAR10	EEAR9	EEAR8
EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0

R/W	R	R	RR	R/W	R/W	R/W	R/W	R/W
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
InVal	0	0	0 0	X	X X	X		
	X	X	X	X	X	X X	X	

R/W – odczyt/zapis
InVal – wartości początkowe

a: bity15..12 – Res: Bity zarezerwowane

Te bity są zarezerwowane i zawsze będą odczytywane jako 0. Podczas zapisu pod te adresy należy przypisać tym bitom zera dla kompatybilności z przyszłymi urządzeniami.

b: bity 11..0 – EEAR11..0: Adresy EEPROM

Rejestry adresu EEPROM – EEARL, EEARH – specyfikują adres w 4k przestrzeni adresów EEPROM. Bajty danych w EEPROM są ułożone linearnie pomiędzy 0 i 4096. Początkowa wartość EEAR jest niezdefiniowana. Jednak odpowiednia wartość musi zostać wpisana nim EEPROM będzie dostępny.

Rejestr danych EEPROM – EEDR

MSB							**LSB**

Wszystkie bity są zarówno do zapisu jak i odczytu. Wszystkie bity są inicjowane wartości logiczną 0.

a: bity 7..0 EEDR7..0 : dane EEPROM

Dla operacji zapisu w EEPROM, rejestr EEDR zawiera dane, które mają zostać zapisane pod adres podany w rejestrze EEAR. Przy operacji odczytu z EEPROM rejestr EEDR zawiera odczytaną daną z pod adresu podanego przez rejestr EEAR.

Rejestr kontroli EEPROM – EECR

	-	-	-	-	EERIE	EEMWE	EEWE	EERE
R/W	R	R	R	R	R/W	R/W	R/W	R/W
InVal	***0***	***0***	***0***	***0***	***0***	***0***	***X***	***0***

a: bity7..4 – Res: bity zarezerwowane

Te bity są bitami zarezerwowanymi i będą zawsze odczytywane jako zero.

b: bit 3 – EERIE: bit gotowości do obsłużenia przerwania

Wpisanie jeden umożliwia obsługę przerwania jeżeli I-bit w SREG jest ustawiony. Wpisanie zera maskuje przerwania. Bit gotowości przerwania EEPROM generuje stałe przerwanie iedy EEWE jest wyzerowane.

c: bit 2 – EEMWE: wzorcowy bit umożliwiający zapis

Bit ten determinuje czy ustawienie EEWE na jeden powoduje zapis do EEPROM. Jeżeli EEMWE jest ustawione na jeden to zapis jedynki do EEWE w przeciągu czterech cykli maszynowych spowoduje zapis danej do EEPROM pod wybranym adresem. Jeżeli EEMWE jest wyzerowane to wpisanie jedynki do EEWE nie będzie miało żadnego efektu. Jedynkując EEMWE przez oprogramowanie, sprzęt wyzeruje tą wartość po czterech cyklach maszynowych.

d: bit 1 – EEWE: Bit umożliwiający zapis do EEPROM

Kiedy adres i dane są dobirze ustawione bit ten musi być ustawiony aby zapisać wartość do EEPROM. EEMWE musi być ustawione kiedy logiczna jedynka jest wpisywana do EEWE, w przeciwnym wypadku nie nastąpi zapis do EEPROM. Następująca progredura powinna być przestrzegana podczas zapisu do EEPROM:

1. Czekać dopóki EEWE stanie się zerem
2. Czekać, aż SPMEN w SPMCR stanie się zerem
3. Zapisać nowy adres do EEAR – opcjonalne
4. Zapisać nową daną do EEDR – opcjonalne
5. Zapisać logiczną jedynkę do EEMWE podczas zapisu zera do EEWE i EECR
6. W czasie czterech cykli maszynowych po ustawieniu EEMWE zapisać logiczną jedynkę do EEWE.

EEPROM nie może być programowany podczas zapisu CPU do pamięci Flash. Oprogramowanie musi sprawdzić czy programowanie Flash jest skończone nim zacznie przygotowywać zapis do EEPROM. Krok 2 jest zależny od tego czy oprogramowanie zawiera boot loadera pozwalającego jednostce centralnej programować Flash. Jeżeli Flash nigdy nie jest aktualizowane przez CPU krok 2 może zostać pominięty.

UWAGA: Przerwanie między 5 i 6 krokiem spowoduję, że zapis będzie nieudany, ponieważ upłynie czas dla wzorcowego bitu umożliwiającego zapis. Jeśli obsługa przerwania mająca dostęp do EEPROM przerywa inny proces mający dostęp do EEPROM rejestry EEAR i EEDR zostaną zmodyfikowane powodując błąd przy dostępie do EEPROM. Zaleca się, aby globalną flagę przerwań zerować przy wykonywaniu czterech ostatnich kroków w celu uniknięcia tych problemów.

Kiedy czas na zapis upłynął bit EEWE jest zerowany przez sprzęt. Oprogramowanie użytkownika może sprawdzać ten bit w oczekiwaniu pojawienia się tam zera przed zapisem następnego bajtu. Kiedy EEWE jest ustawiony, jednostka centralna jest wstrzymywana na dwa cykle i nim następna instrukcja zostanie wykonana.

e: bit 0 – EERE: bit umożliwiający odczyt z EEPROM

Kiedy odpowiedni adres jest ustawiony w rejestrze EEAR bit EERE musi zostać ustawiony na logiczne jeden aby spowodować odczyt z EEPROM. Odczyt z EEPROM zajmuje jedną instrukcję i żądana dana jest dostępna od razu. Kiedy następuje odczyt z EEPROM jednostka centralna jest wstrzymywana na cztery cykle nim zostanie wykonana następna instrukcja.

Użytkownik powinien sprawdzać bit EERE przed rozpoczęciem operacji odczytu. Jeśli jest wykonywana instrukcja zapisu, to nie można ani odczytać EEPROM, ani zmienić danych w rejestrze EEAR.

Wzorcowy oscylator jest używany do regulacji dostępu do EEPROM. Tabela 2 przedstawia typowe czasy programowania dla dostępu do EEPROM z jednostki cendtralnej.

Symbol	Numer wzorcowego RC Cykle oscylatora	Typ programowanego czasu
Zapis EEPROM (z CPU)	8448	8.5 ms

Uwaga: Użycie zegara o częstotliwości 1MHz zależnego od ustawień CKSEL-bezpiecznika.

Następujące przykłady kodu pokazują funkcje zapisujące do EEPROM. Przykłady zakładają, że przerwania są kontrolowane tak, że nie zajdzie żadne przerwanie podczas wykonywania tych funkcji. Przykłady te zakładaj ą również, że nie ma flash boot loadera obecnego w oprogramowaniu. Jeśli taki kod jest obecny, funkcje zapisu muszą również oczekiwać na rozkaz zakończenia z SPM.

Kod w asemblerze:

```
EEPROM_write:
;czekaj, aż zostanie zakończony zapis poprzednich danych
sbic EECR, EERE
rjmp EEPROM_write
;ustaw adresy r18:r17 w rejestrze adresów
out EEARH ,r18
out EEARL, r17
; zapisz dane do rejestru r16
out EEDR, r16
; ustaw EEMWE na jeden
sbi EECR, EEMWE
;rozpocznij zapis
```

```
sbi EECR, EEWE
ret
```

Kod e C:

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
        while(EECR & (1<<EEWE      ))
        EEAR = uiAddress;
        EEDR  =ucData;
        EECR |= (1<<EEMWE);
        EECR |= (1<<EEWE);
}
```

Następne przykłady kodu przedstawiają funkcje odczytujące z EEPROM. Przykłady te zakładają, że przerwania są kontrolowane i że nie zajdzie żadne podczas obsługi tych funkcji..

Kod w Asemblerze:

```
EEPROM_read;
// czekaj na zakończenie wcześniejszego zappisu
sbic EECR, EEWE
rjmp EEPROM_read
// ustaw  rejestr adresów
out EEARH, r18
out EEARL, r17
//rozpocznij odczyt
sbi EECR, EERE
// odczytaj dane z rejestru danych
in r16, EEDR
ret
```

Kod w C:

```
unsigned char EEPROM_read(unsigned char uiAddress)
{
        while(EECR & (1<<EEWE))
                ;
        EEAR= uiAddress;
        EECR |= (1<<EERE);
        return EEDR;
}
```

Zapobieganie zniekształceniom EEPROM

W okresach kiedy jest niski stan napięcia dane w EEPROM mogą ulec zniekształceniu ponieważ zasób napięci jest zbyt niski by CPU i EEPROM mogły działać poprawnie. Rezultaty są takie same jak dla tablicowego poziomu systemów korzystających z EEPROM i takie same rodzaje rozwiązań powinny zostać zastosowane.

Zniekształcenia danych w EEPROM mogą zajść w dwóch przypadkach gdzy napięcie jest za niskie. Po pierwsze, normalna sekwencja zapisu do EEPROM wymaga minimalnego napięcia aby przebiegać

poprawnie. Po drugie, jednostka centralna sama w sobie może wykonywać rozkazy niepoprawnie przy zbyt niskim napięciu.

Błędów w danych zapisanych na EEPROM można łatwo uniknąć postępując w następujący sposób:

Należy utrzymywać AVR RESET w stanie aktywnym (stan niski) w okresach niewystarczającego zasobu mocy. To może zostać zapewnione przez odblokowanie wewnętrznego detektora zużycia energii elektrycznej (BOD). Jeżeli poziom wykrycia nie odpowiada wymaganemu poziomowi zewnętrzny obwód ochrony przed resetem może zostać wykorzystany. Jeżeli podczas operacji zapisu procesor zostanie zresetowany to operacja zapisu zostanie dokończona po zapewnieniu odpowiedniego zasobu napięcia.

Pamięć I/O

Definicja przestrzeni pamięci I/O dla ATmega128 jest pokazana w Streszczeniu rejestrów na stronie 341.

Wszystkie urządzenia zewnętrzne i peryferyjne ATmega128 są umiejscowione w przestrzenie pamięci I/O. Wszystkie te lokacje są osiągalne poprzez rozkazy LD/LDD/LDS i ST/STS/STD, które przekazują dane pomiędzy 32 ogólnymi roboczymi rejestrami i przestrzenią pamięci I/O. Rejestry I/O o adresach rzędu $00 - $1F są bezpośrednio bitowo dostępne korzystając z rozkazów SBI i CBI. W tych rejestrach wartość poszczególnych bitów może zostać sprawdzona poprzez rozkazy SBIS i SBIC. Szczegóły w sekcji o rozkazach. Korzystając ze specyficznych rozkazów I/O – IN, OUT muszą zostać wykorzystane adresy $00 - $3F. Adresując rejestry I/O jako przestrzeń danych korzystając z rozkazów LD i ST $20 musi zostać dodane do pierwotnego adresu. ATmega128 jest złożonym mikrokontrolerem z większą ilością jednostek peryferyjnych niż może zostać obsłużonych przez 64 lokacje zarezerwowane w kodzie operacyjnym dla rozkazów IN i OUT. Dla rozszerzonej pamięci I/O od $60 - $FF w SRAM tylko rozkazy ST/STS/STD i LD/LDS/LDD mogą zostać użyte. Rozszerzona przestrzeń pamięci I/O jest wymieniana z SRAM kiedy ATmega128 jest w trybie kompatybilności z ATmega103.

W celu zapewnienia współpracy z przyszłymi urządzeniami, bity zarezerwowane powinny być ustawione na zero jeśli są dostępne. Zarezerwowane adresy pamięci I/O nigdy nie powinny być zapisywane.

Niektóre flagi statusu są zerowane przed zapisywaniem logicznych jedynek do nich. Należy zauważyć, że rozkazy CBI i SBI będą operowały na wszystkich bitach rejestru I/O zapisując jedynkę z powrotem do każdej odczytanej flagi, tak więc zerując te flagi. CBI i SBI pracują tylko na rejestrach $00 - $1F.

Rejestry kontrolne dla I/O i urządzeń peryferyjnych są opisane w późniejszych sekcjach.

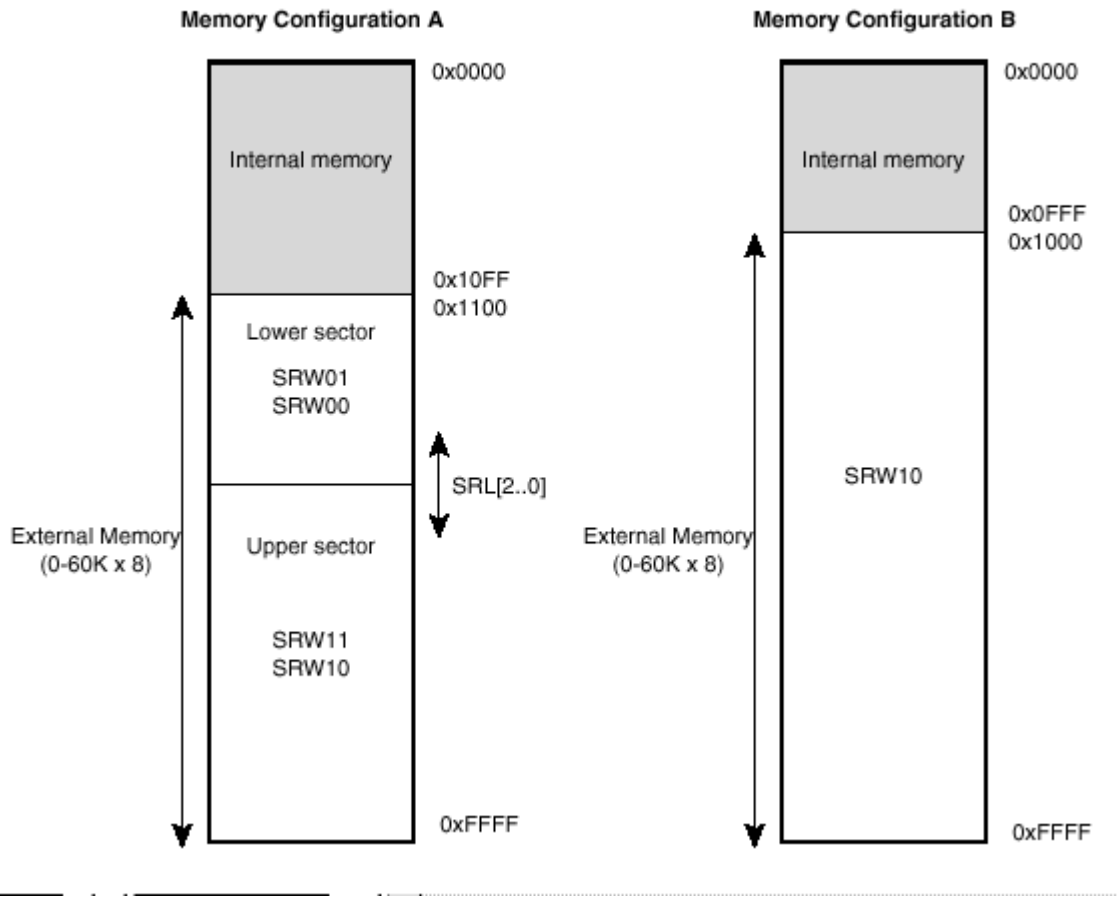
- **Interfejs zewnętrznej pamięci.**

Z wszystkimi swoimi cechami Zewnętrzny interfejs pamięci zapewnia dobre dopasowanie do operowania na nim jako na interfejsie do pamięci takich jak zewnętrzny SRAM i Flash oraz do urządzeń peryferyjnych takich jak wyświetlacze LCD, A/D i D/A. Główne cechy to:

  - Cztery różne ustawienia stanów oczekiwania
  - Uzależnienie ustawień stanów oczekiwania od różnych rodzai zewnętrznych sektorów pamięci (konfigurowalny rozmiar sektora)
  - Liczba bitw przeznaczona na zaadresowanie wyższych bajtów jest obieralna.
  - Strażnik magistrali na liniach danych minimalizujący pobór prądu. (opcjonalne)

Kiedy pamięć zewnętrzna jest dostępna (XMEM) przestrzeń adresowa na zewnątrz wewnętrznego SRAM staje się dostępna poprzez korzystania z przeznaczonych do pamięci zewnętrznej wyjść. Konfiguracja pamięci przedstawiona jest na rysunku 11.

Figure 11. External Memory with Sector Select



Uwaga: ATmega128 nie pracująca w trybie kompatybilności z ATmega103 Dostępna konfiguracja pamięci A (Konfiguracja pamięci B N/A)
ATmega128 pracująca w trybie kompatybolności z ATmega103: Dostępna konfiguracja pamięci B (Konfiguracja pamięci A N/A)

- **Kompatybilność z ATmega103**

Obydwa rejestry kontroli pamięci zewnętrznej (XMCRA i XMCRB) są umieszczone w przestrzeni rozszerzonej I/O. W trybie kompatybilności z ATmega103 rejestry te nie są dostępne jak i właściwości przez nie specyfikowane. Ograniczenia w kompatybilności z ATmega103 to:

  - Tylko dwa stany oczekiwania są dostępne (SRW1n = 0b00 i SRW1n = 0b01);
  - Liczba bitw przypisanych wyższym bajtom adresowym jest stała.
  - Pamięć zewnętrzna nie może być podzielona na sekcje z różnymi ustawieniami stanów oczekiwania.
  - Strażnik magistrali nie jest dostępny
  - Wyjścia RD, WR i ALE są tylko wyjściowe (port G w ATmega128)

- **Korzystanie z interfejsu zewnętrznej pamięci**

Interfejs składa się z:

- AD7..0: multipleksowana magistrala adresowa i magistrala danych aktywna niskim poziomem
- A15..8: aktywna wysokim poziomem magistrala adresowa (konfigurowalna ilość bitów)
- ALE: Zatrzask adresu
- RD: strob odczytu
- WR: strob zapisu

Bity kontrolne dla interfejsu zewnętrznej pamięci są ulokowane w trzech rejestrach: rejestr kontroli MCU (MCUCR), rejestr kontroli pamięci zewnętrznej A (XMCRA) i rejestr kontroli zewnętrznej pamięci B – (XMCRB).

Kiedy XMEM interfejs jest odblokowany przepisze ustawienia rejestru namiaru danych, który odpowiada portom przeznaczonym na interfejs XMEM. Szczegóły w sekcji „ Porty I/O" na stronie 60. Interfejs XMEM wykrywa czy jest dostęp zewnętrzny czy wewnętrzny. Jeżeli dostęp jest z zewnątrz interfejs XMEM będzie wystawiał adres, dane i sygnały kontrolne na portach tak jak jest to pokazane na rysunku 13 (rysunek nie bierze pod uwagę stanów oczekiwania) . Kiedy sygnał ALE przejdzie z jedynki logicznej do zera oznacza to wystawienie dobrego adresu na liniach AD7..0. ALE jest w stanie niskim podczas transferu danych. Kiedy interfejs XMEM jest odblokowany również wewnętrzny dostęp do pamięci spowoduje aktywność na liniach adresów, danych i portach ALE, jednak stroby WR i RD nie będą aktywne. Kiedy interfejs zewnętrznej pamięci jest zablokowany normalne wyjścia i ustawienia namiaru danych są używane. Należy zauważyć, że kiedy interfejs XMEM jest zablokowany przestrzeń adresowa ponad granicą przestrzenią w pamięci wewnętrznej SRAM nie jest mapowany w tej pamięci. Rysunek 12 ilustruje jak podłączyć zewnętrzny SRAM do AVR korzystając z ósemkowego zatrzasku (zazwyczaj „74x573" lub równoważnego), który jest niewidoczny dla G w stanie wysokim.

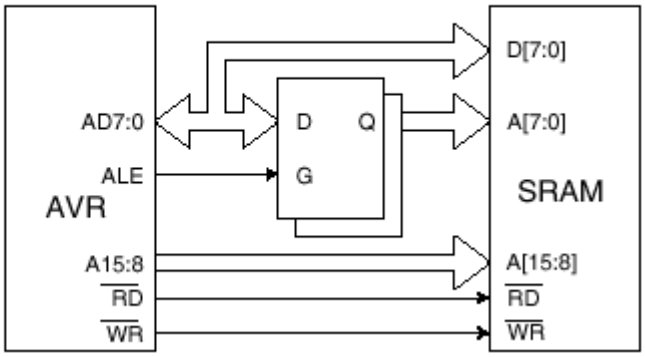
- **Wymagania zatrzasku adresu**

Ze względu na dużą szybkość operacji na interfejsie XRAM zatrzask adresu musi zostać dobrany z ostrożnością dla systemów o częstotliwościach większych niż 8MHz i 4V lub 4MHz i 2.7V. Podczas operowania w przy większych częstotliwościach typowy zatrzask 74HC staje się nieodpowiedni. Interfejs pamięci zewnętrznej został zaprojektowany zgodnie z tą serią zatrzasku. Chociaż większość zatrzasków może być używana jak długo przestrzegają parametrów głównego taktowania. Parametry te dla zatrzasków adresowych to:

- Od D do Q opóźnienie czasu propagacji (t_{PD})
- Czas ustalania danych przed G w stanie niskim (t_{SU})
- Czas podtrzymywania adresu po przejściu G do stanu niskiego ($_{TH}$)

Interfejs zewnętrznej pamięci został zaprojektowany w celu zagwarantowania minimalnego czasu utrzymywania adresu po zapewnieniu G niskiego stanu t_H = 5ns. Odnosząc się do t_{LAXX_LD}/t_{LLAXX_ST} w „Taktowaniu zewnętrznej pamięci danych" tabela 137 do 144 na stronach 321 323. Opóźnienie czasu propagacji od D do Q musi być brane pod uwagę podczas obliczania wymaganego czasu dostępu poprzez zewnętrzny komponent. Czas ustalania się danych nim G przejdzie w stan niski nie może przekraczać adresu obowiązującego w niskim stanie ALE pomniejszonym o opóźnienie zapisu PCB (zależne od pojemnościowego obciążenia).

Figure 12. External SRAM Connected to the AVR



- **Rezystory podciągające i strażnik magistrali**

Rezystory podciągające i porty AD7..0 mogą być uaktywnione jeżeli w odpowiadającym im rejestrze portu jest zapisane jeden. Aby zmniejszyć pobór mocy w trybie snu jest rekomendowany aby odciąć rezystory podciągające poprzez wpisanie do rejestru portów zera przed przejściem w tryb snu.

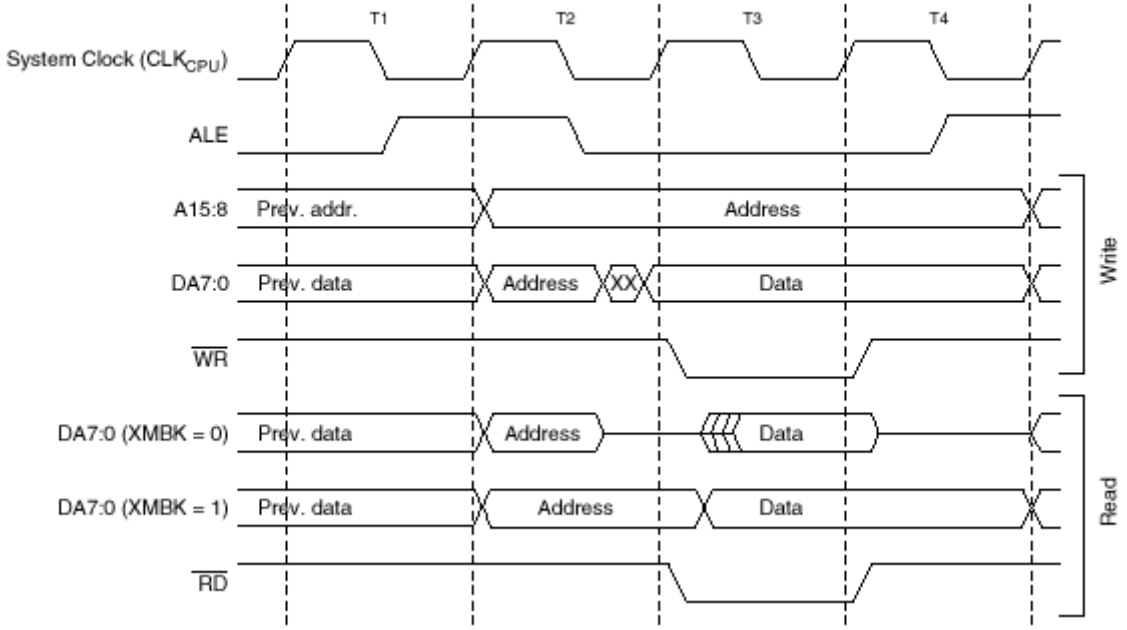
Interfejs XMEM zapewnia również strażnika magistrali na liniach AD7..0. Kontroler magistrali może być odcinany i blokowany poprzez oprogramowanie co jest opisane w sekcji „Rejestr kontrolny zewnętrznej pamięci B XMCRB" na stronie 31. Kiedy linie są odblokowane strażnik magistrali będzie utrzymywał poprzednią wartość na liniach AD7..0 magistrali podczas, gdy interfejs XMEM ustawi je w trzeci stan.

W przypadku gdy, ani kontroler magistrali, ani rezystor podciągające nie są odblokowane interfejs XMEM pozostawi linie AD7..0 w trzecim stanie podczas dostępu do odczytu dopóki nie pojawi się następny dostęp do RAM.

Pamięci zewnętrzne mają różne wymagania taktowania. Aby sprostać tym wymaganiom interfejs XMEM ATmega128 zapewnia cztery różne stany oczekiwania jak pokazuje tabela 4. Przed wyborem stany oczekiwania należy zapoznać się ze specyfikacją zewnętrznej pamięci. Najważniejszymi parametrami są czas dostępu do zewnętrznej pamięci porównywany z wymaganiami załączania ATmega128. Czas dostępu do pamięci zewnętrznej jest zdefiniowany jako czas, który upłynął od otrzymania adresu do wyprowadzenia danych przez niego opisywanych na magistralę. Czas dostępu nie może przekroczyć czasu impulsu ALE, który musi być w stanie niskim by zapewnić stałość danych przy odczycie. (przejrzyj $t_{LLRL} + t_{RLRH} - t_{DVRH}$ w tabelach 137..144 na stronach 321 – 323). Różne czasy oczekiwania są ustawione w oprogramowaniu. Dodatkową cechą jest możliwość podziału pamięci zewnętrznej na dwa sektory z indywidualnymi ustawieniami czasów oczekiwania. Takie rozwiązanie umożliwia podłączenie dwóch różnych pamięci z różnymi wymaganiami czasowymi do jednego interfejsu XMEM. Szczegóły w tablicach 137 –144 oraz na rysunkach 156 – 159 w rozdziale „Taktowanie zewnętrznej pamięci danych" na stronie 321.

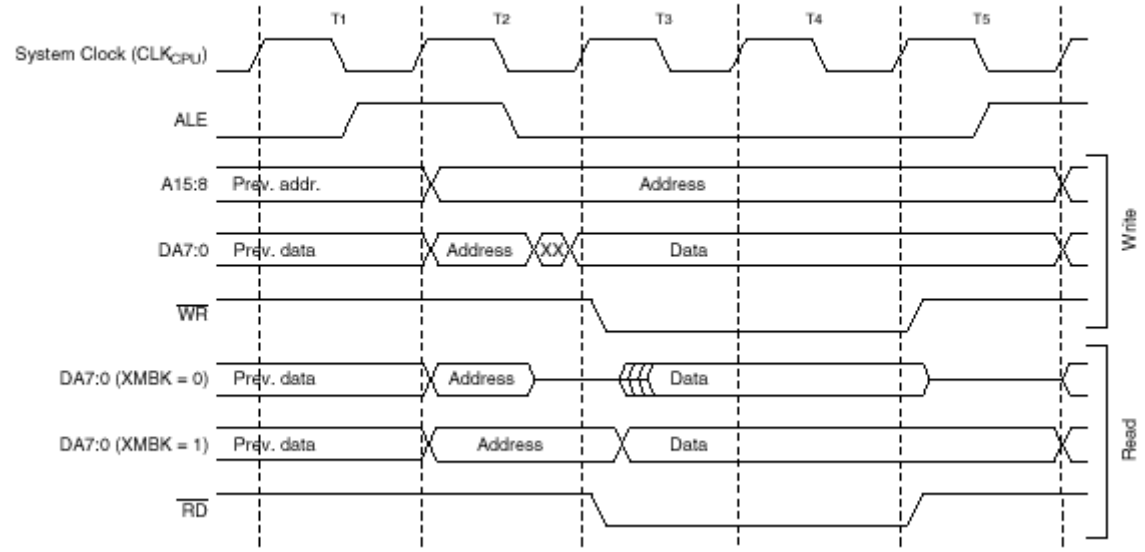
Należy zauważyć, że interfejs XMEM jest asynchroniczny i kształt fali na następujących rysunkach jest odniesiony do wewnętrznego systemu zegarowego. Skos pomiędzy zewnętrznym zegarem XTAL1 nie jest gwarantowana (zmienia się wraz z temperaturą urządzenia i doprowadzonym napięciem. Zgodnie z tym, interfejs XMEM jest nie dopasowany do synchronicznych operacji.

Figure 13. External Data Memory Cycles without Wait-state (SRWn1=0 and SRWn0=0)



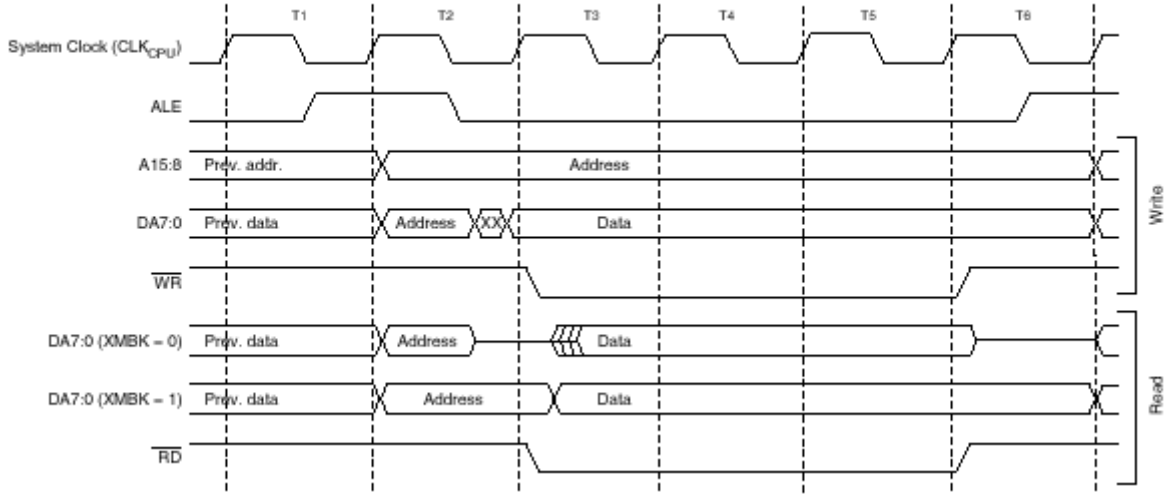
Uwaga: SRWn1 = SRW11 (wyższy sektor) lub SRW01 (niższy sektor), SRWn0 = SRW10 (wyższy sektor) lub SRW00 (niższy sektor. Impuls ALE w T4 jest obecny tylko jeżeli następna instrukcja żąda dostępu do pamięci.

Figure 14. External Data Memory Cycles with SRWn1 = 0 and SRWn0 = 1(1)



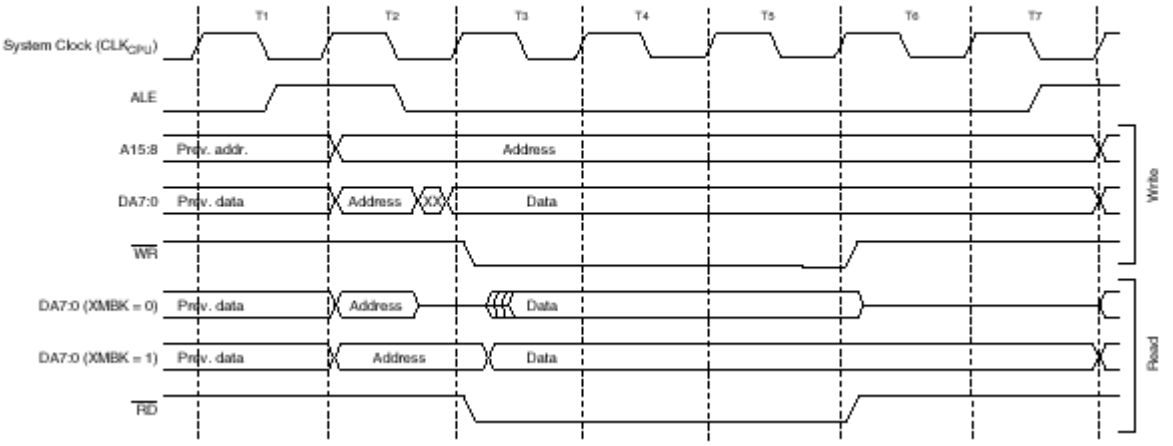
Uwaga: SRWn1 = SRW11 (wyższy sektor) lub SRW01 (niższy sektor), SRWn0 – SRW10 (wyższy sektor) lub SRW00 (niższy sektor). Impuls ALE w T5 jest obecny tylko jeżeli następna instrukcja żąda dostępu do pamięci.

Figure 15. External Data Memory Cycles with SRWn1 = 1 and SRWn0 = 0[1]



Uwaga: SRWn1 = SRW11 (wyższy sektor) lub SRW01 (niższy sektor), SRWn0 = SRW10 (wyższy sektor) lub SRW00 (niższy sektor) Impuls ALE w T6 jest obecny tylko jeżeli następna instrukcja żąda dostępu do pamięci.

Figure 16. External Data Memory Cycles with SRWn1 = 1 and SRWn0 = 1[1]



Uwaga: SRWn1 = SRW11 (wyższy sektor) lub SRW01 (niższy sektor), SRWn0 – SRW10 (wyższy sektor) lub SRW00 (niższy sektor). Impuls ALE w T7 jest obecny tylko jeżeli następna instrukcja żąda dostępu do pamięci.

- **Opis rejestru XMEM**
  - **Rejestr kontroli MCU – MCUCR**

Bit	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUCR
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	

a: bit 7 – SRE: Odblokowanie zewnętrzne SRAM/XMEM

Zapisując jeden do tego bitu odblokowuje się interfejs zewnętrznej pamięci. Funkcje wyjść AD7.0, A15..8, ALE, WR i RD są aktywne jako alternatywne funkcje wyjść. Bit SRE przeciąża ustawienia kierunku każdego wyjścia w odpowiednim rejestrze kierunku danych. Ustawiając SRE na zero zablokowuje się ten interfejs i normalne wyjścia i ustawienia kierunków danych są używane.

b: bit 6 – SRW10: bit wyboru trybu oczekiwania

Dokładny opis w trybie bez współpracy z ATmega103 przejrzyj zwyczajny opis poniżej dla bitów SRWn (XMCRA). Natomiast w trybie współpracy z ATmega103 zapisanie jedynki SRW10 dodaje jeden dodatkowy takt podczas strobu zapisu lub odczytu w stanie oczekiwania. Rysunek 14.

- **Rejestr kontroli pamięci A – XMCRA**

Bit	7	6	5	4	3	2	1	0	
	–	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11	–	XMCRA
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

a: bit 7 – Res: bit zerezerwowany

Ten bit jest zarezerwowany i zawsze będzie odczytywany jako zero. Podczas zapisu do tego adresu należy wpisać zero w celu zapewnienia kompatybilności z przyszłymi urządzeniami.

b: bity 6..4 – SRL2, SRL1, SRL0: sektor limitu czasu oczekiwania

Dla różnych adresów zewnętrznych pamięci można skonfigurować różne stany oczekiwania. Adresy zewnętrznej pamięci mogą być podzielone na dwa sektory z oddzielnymi bitami stanów oczekiwania. SRL2, SRL1, SRL0 wybierają miejsce podziału na sektory, zobacz tabelę 3 i rysunek 11. Domyślnie bity SRL2, SRL1, SRL0 są ustawione na zero i cała zewnętrzna przestrzeń adresowa pamięci jest traktowana jako jeden sektor. Kiedy cała przestrzeń adresowa SRAM jest skonfigurowana jako jeden sektor bity SRW11 i SRW10 określają stan oczekiwania.

Table 3. Sector limits with different settings of SRL2..0

SRL2	SRL1	SRL0	Sector Limits
0	0	0	Lower sector = N/A Upper sector = 0x1100 - 0xFFFF
0	0	1	Lower sector = 0x1100 - 0x1FFF Upper sector = 0x2000 - 0xFFFF
0	1	0	Lower sector = 0x1100 - 0x3FFF Upper sector = 0x4000 - 0xFFFF
0	1	1	Lower sector = 0x1100 - 0x5FFF Upper sector = 0x6000 - 0xFFFF
1	0	0	Lower sector = 0x1100 - 0x7FFF Upper sector = 0x8000 - 0xFFFF
1	0	1	Lower sector = 0x1100 - 0x9FFF Upper sector = 0xA000 - 0xFFFF
1	1	0	Lower sector = 0x1100 - 0xBFFF Upper sector = 0xC000 - 0xFFFF
1	1	1	Lower sector = 0x1100 - 0xDFFF Upper sector = 0xE000 - 0xFFFF

c: bit 1 i bit 6 MCUCR – SRW11, SRW10: bity wyboru stanu oczekiwania dla wyższego sektora

SRW11 I SRW10 są bitami kontroli liczby stanów oczekiwania dla wyższego sektora przestrzeni adresowej pamięci zewnętrznej. Patrz tabela 4.

d: bity 3..2 – SRW01, SRW00: bity wyboru stanu oczekiwania dla sektora niższego

Bity te kontrolują liczbę stanów oczekiwania dla niższego sektora przestrzeni adresowej pamięci zewnętrznej. Patrz tabela 4.

Tabela 4

SRWn1	SRWn0	Stany oczekiwania
0	0	Nie ma stanu oczekiwania
0	1	Czekaj jeden cykl maszynowy przy strobie zapisu/odczytu
1	0	Czekaj dwa cykle maszynowy przy strobie zapisu/odczytu
1	1	Czekaj dwa cykle maszynowy przy strobie zapisu/odczytu i czekaj jeden cykl przed wyprowadzeniem nowego adresu

Uwaga: n=0 lub 1 (wyższy/niższy sektor)
Dalsze informacje o taktowaniu stanów oczekiwań dla interfejsu pamięci zewnętrznej obejrzyj rysunki 13-16.

e: bit 0 –Res: bit zarezerwowany

Ten bit jest zarezerwowany i zawsze będzie odczytywany jako zero. Podczas zapisu do tego adresu należy wpisać zero w celu zapewnienia kompatybilności z przyszłymi urządzeniami.

- **Rejestr kontroli pamięci zewnętrznej BXCRB**

Bit	7	6	5	4	3	2	1	0	
	XMBK	–	–	–	–	XMM2	XMM1	XMM0	XMCRB
Read/Write	R/W	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

a: bit 7 – XMBK: bit umożliwiający kontrolę magistrali zewnętrznej pamięci

Wpisanie jedynki do tego bitu odblokowuje kontrolera magistrali na liniach AD7..0. Kiedy kontroler ten jest odblokowany ,linie danych będą przetrzymywać ostatnią wpisaną wartość nawet jeżeli interfejs XMEM ustawił linie w trzecim stanie. Wpisując XMBK zero odcinamy kontrolera magistrali XMBK nie zależy od SRE, więc gdy interfejs XMEM jest odcięty kontrolerzy magistrali są aktywni dopóki XMBK jest jedynką.

b: bity 6..4 – Res: bit zarezerwowany

Ten bity są zarezerwowane i zawsze będą odczytywane jako zero. Podczas zapisu do tego adresu należy wpisać zero w celu zapewnienia kompatybilności z przyszłymi urządzeniami

c: bity 2..0 – XMM2, XMM!, XMM0: Maska pamięci zewnętrznej

Kiedy jest możliwy dostęp do zewnętrznej pamięci wszystkie wyjścia portu C są domyślnie używane dla starszych bajtów adresowych. Jeżeli pełnie 60K przestrzeni adresowej nie jest potrzebne dla dostępu do pamięci zewnętrznej, niektóre albo i wszystkie wyjścia portu C mogą być zwolnione i działać zgodnie z pierwotnym przeznaczeniem – patrz tabela 5. Jak opisano w „Wykorzystywanie wszystkich 64K lokacji zewnętrznej pamięci" na stronie 32, jest możliwe by bity XMMn były używane jako bity dostępu do 64KB pamięci zewnętrznej.

Table 5. Port C Pins Released as Normal Port Pins when the External Memory is Enabled

XMM2	XMM1	XMM0	# Bits for External Memory Address	Released Port Pins
0	0	0	8 (Full 60 KB space)	None
0	0	1	7	PC7
0	1	0	6	PC7 - PC6
0	1	1	5	PC7 - PC5
1	0	0	4	PC7 - PC4
1	0	1	3	PC7 - PC3
1	1	0	2	PC7 - PC2
1	1	1	No Address high bits	Full Port C

- **Wykorzystanie wszystkich adresów 64kB pamięci zewnętrznej.**

Odkąd pamięć zewnętrzna jest mapowana za pamięcią wewnętrzną jak pokazana na rysunku 11 tylko 60 KB tej pamięci jest dostępnych domyślnie (przestrzeń adresowa 0x0000 – 0x10FF jest zarezerwowana dla

pamięci wewnętrznych). Chociaż, jest możliwe korzystanie z całej pamięci zewnętrznej poprzez maskowanie starszych bitów adresów przez zero. Można w tym celu wykorzystać bity XMMn i kontrolować poprzez oprogramowanie znaczące bity adresu. Ustawiając port C jako wyjściowy 0x00 i umożliwiając normalną pracę ważniejszych bitów portu C interfejs pamięci będzie adresowany następująco 0x0000 – 0x1FFF. Przejrzyj przykłady kodów:

Kod w asemblerze:

```
//OFSET jest zdefiniowany jako 0x2000 by zapewnić dostęp do zewnętrznej pamięci
//starsze bity portu C są skonfigurowane jako wyjściowe dla tych  wyjść pracujących normalnie.
ldi   r16, 0xFF
out   DDRC, r16
ldi   r16, 0x00
out   PORTC, r16
; release PC7:5
ldi   r16, (1<<XMM1)|(1<<XMM0)
sts   XMCRB, r16
; write 0xAA to address 0x0001 of external
; memory
ldi   r16, 0xaa
sts   0x0001+OFFSET, r16
; re-enable PC7:5 for external memory
ldi   r16, (0<<XMM1)|(0<<XMM0)
sts   XMCRB, r16
; store 0x55 to address (OFFSET + 1) of
; external memory
ldi   r16, 0x55
sts   0x0001+OFFSET, r16
```

Przykład kodu w C:

```
#define OFFSET 0x2000

void XRAM_example(void)
{
unsigned char *p = (unsigned char *) (OFFSET + 1);

DDRC = 0xFF;
PORTC = 0x00;

XMCRB = (1<<XMM1) | (1<<XMM0);

*p = 0xaa;

XMCRB = 0x00;

*p = 0x55;
}
```

Uwaga: Przykłady kodu zakładają, że odpowiednie pliku nagłówkowe są dołączone.

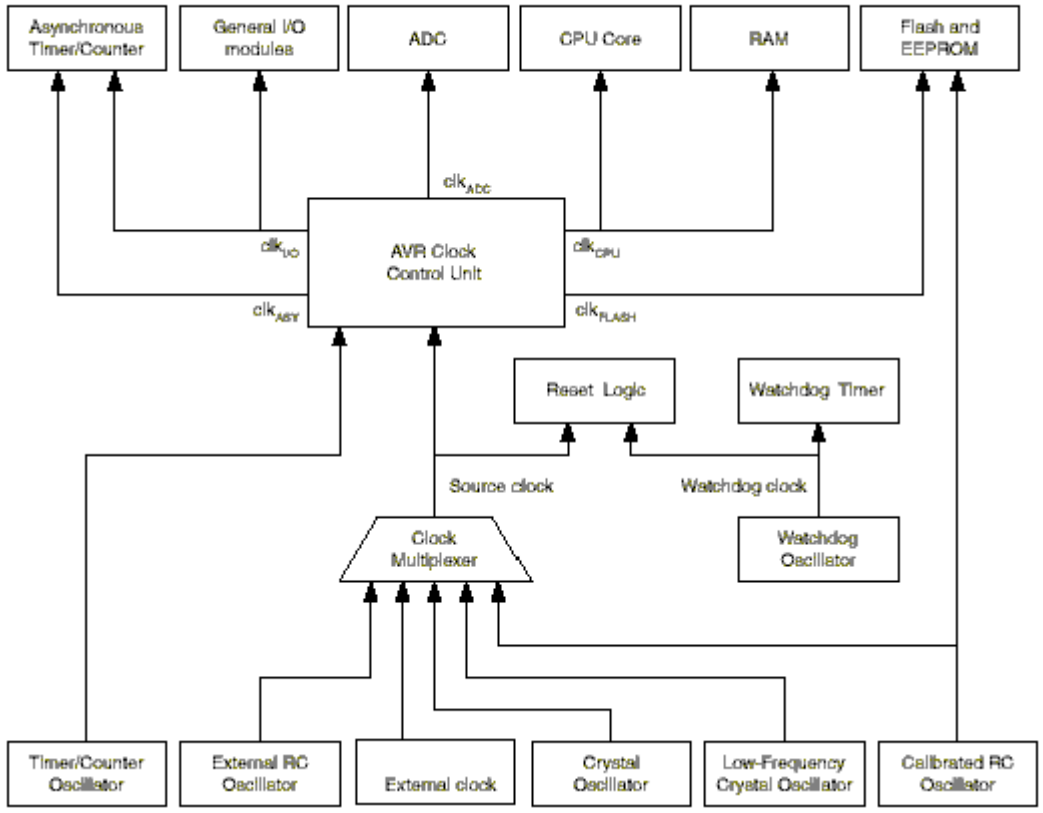
Należy korzystać z tych możliwości z ostrożnie gdyż większość pamięci jest zamaskowana.

ZEGAR SYSTEMOWY I JEGO OPCJE

- **Systemy zegarowe i ich dystrybucja**

Rysunek 17 przedstawia główne systemy zegarowe w AVR i ich rozdział. Wszystkie te zegary nie muszą działać razem. W celu zmniejszenia poboru mocy zegary do poszczególnych nieużywanych modułów mogą zostać zatrzymane przy wykorzystaniu różnych trybów snu, co zostało opisane w „Zarządzanie mocą i tryby snu" na stronie 31. Systemy zegarowe są opisane szczegółowo poniżej.

Figure 17. Clock Distribution



- **Zegar CPU**

Zegar CPU taktuje operacje systemu na rdzeniu AVR. Przykładami takich modułów jest zestaw rejestrów ogólnego dostępu, rejestr statusu oraz pamięć danych przetrzymująca wskaźnik stosu. Zatrzymując ten zegar zatrzymujemy główną jednostkę, tak, że nie mogą być wykonywane główne operacje i obliczenia.

- **Zegat I/O**

Zegar I/O jest używany w większości modułów I/O jak stopery – liczniki, SPI i USART. Zegar ten jest również używany przez wewnętrzny moduł przerwań, ale należy pamiętać, że niektóre zewnętrzne przerwania są wykrywane asynchroniczną logikę, co pozwala je wykrywać nawet jeżeli zegar I/O jest zatrzymany. Również należy zauważyć, że rozpoznawanie adresu w module TWI jest wykonywane asynchronicznie kiedy zegar I/O jest wstrzymany, umożliwiając pobór adresu TWI we wszystkich trybach snu.

- **Zegar Flash**

Zegar Flash kontroluje operacje interfejsu Flash. Jest zazwyczaj aktywny równolegle z zegarem CPU.

- **Asynchroniczny układ czasowy**

Asynchroniczny układ czasowy umożliwia asynchronicznemu licznikowi by był zliczany bezpośrednio z zewnętrznego krzemowego zegara 32kHZ. Dziedzina przeznaczenia tego zegara umożliwia taktowanie w czasie rzeczywistym nawet gdy urządzenie jest w trybie snu.

- **Zegar ADC**

ADC jest zaopatrzony w wyspecjalizowaną domenę zegara. Pozwala to zatrzymywać zegary CPU i I/O w celu redukcji zakłóceń generowanych przez cyfrowe obwody. To daje bardziej dokładne rezultaty konwersji w ADC.

- **Źródła zegara**

Urządzenie to ma następujące opcje źródeł taktowania, wybierane przez bity bezpiecznika Flash jak pokazano niżej. Zegar z wybranych źródeł jest wejściem do generatora AVR i ***(routed)*** do odpowiedniego modułu.

Opcje taktowania	CKSEL3.0
Zewnętrzny kwarcowy/ceramiczny rezonator	1111-1010
Zewnętrzny niskoczęstotliwościowy kwarc	1001
Zewnętrzny oscylator RC	1000-0101
Wykalibrowany wewnętrzny oscylator RC	0100-0001
Zewnętrzny zegar	0000

Uwaga: Dla wszystkich bezpieczników 1 oznacza nie zaprogramowany, podczas gdy 0 oznacza zaprogramowany.

W następnych sekcjach są podane różne wybory dla każdej opcji taktowania. Kiedy CPU budzi się ze stanu obniżonego poboru mocy lub oszczędności mocy korzysta się z wybranego źródła taktowania do procedury uruchamiającej, zapewniającej stabilizację oscylatora nim rozpocznie się wykonywanie operacji. Kiedy CPU startuje po resecie jest dodawane dodatkowe opóźnienie umożliwiające osiągnięcie stabilnego poziomy poprzez moc przed rozpoczęciem normalnego działania. Oscylator Watchdog'a jest używany do taktowania części uruchamiania w czasie rzeczywistym. Liczba cykli oscylatora WDT potrzebna w każdym z czasów przerwy pokazuje tabela 7. Częstotliwość Watchdog'a zależy od napięcia co opisuje „ATmega128 typowe charakterystyki" na stronie 326. Urządzenie to jest załadowane***(shipped)*** z CKSEL = „0001" i SUT = „10" (1MHz Wewnętrzny oscylator RC, wolno podnoszący moc).

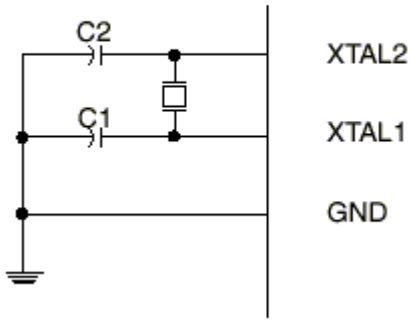
Typowy czas przerwy 5.0V	Typowy czas przerwy 3.0V	Liczba cykli
4.1ms	4.3ms	4K(4,096)
65ms	69ms	64K(65,536)

- **Kwarcowy oscylator**

XTAL1 i XTAL2 są indywidualnie wejściowym i wejściowym odwracającym wzmacniaczem , który może zostać skonfigurowany jako oscylator na płytce, co pokazuje rysunek 18. Każdy kwarcowy lub ceramiczny rezonator może być użyty. Bezpiecznik CKOPT wybiera pomiędzy dwoma różnymi trybami wzmacniacza oscylatorów. Kiedy CKOPT jest zaprogramowany, wyjście oscylatora będzie oscylował z pełnym wahaniem pomiędzy minimum i maksimum na wyjściu. Ten tryb jest odpowiedni dla operowania w bardzo głośnym środowisku lub kiedy wyjście z XTAL2 napędza drugi bufor zegara. Ten tryb ma szeroki zakres częstotliwości. Kiedy CKOPT jest nie zaprogramowane oscylator ma mniejsze wahania na wyjściu. Co zmniejsza pobór mocy. Tryb ten ma limitowany zakres częstotliwości i nie może być używany do napędzania innych buforów zegarów.

Dla rezonatorów minimalna częstotliwość to 8MHz z nie zaprogramowanym CKOPT i 16MHz z zaprogramowanym CKOPT. C1 i C2 powinny być zawsze równe dla oby kwarców i rezonatorów. Optymalna wartość dla kondensatorów zależy od kwarcu lub rezonatora, który jest używany wielkości rozproszonej reaktancji pojemnościowej i zakłóceń elektromagnetycznych środowiska. Niektóre początkowe wytyczne dla wyboru kondensatora dla pracy z kwarcem są oddane w tabeli 8. Dla ceramicznych rezonatorów wartości kondensatora zostały podane przez producenta. Większej ilości informacji na temat doboru kondensatorów znajduje się w nocie aplikacyjnej.

Figure 18. Crystal Oscillator Connections



Oscylator może pracować w trzech trybach, każdy jest zoptymalizowany dla specyficznego zakresu częstotliwości. Tryb pracy oscylatora wybierają bezpieczniki CKSEL3..1 jak pokazuje tabela 8.

CKOPT	CKSEL3..1	Zakres częstotliwości[1] MHz	Rekomendowany zakres dla kondensatorów C1 i C2 przy współpracy z kwarcem
1	101[2]	0.4-0.9	-
1	110	0.9 – 3.0	12pF – 22pF
1	111	3.0 –8.0	12pF – 22pF
0	101,110,111	1.0 -	12pF – 22pF

Uwaga: 1: zakres częstotliwości są wartościami wstępnymi. Wartości aktualne to TBD
2: opcja ta nie powinna być używana wraz z kwarcem, tylko z rezonatorem ceramicznym.

Bezpiecznik CKSEL0 wraz z bezpiecznikami SUT1..0 wybierają czasy uruchamiania co przedstawia tabela 9.

CKSEL0	SUT1..0	Czas startu po (wyłączenie zasilania, trybie oszczędzania mocy)	dodatkowe opóźnienie od resetu V = 5V	Rekomendowane użycie

0	00	258CK[(1)]	4.1 ms	Rezonator ceramiczny, szybki wzrost mocy
0	01	258CK[(1)]	64ms	Ceramiczny rezonator, wolniejszy wzrost mocy
0	10	1K CK[(2)]	-	Ceramiczny rezonator Odblokowany BOD
0	11	1K CK[(2)]	4.1ms	Ceramiczny rezonator Szybki wzrost mocy
1	00	1K CK[(2)]	65 ms	Ceramiczny rezonator Wolny wzrost mocy
1	01	16K CK	-	Kwarcowy oscylator Odblokowane BOD
1	10	16K CK	4.1ms	Kwarcowy oscylator Szybki wzrost mocy
1	11	16K CK	65 ms	Kwarcowy oscylator Wolny wzrost mocy

Uwagi: 1. Te opcje powinny być używane tylko kiedy nie operuje się na wartościach w pobliżu minimum częstotliwości urządzenia i tylko jeżeli stabilność częstotliwość przy uruchamianiu nie jest ważna dla programu. Te opcje nie są odpowiednie dla kwarców.
2. TE opcje mają współdziałać z ceramicznymi rezonatorami i zapewniają stabilną częstotliwość przy uruchamianiu. Mogą być one używane wraz z kryształami kiedy częstotliwość nie zbliża się do minimum i jest stabilna przy uruchamianiu oprogramowania.

- **Kryształowy oscylator o niskiej częstotliwości**

Aby korzystać z 32.768 kHz kwarcowego zegara jako źródła taktowania dla urządzenia kwarcowy niskoczęstotliwościowy oscylator musi zostać wybrany poprze ustawienie CKSEL na 1001. Kwarc powinien zostać podłączony jak pokazuje rysunek 18. Poprzez programowanie bezpiecznika CKOPT użytkownik może odbezpieczyć wewnętrzny kondensator n aXTAL1 i XTAL2 co sprawia, że niepotrzebne stają się zewnętrzne kondensatory. Wewnętrzny kondensator ma nominalną wartość 36pF. Dalsze informacje w nocie aplikacyjnej kwarcowego oscylatora 32MHz.

Kiedy ten oscylator jest wybrany czasy uruchamiania sa determinowane przez bezpieczniki SUT jak pokazuje tabela 10.

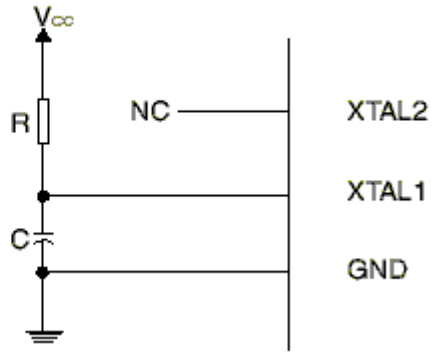
SUT1..0	Czas startu po (wyłączenie zasilania, trybie oszczędzania mocy)	dodatkowe opóźnienie od resetu V = 5V	Rekomendowane użycie
00	1K CK[(1)]	4.1ms	Pierwszy pobór mocy lub odblokowany BOD
01	1K CK[(1)]	65ms	Wolny wzrost mocy
10	32K CK	65ms	Stabilna częstotliwość przy uruchamianiu
11	Zarezerwowane		

Uwaga: 1. TE opcje nie powinny być wykorzystywane jeżeli stabilność częstotliwości przy uruchamianiu nie jest istotna dla aplikacji.

- **Zewnętrzny oscylator RC**

Przy taktowaniu nie wrażliwych aplikacji konfiguracja zewnętrznego RC pokazana jest na rysunku 19. Częstotliwość z grubsza oszacowana jest przez równanie f = 1/(3RC). C najmniej powinno wynosić 22pF. Poprzez programowanie bezpiecznika CKOPT użytkownik może odblokować wewnętrzny kondensator (36pF) pomiędzy XTAL1 i GND, co sprawia, że zewnętrzny kondensator staje się nie potrzebny. Więcej informacji w noce aplikacyjnej Zewnętrzny oscylator RC.

Figure 19. External RC Configuration



Oscylator może pracować w czterech różnych trybach, każdy jest zoptymalizowany dla specyficznego zakresu częstotliwości. Tryb operacji jest wybierany poprzez bezpiecznik CKSEL3..0 co pokazuje tabela 11.

CKSEL3..0	Zakres częstotliwości (MHz)
0101	-.0.9
0110	0,9-3,0
0111	3,0-8,0
1000	8,0-12,0

Kiedy oscylator jest wybrany czas uruchamiania seterminuje bezpiecznik SUT co przedstawia tabela 12.

SUT1..0	Czas startu po (wyłączenie zasilania, trybie oszczędzania mocy)	dodatkowe opóźnienie od resetu V = 5V	Rekomendowane Użycie
00	18 CK	-	Odblokowany BOD
01	18 CK	4.1ms	Szybki wzrost mocy
10	18 CK	65ms	Wolny wzrost mocy
11	6 CK(1)	4.1ms	Szybki wzrost mocy Odblokowany BOD

Uwaga: 1. Te opcje nie powinny być używane przy operowaniu na granicy minimalnej częstotliwości dal urządzenia.

- **Wzorcowy wewnętrzny oscylator RC**

Wzorcowy wewnętrzny oscylator zapewnia stały zegar o częstotliwości 1, 2, 4 i 8 MHz. Wszystkie te częstotliwości są wyznaczone dla napięcia nominalnego 5V w temperaturze 25 stopni C. Zegar ten może zostać wybrany jako zegar systemowy poprzez zaprogramowanie bezpieczników CKSEL jak pokazuje tabela 13. Jeżeli zostanie wybrany nie będzie współpracował z żadnymi zewnętrznymi elementami.

Bezpiecznik CKOPT powinien zawsze być nie zaprogramowany kiedy korzysta się z opcji zegara. Podczas resetu sprzęt ładuje bit wzorcowy do rejestru OSCCAL i następuje automatyczne wzorcowanie oscylatora RC. Przy napięciu 5 V w temperaturze 25 stopni częstotliwość oscylatora 1,0MHz nie odchyla się od nominalnej częstotliwości o więcej niż 1%. Kiedy oscylator jest wykorzystywany jako zegar modułu, oscylator Watchdog'a nadal będzie wykorzystywany do taktowanie Watchdog'a i czasu przerw w resecie. Więcej informacji zawartych jest w sekcji „Bajt wzorcowy" na stronie 285.

CKSEL3..0	Częstotliwość nominalna (MHz)
0001(1)	1.0
0010	2.0
0011	4.0
0100	8.0

Uwaga: Urządzenie jest załadowany z wybranej opcji

Kiedy ten oscylator zostanie wybrany, czas uruchamiania jest determinowany przez bezpieczniki SUT co pokazuje tabela 14. XTAL1 i XTAL2 powinny zostać nie podłączone.

SUT1..0	Czas startu po (wyłączenie zasilania, trybie oszczędzania mocy)	dodatkowe opóźnienie od resetu V = 5V	Rekomendowane Użycie
00	6 CK	-	Odblokowany BOD
01	6 CK	4.1ms	Szybki wzrost mocy
10(1)	6 CK	65ms	Wolny wzrost mocy
11	Zarezerwowane		

Uwaga: Urządzenie jest załadowany z wybranej opcji

- **Rejestr wzorcowego oscylatora OSCCAL**

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

Uwaga: rejestr OSCAL nie jest dostępny w trybie kompatybilności z ATmega103

a: bity 7..0 – CAL7..0 – wartości wzorcowe oscylatora

Zapisując bit wzorcowy pod ten adres wyregulujemy wewnętrzny oscylator by pozbyć się procesowych wahań z częstotliwości oscylatora. Operacja ta jest wykonywana automatycznie przy resecie. Dla OSCCAL równego zero najniższa dostępna częstotliwość została wybrana. Wpisywanie wartości niezerowych do tego rejestru zwiększa częstotliwość wewnętrznego oscylatora. Wpisanie $FF do rejestru umożliwia osiągnięcie najwyższej częstotliwości. Wzorcowy oscylator jest wykorzystywany do taktowania EEPROM i Flash. Jeżeli EEPROM lub Flash jest zapisywane nie należy kalibrować o więcej niż 10% nominalnej częstotliwości. Inaczej zapis do EEPROM i Flash może się nie udać. Należy zauważyć, że oscylator dąży do wzorcowych częstotliwości 1.0, 2.0, 4.0, 8.0. Dostrajanie do innych częstotliwości nie zapewnia bezpieczeństwa działania co pokazuje tabela 15.

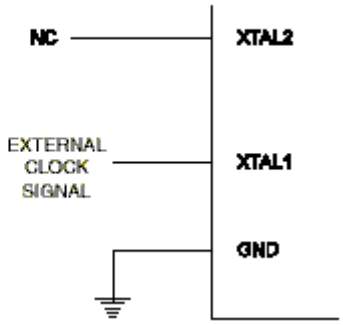
Wartości OSCCAL	Min. częstotliwość Jako odchylenie od	Max. częstotliwość Jako odchylenie od

	wartości nominalnej(%)	wartości nominalnej(%)
$00	55	100
$7F	75	150
$FF	100	200

- **Zewnętrzny zegar**

Aby napędzać urządzenie z zewnętrznego źródła , XTAL1 powinien być podłączony jak pokazuje rysunek 20. Aby urządzenie działało na zewnętrznym zegarze, bezpiecznik CKSEL musi być zaprogramowany na 0000. Poprzez programowanie bezpiecznika CKOPT użytkownik może odblokować wewnętrzny 36pF kondensator pomiędzy XTAL1 i GND

Figure 20. External Clock Drive Configuration



Kiedy to źródło zegara zostanie wybrane, czas uruchamiania jest determinowany przez bezpiecznik SUT co pokazuje tabela 16.

SUT1..0	Czas startu po (wyłączenie zasilania, trybie oszczędzania mocy)	dodatkowe opóźnienie od resetu V = 5V	Rekomendowane Użycie
00	6 CK	-	Odblokowany BOD
01	6 CK	4.1ms	Szybki wzrost mocy
10	6 CK	65ms	Wolny wzrost mocy
11	Zarezerwowane		

- **Oscylator licznika – stopera**

Dla mikrokontrolerów z wyjściami licznika – stopera (TOSC1 i TOSC2), kwarc podłączamy bezpośrednio pomiędzy te wyjścia. Nie potrzebne są żadne zewnętrzne kondensatory. Oscylator jest przystosowany do pracy z kwarcem o częstotliwości 32.768 MHz. Stosując zewnętrznego źródła zegara do TOSC1 nie jest zalecane.

  - **XTAL rejestr podziału kontroli – XDIV**

Rejestr podziału kontroli XTAL1 jest wykorzystywany do dzielenia źródła częstotliwości porze liczby z zakresu 2 – 129. Cecha ta jest wykorzystywana do zmniejszenia poboru mocy kiedy wymagania procesu są niskie.

Bit	7	6	5	4	3	2	1	0	
	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0	XDIV
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	

a: bit 7 XDIVEN :bit umożliwiający podział XTAL

Kiedy bit ten jest zapisany zerem częstotliwość zegara CPU i wszystkich urządzeń peryferyjnych jest dzielona przez czynnik zdefiniowany przez ustawienia pozostałych bitów rejestru. Ten bit może być zapisywany w czasie działania w celu zmiany częstotliwości zegara tak aby dostosować ją do aplikacji.

b: bity 6..0 – XDIV6..0: bity wyboru podziału XTAL

Bity te definiują czynnik podziału, który stosuje się kiedy bit XDIVEN jest ustawiony(zapisany na jeden). Jeżeli wartość tych bitów jest oznaczona przez d, następujące równanie definiuje częstotliwość zegara CPU i urządzeń peryferyjnych.

$$f_{CKL} = \frac{zegar\_źródowy}{129 - d}$$

Wartości tych bitów mogą być zmieniane tylko kiedy XDRIVEN jest równy zero. Kiedy XDIVEN jest zapisany jedynką, wartość zapisywana równolegle do XDIV6..0 jest brana jako czynnik podziałów. Kiedy XDIVEN jest równy zero wartość wpisywana równolegle do XDIV6..0 jest odrzucana. Jako, że dzielnik dzieli główne wejście zegara do MCU prędkość urządzeń peryferyjnych jest zredukowana kiedy czynnik podziału jest używany.
Uwaga: Licznik0 nie powinien być używany kiedy zegar systemowy jest dzielony. Odkąd Licznik służy również jako asynchroniczny licznik – stoper zegar nie będzie dzielony dla danego modułu zgodnie z ustawieniami rejestru XDIV nawet jeżeli Licznik – stoper pracuje synchronicznie. W konsekwencji przerwania mogą się gubić i dostęp do licznika – stopera może zakończyć się niepowodzeniem.

ZARZĄDZANIE POBOREM MOCY I TRYBY SNU

Tryby sny umożliwiają aplikacji zamykać moduły, z których nie korzystają w MCU, co oszczędza moc. AVR zapewnia różne tryby snu umożliwiające użytkownikowi dostosować pobór mocy do wymagań aplikacji/

Aby przejść w jeden z sześciu trybów snu należy wpisać logiczną jedynkę do bitu SE w MCUCR i rozkaz SLEEP musi zostać wykonany. Bity SM2..0 w rejestrze MCUCR wybierają tryb snu (Idyl, redukcja szumów ADC, obniżenie poboru mocy, oszczędności mocy, oczekiwanie, rozszerzone oczekiwanie), który jest aktywowany przez rozkaz SLEEP. Streszczenie w tabeli 17. Jeżeli odblokowane przerwanie zajdzie podczas gdy MCU jest w trybie snu, MCU budzi się. MCU jest wtedy wstrzymywane na cztery dodatkowe cykle maszynowe podczas uruchamiania, następnie jest rozpoczynana obsługa przerwania i jest wznawiane wykonywanie następujące po rozkazie SLEEP. Zawartość zestawu rejestrów i SRAM pozostaje niezmieniona kiedy urządzenie budzi się z trybu snu. Kiedy w czasie trybu snu nastąpi reset, MCU budzi się i zaczyna wykonywać rozkazy od wektora resetu.

Rysunek 17 na stronie 33 przedstawia różne systemy zegarowe a ATmega128 i ich rozkład. Rysunek ten jest pomocny przy wybieraniu odpowiedniego trybu snu.

- **Rejestr kontroli MCU – MCUCR**

Rejestr kontroli MCU zawiera bity kontrolne dla zarządzania mocą.

Bit	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUCR
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	

a: bit 5 SE: bit uaktywniający spanie

Bit ten musi zostać zapisany przez logiczną jedynkę, aby MCU weszło w tryb snu kiedy zostanie wykonany rozkaz SLEEP. Aby uniknąć wejścia MCU w tryb snu jeśli nie jest to celem programisty, należy wpisywać do SE jedynkę tylko przed wykonywaniem rozkazu SLEEP i wyzerować ten bit zaraz po przebudzeniu.

b: bity 4..2 – SM2..0: bity wyboru trybu snu

Ustawiając te bity można wybrać jeden z sześciu trybów snu, co przedstawia tabela 17.

SM2	SM1	SM0	Tryb snu
0	0	0	Idyl
0	0	1	Redukcja zakłóceń ADC
0	1	0	Tryb obniżonego poboru mocy
0	1	1	Tryb oszczędności mocy
1	0	0	Zarezerwowany
1	0	1	Zarezerwowany
1	1	0	Tryb oczekiwania(1)
1	1	1	Rozszerzony tryb oczekiwania(1)

Uwaga: 1. Tryb oczekiwania i rozszerzony tryb oczekiwania są dostępne tylko przy zewnętrznym kwarcu lub rezonatorze.

- **Idyll tryb**

Kiedy do bitów SM2..0 zostanie wpisane 000 rozkaz SLEEP sprawia, żę MCU wchodzi w tryb Idyll zatrzymując pracę CPU, ale pozwalając SPI, USART, komparatorowi analogowemu, ADC, interfejsowi łącza szeregowego dwuprzewodowego, Licznikowi – stoperowi, Watchdog'owi i systemowi przerwań kontynuować działanie. Ten tryb snu zasadniczo zatrzymuje clk_{CPU} i clk_{Flash} pozwalając pozostałym zegarom pracować.

Tryb idyll umożliwia MCU budzenie się gdy wystąpi zewnętrzne wywołanie przerwania jak również wewnętrzne takie jak przepełnienie układu czasowego i przerwanie zakończenia transmisji USART. Jeżeli wybudzenie nie jest potrzebne przy przerwaniu pochodzącemu od analogowego komparatora, komparator może otrzymać mniej mocy przez ustawienie bitu ACD w rejestrze kontroli i statusu analogowego komparatora ACSR. To zmniejszy pobór mocy w trybie idyll. Jeżeli ADC jest odblokowane, konwersja zaczyna się automatycznie przy wejściu w ten tryb.

- **Tryb redukcji szumów ADC**

Kiedy bity SM2..0 mają wpisaną wartość 001 rozkaz SLEEP sprawi, że MCU wejdzie w tryb redukcji szumów, zatrzymując działanie CPU, ale pozwalając pracować ADC, zewnętrznemu układowi przerwań,

kontrolerowi adresów interfejsu dwuprzewodowego szeregowego interfejsu, Licznikowi- stoperowi i Watchdog'owi jeżeli ma możliwość pracy. Ten tryb snu zasadniczo zatrzymuje $clk_{I/O}$ i clk_{CPU} i clk_{Flash} pozwalając pozostałym zegarom pracować.

To udoskonala środowisko zakłóceń dla ADC, umożliwiając wyższą rozdzielczość pomiarów. Jeśli ADC jest odblokowane , konwersja zaczyna się automatycznie kiedy urządzenie wejdzie w ten tryb. Poza kompletną konwersją przerwań ADC tylko zewnętrzny reset, Watchdog reset, reset spowodowany obniżeniem napięcia sieciowego, przerwanie oznajmiające zgodność adresów szeregowego dwuprzewodowego interfejsu, przerwania licznika- stopera, przerwanie gotowości SPM/EEPROM, przerwanie zewnętrznego poziomu INT7..4 lub przerwanie zewnętrzne INT3..0 mogą wybudzić MCU z tego trybu snu.

- **Tryb obniżonego poboru mocy**

Kiedy na bity SM2..0 wpiszemy 010 rozkaz SLEEP spowoduje wejście urządzenia w tryb obniżonego poboru mocy. W tym trybie, zewnętrzny oscylator jest zatrzymywany, podczas gdy przerwania zewnętrzne , kontroler adresów interfejsu szeregowego dwuprzewodowego, i Watchdog kontynuują pracę. Tylko zewnętrzny reset, reset Watchdog'a, reset spowodowany obniżeniem napięcia sieciowego, przerwanie dopasowania adresu szeregowego dwuprzewodowego interfejsu, zewnętrznego poziomu przerwania INT7..4 lub zewnętrznego przerwania INT3..0 poże wybudzić MCU. Ten tryb snu zasadniczo zatrzymuje wszystkie zegary pozwalając pracować tylko modułom asynchronicznym.

Należy zauważyć, że wyzwalacz poziomu przerwania jest używany do wybudzania urządzenia z tego trybu snu, zmieniony poziom musi być przytrzymany przez jakiś czas. Szczegóły na stronie 84 w „Przerwania zewnętrzne".

Po przebudzeniu z tego trybu snu następuje opóźnienie przy stanie przebudzenia trwające do czasu aż stan przebudzenia będzie aktywny. To pozwala zegarowi zrestartować się i stać się stabilnym po zatrzymaniu. Okres przebudzenia jest zdefiniowany przez bezpiecznik CKSEL, który definiuje okres czasu przerwy restartu, co opisuje sekcja „Źródła zegara" na stronie 34.

- **Tryb oszczędności mocy**

Kiedy na bitach SM2..0 jest wpisane 011 rozkaz SLEEP sprawia, że MCU wchodzi w stan oszczędności mocy. Tryb ten jest podobny od trybu obniżenia pobieranej mocy z jednym wyjątkiem:

Jeżeli licznik – stoper jest taktowany asynchronicznie , np. bit AS0 w ASSR jest ustawiony na 1, licznik – stoper będzie pracował podczas snu. Urządzenie może wybudzić się ze snu albo ze względu na przepełnienie licznika, albo na zdarzenie zewnętrznego porównania z licznikiem – stoperem jeżeli odpowiedni bit przerwania licznika-stopera jest odblokowany w TIMSK i również jest odblokowany globalny bit przerwań w SREG.

Jeśli asynchroniczny zegar nie jest taktowany asynchronicznie tryb obniżenie poboru mocy jest rekomendowany zamiast trybu oszczędności mocy ponieważ zawartość rejestrów w asynchronicznym liczniku powinna być brana pod uwagę jako nie zdefiniowana po każdym wybudzeniu z trybu oszczędności mocy jeżeli AS0 jest 0.

Ten tryb generalnie wstrzymuje wszystkie zegary z wyjątkiem clk_{ASY} , pozwalając na działanie tylko operacjom w asynchronicznych modułach, w skład których wchodzi również licznik –stoper jeżeli jest taktowany asynchronicznie.

- **Tryb oczekiwania**

Kiedy na bitach SM2..0 jest ustawiona 110 i opcja zewnętrznego kwarcu lub rezonatora zegara jest wybrana, rozkaz SLEEP wprowadza MCU w stan oczekiwania. Ten tryb jest podobny do trybu oszczędności mocy z tym wyjątkiem, że nadal pracuje oscylator. Z rozszerzonego trybu oczekiwania urządzenie wybudza się w czasie sześciu cykli maszynowych.

- **Rozszerzony tryb oczekiwania**

Kiedy na bitach SM2..0 są ustawione 111 opcja zewnętrznego kwarcu i rezonatora jest wybrana, rozkaz SLEEP sprawia, że MCU przechodzi w tryb rozszerzonego oczekiwania. Ten tryb jest podobny do trybu oszczędności mocy z tym wyjątkiem, że nadal pracuje oscylator. Z rozszerzonego trybu oczekiwania urządzenie wybudza się w czasie sześciu cykli maszynowych.

	Aktywna domena zegara					oscylatory		Źródła budzenia				
Tryby snu	clk_{CPU}	clk_{FLASH}	clk_{ID}	clk_{ADC}	clk_{ASY}	źródło odbokowywujące główny zegar	Odblokowanie zegara OSC	INT7..0	TWI Dopasowanie adresu	Licznik 0	gotowość SPM/ EEPROM	A
Idyl			X	X	X	X	X(2)	X	X	X	X	
ADC Redukcja Zakłóceń				X	X	X	X(2)	X(3)	X	X	X	
Obniżenie Poboru Mocy								X(3)	X			
Oszczędność mocy					X(2)		X(2)	X(3)	X	X(2)		
Oczekiwanie(1)						X		X(3)	X			
Rozszerzone oczekiwanie(1)					X(2)	X	X(2)	X(3)	X	X(2)		

Uwagi: 1 .Zewnętrzny kwarc lub rezonator jest wybrany jako źródło zegara
2. Jeśli bit AS0 w ASSR jest ustawiony na 1
3. Tylko INT3..0 lub poziom przerwań INT7..4

- **MINIMALIZACJA POBORU MOCY**

Jest wiele wyników, które należy wziąć pod uwagę kiedy próbuje się zmniejszyć pobór mocy w systemie kontrolowanym przez AVR. Generalizując, tryby snu powinny być wykorzystywane jak najczęściej jest to możliwe i powinny być tak dobierane, aby jak najmniej funkcji urządzeń pracowało. Wszystkie funkcje nie potrzebne powinny być wyłączone. W szczególności następujące moduły należy wziąć pod specjalną uwagę przy próbach uzyskania najniższych możliwych poborów mocy.

  - **Konwerter analogowo – binarny**

Jeżeli jest możliwe, ADC będzie umożliwiało wszystkie tryby snu. W celu zaoszczędzenia mocy, ADC powinno być odcięte przed przejściem w jakikolwiek z trybów snu. Kiedy ADC jest wyłączane i włączane na zmianę następna konwersja będzie konwersją oczekiwaną. Szersze informacje w sekcji „konwerter analogowo –binarny” na stronie 225.

- **Komparator analogowy**

Wchodząc w tryb Idyl, komparator analogowy powinien być zablokowany jeżeli nie jest używany. Wchodząc w tryb redukcji zakłóceń, analogowy komparator powinien być wyłączony. W pozostałych trybach snu analogowy komparator jest automatycznie odcinany. Jednakże, gdy analogowy komparator jest ustawiony do korzystania z wewnętrznego odniesienia napięcia jako wejścia, analogowy komparator powinien być wyłączony we wszystkich trybach snu. Dalsze szczegóły na stronie 222 w sekcji „analogowy komparator".

- **Detekcja obniżenia napięcia sieciowego**

Jeżeli detektor obniżonego napięcia sieciowego jest niepotrzebny dla aplikacji moduł ten powinien być wyłączony. Jeżeli detektor jest odblokowany poprzez bezpiecznik BODEN, będzie on odblokowany we wszystkich trybach snu i będzie pobierał moc. W głębszych trybach snu przyczyni się to znacznie do całkowitego poboru prądu. Szczegóły w sekcji „Detektor obniżenia napięcia" na stronie 44.

- **Wewnętrzne odniesienie napięcia**

Wewnętrzne odniesienie napięcia będzie odblokowane kiedy będzie potrzebne przez detektor obniżenia napięcia, analogowy komparator lub ADC. Jeżeli te moduły są odcięte jak opisano w powyższych sekcjach, wewnętrzne odniesienie napięcia będzie odcięte i nie będzie pobierało mocy. Po ponownym włączeniu, użytkownik musi pozwolić odniesieniu wystartować nim wyjście jest używane. Jeżeli odniesienie jest utrzymywane w trybie snu, wyjście może wykorzystane natychmiastowo. Więcej informacji w sekcji „wewnętrzne odniesienie napięcia" na stronie 50.

- **Regulator czasowy Watchdog'a**

Jeżeli regulator czasowy Watchdog'a nie jest potrzebny aplikacji, moduł ten powinien być wyłączony. Jeżeli jest on włączony, będzie on także działał w trybie snu i w związku z tym pobierał moc. W głębszych trybach snu przyczyni się to znacznie do całkowitego poboru prądu. Szczegóły w sekcji „Regulator czasowy Watchdog'a" na stronie 50.

- **Wyjścia portów**

Wchodząc w tryb snu wszystkie wyjścia portów powinny być skonfigurowane, tak, aby korzystać z najmniejszej mocy. Najważniejszą rzeczą jest by zapewnienie żadne z wyjść nie miało obciążenia. W trybie snu gdzie zarówno zegar I/O jak i zegar ADC są zatrzymane, bufor wejściowy urządzenia będzie odcięty. Zapewnienia to, że nie będzie pobierana żadna moc przez logiczne wejście kiedy nie jest to potrzebne. W niektórych przypadkach wejście logiczne jest potrzebne, aby wykryć kondycję wybudzania i wtedy będzie odblokowane. Szczegóły w sekcji „Binarne wejście i tryby snu" na stronie 64. Jeżeli bufor wejściowy jest odblokowany i sygnały wejściowe są nie uziemione lub mają analogowy sygnał bliski V/2, bufor wejściowy będzie korzystał z nadmiernej mocy.

SYSTEM KONTROLI I RESET

- **Resetowanie AVR**

W czasie resetu wszystkie rejestry I/O są ustawione na swoją wartość początkową i program zaczyna pracę od wektora resetu. Rozkazem umieszczonym w wektorze resetu musi być JPM – skok bezwzględny

– do rozkazów obsługujących procedurę resetu Jeżeli program nigdy nie uaktywnia źródeł przerwań , wektory przerwań są niewykorzystywane i kod programu może być zapisany w tych lokacjach. Ten przypadek dotyczy również wektora resetu w sekcji programowej pamięci podczas gdy wektory przerwań są w sekcji ładowalnej i vice versa. Schemat połączeń na rysunku 21 pokazuje resetowanie logiczne. Tabela 19 definiuje elektryczne parametry tego obwodu.

Porty I/O w AVR są natychmiastowo resetowane do ich początkowych stanów kiedy uaktywnia się wejście resetu. Nie wymaga to żadnego działającego zegara.

Kiedy wszystkie źródła resetu staną się nieaktywne, licznik opóźnienia jest przywoływany rozciągając czas trwania wewnętrznego resetu. Pozwala to mocy osiągnąć stabilny poziom przed rozpoczęciem normalnego działania. Czas przerwy licznika opóźnienia jest zdefiniowany przez użytkownika poprzez bezpieczniki CKSEL. Szczegóły w sekcji „Źródła zegara” na stronie 34.

- **Źródła resetu**

ATmega128 ma pięć źródeł resetu:

- Reset przy włączonej mocy. MCU jest resetowane kiedy zasób napięcia spadnie poniżej progu minimalnej mocy.
- Reset zewnętrzny. MCU jest resetowane kiedy na wejściu RESET(zanegowane) jest niski poziom dłużej niż minimalna długość impulsu.
- Reset Watchdog’a. MCU jest resetowane kiedy okres licznika Watchdog’a wygaśnie i Watchdog zostanie odblokowany.
- Reset przy obniżonym napięciu zasilania. MCU jest resetowane kiedy zasób napięcia spadnie poniżej poziomu i detektor obniżonego napięcia zasilania jest odblokowany.
- Reset JTAG AVR. MCU jest resetowane tak długo jak w rejestrze resetu jest wpisana logiczna jedynka, jeden ze skanowanych łańcuchów systemu JTAG. Szczegóły w sekcji „ IEEE 1149.1 (JTAG) przeszukiwanie obszaru granicznego” na stronie 248.

Figure 21. Reset Logic

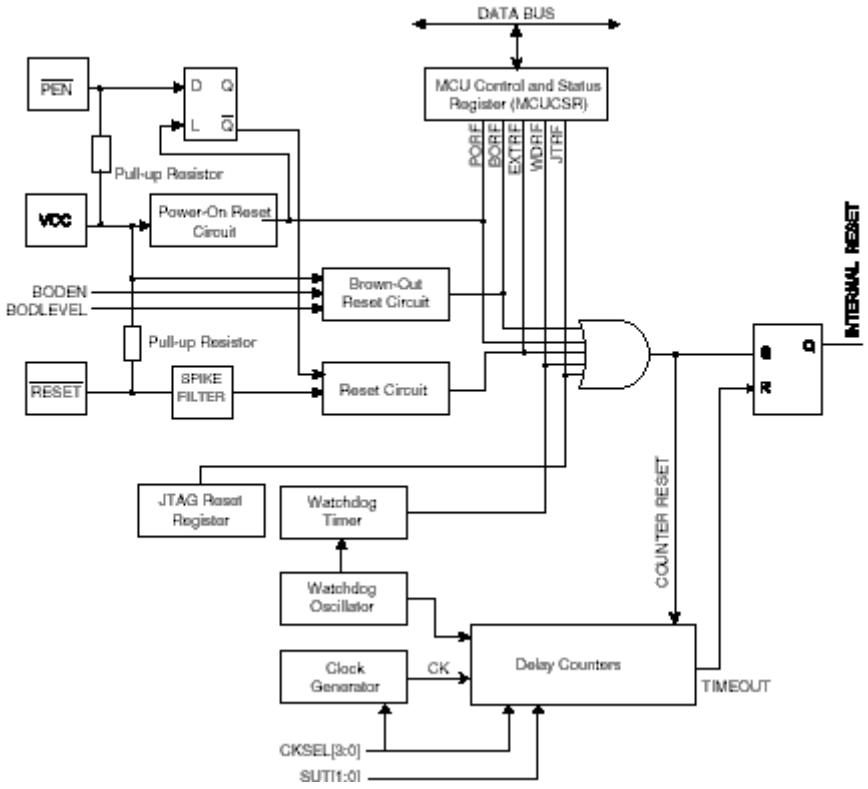


Tabela 19.

Symbol	Parametry	Warunek	Minimum	Typ	Maksimum	Jednostka
V_{POT}	Reset przy poziomie napięcia przy włączaniu.(narastanie)			1.4	2.3	V
	Reset przy poziomie napięcia przy włączaniu. (opadanie)(1)			1.3	2.3	V
V_{RST}	Poziom napięcia na wejściu RESET		0.2 V_{cc}		0.85 V_{cc}	V
t_{RST}	Minimalny czas trwania impulsu Wejścia RESET			50		ns
V_{BOT}	Poziom napięcia przy obniżonym zasialaniu sieciowym	BODLEVEL =1	2.5	2.7	3,2	V
		BODLEVEL = 0	3,7	4,0	4,2	
t_{BOD}	Minimalny okres niskiego napięcia dla detektora obniżonego napięcia sieci	BODLEVEL = 1		2		Mikrosekund
		BODLEVEL = 1		2		
V_{HYST}	Histereza detektora obniżonego napięcia sieciowego			50		mV

Uwaga: 1. Reset ten nie będzie działał dopóki zasób napięcia nie będzie poniżej V_{POT} (opadanie)

- **Reset przy włączonej mocy**

Impuls resetu przy włączonej mocy (POR) jest generowany przez układ detekcji w strukturze. Poziom detekcji jest przedstawiony w tabeli 19. POR jest aktywowany zawsze, kiedy V_{cc} jest poniżej poziomy detekcji. Obwód POR może być wykorzystany jak czynnik powodujący Reset przy uruchamianiu, jak również jako wykrywacz błędów przy dostarczaniu napięcia.

Obwód POR zapewnia, że urządzenie jest resetowane przy włączaniu zasilania. Osiągając poziom napięci a określony dla resetu przy uruchamianiu przywołuje licznik opóźnienia, co determinuje jak długo urządzenie jest zresetowane po podniesieniu V_{cc}. Sygnał RESET jest aktywowany , beż żadnego opóźnienia, kiedy V_{cc} opadnie poniżej poziomu detekcji.

Figure 22. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}.

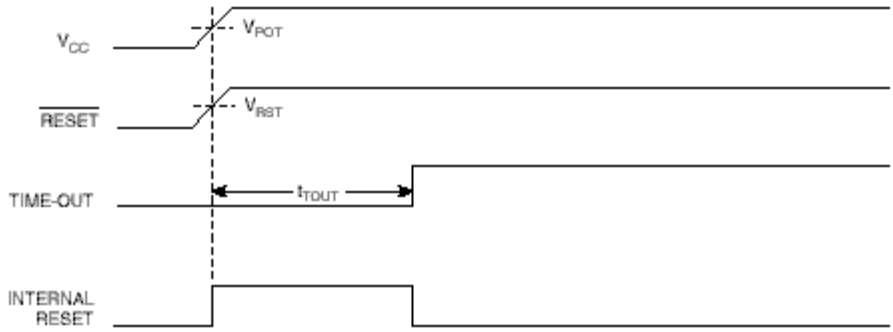
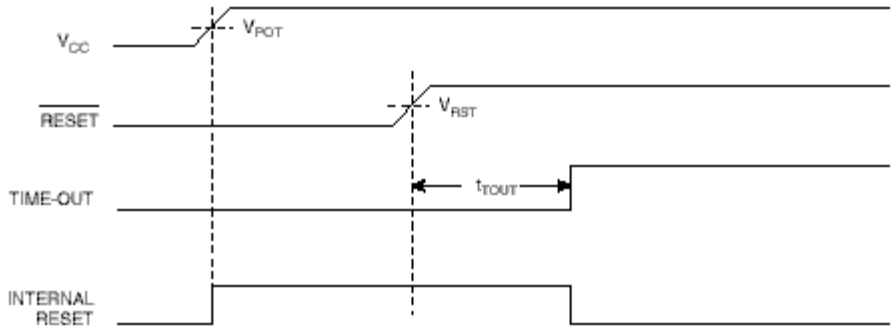


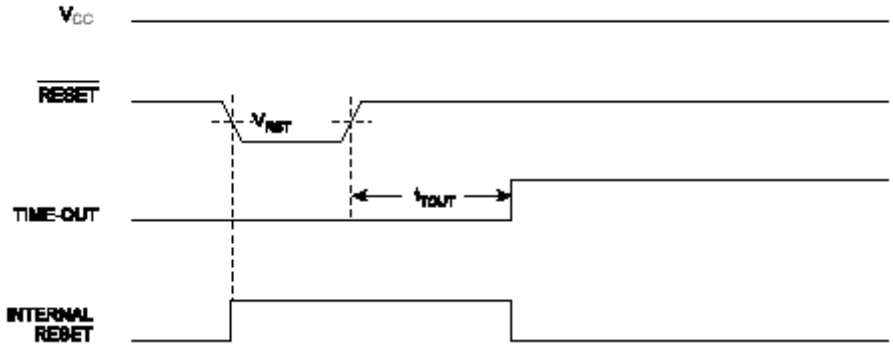
Figure 23. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



- **Zewnętrzny reset**

Zewnętrzny reset jest generowany poprzez niski poziom wyjścia RESET. Impuls resetu musi trwać dłużej niż minimalna szerokość impulsu podana w tabeli 19 aby wygenerować reset, nawet jeśli zegar nie pracuje. Krótsze impulsy nie gwarantują wygenerowania resetu. Kiedy zastosowany sygnał osiągnie napięcie poziomu resetu V_{RST} na swoim dodatnim zboczu, licznik opóźnienia uruchamia MCU po okresie równym czasowi przerwy t_{OUT}.

Figure 24. External Reset During Operation



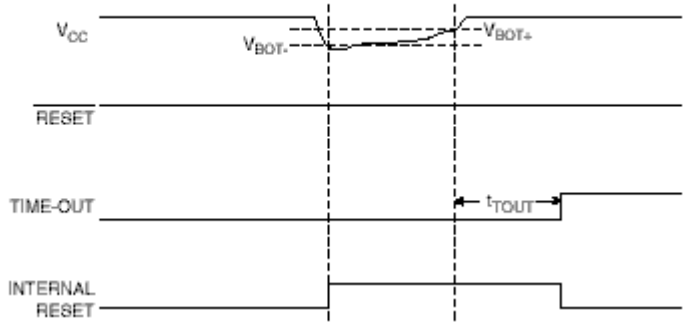
- **Wykrywanie obniżonego napięcia sieciowego**

Atmega128 ma wbudowany obwód detekcji obniżenia napięcia zasilania (BOD) w celu monitorowania poziomu V_{cc} podczas działania poprzez porównywanie napięcia do zewnętrznego poziomu porównawczego. Poziom porównawczy dla BOD może zostać wybrany poprzez ustawienie bezpiecznika BODLEVEL na 2.7V(nie zaprogramowany) lub 4.0V (zaprogramowany). Poziom porównawczy jest histerezą w celu zapewnienia ostrego impulsu uwalniający detekcję obniżonego napięcia zewnętrznego. Histereza poziomu detekcji powinna być interpretowana jako $V_{BOT+} = V_{BOT} + V_{HYST}/2$ i $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

Obwód BOD może być odcinany i dołączany poprzez bezpiecznik BODEN. Kiedy BOD jest dołączone (BODEN zaprogramowany) i V_{cc} opada do wartości poniżej poziomu porównawczego (V_{BOT-} na rysunku 25), reset przy obniżonym napięciu zasilania jest natychmiastowo aktywowany. Kiedy V_{cc} wzrasta powyżej poziomu porównawczego (V_{BOT+} na rysunku 25) licznik opóźnienia uruchamia MCU po okresie równym czasowi przerwy.

Obwód BOD wykryje tylko spadek V_{cc} jeżeli napięci pozostaje poniżej poziomu porównawczego dłużej niż t_{BOD} podany w tabeli 19.

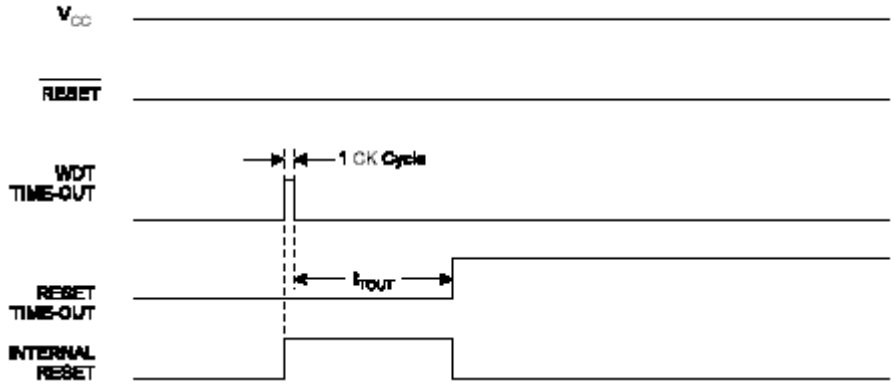
Figure 25. Brown-out Reset During Operation



- **Reset Watchdog’a**

Kiedy czas przerwy Watchdog’a upłynie zostanie wygenerowany krótki impuls resetu o trwaniu 1CK cyklu maszynowego. Na opadającym zboczu tego impulsu licznik opóźnienia zaczyna zliczanie czasu okresu przerwy t_{TOUT}. Szczegóły na stronie 50 przy opisie Watchdog’a.

Figure 26. Watchdog Reset During Operation



- **Rejestr statusu i kontroli MCU – MCUCSR**

Rejestr kontroli i statusu MCU zapewnia informacje na podstawi której źródło resetu powoduje zresetowanie MCU.

Bit	7	6	5	4	3	2	1	0	
	JTD	–	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

Należy zauważyć, że tylko EXTRF i PORF są dostępne w trybie kompatybilności z ATmega103.

a: bit 4 – JTRF: flaga resetu JTAG

Ten bit jest ustawiony jeżeli przyczyną resetu było wpisanie jedynki logicznej do rejestru resetu JTAG wybranego przez rozkaz AVR_RESET. Bit ten jest resetowany przez reset przy włączonej mocy lub poprzez przypisanie mu logicznego zera.

b: bit 3 – WDRF: flaga resetu Watchdog'a

Ten bit jest ustawiany jeżeli zajdzie reset Wathcdog'a. Jest on resetowany przez reset przy włączonej mocy lub przez wpisanie mu zera.

c: bit 2 – BORF: flaga resetu przy obniżonym napięciu sieci

Bit ten jest ustawiony jeżeli zajdzie reset przy obniżonym napięciu zasilania. Jest on resetowany przez reset przy włączonej mocy lub przez wpisanie mu zera.

d: bit 1 – EXTRF: flaga zewnętrznego resetu

Bit ten jest ustawiony jeżeli zajdzie zewnętrzny reset. Jest on resetowany przez reset przy włączonej mocy lub przez wpisanie mu zera.

e: bit 0 – PORF: flaga resetu przy włączaniu mocy

Bit ten jest ustawiony jeżeli zajdzie reset przy włączonej mocy. Jest on resetowany tylko przez wpisanie mu zera.

Aby korzystać z flag resetu do rozróżniania poszczególnych warunków wystąpienia resetu, użytkownik powinien odczytać i dopiero później resetować MCUCSR najwcześniej jak tylko to możliwe w programie. Jeżeli rejestr ten jest wyzerowany nim zajdzie inny reset, źródło resetu może zostać ustalone poprzez sprawdzanie flag reset.

- **Odniesienie do wewnętrznego napięcia**

Cechy ATmega128 jako wewnętrznego odniesienia dla pasma wzbronionego. Odniesienie to jest wykorzystywane przy detekcji obniżonego napięcia sieciowego i może być wykorzystywane jako wejście dla analogowego komparatora lub ADC. Napięcie odniesienia 2.56 do ADC jest generowane poprzez wewnętrzne odniesienie pasma wzbronionego.

- **Odniesienie do napięcia odblokowującego sygnały i czas uruchamiania**

Odniesienie napięcia ma czas uruchamiania, który może wpływać na sposób w jaki będzie używany. Czasy uruchamiania są podane w tabeli 20. Aby oszczędzać moc, odniesienie to nie zawsze jest włączone. W następujących sytuacja odniesienie jest włączone:

1. Kiedy BOD jest odblokowany (poprzez programowanie bezpiecznika BODEN)
2. Kiedy odniesienie pasma wzbronionego jest połączone z analogowym komparatorem (poprzez ustawienie ACBG i ACSR)
3. Kiedy ADC jest odblokowane.

Dlatego, kiedy BOD nie jest odblokowany, po ustawieniu bitu ACBG lub odblokowaniu ADC, użytkownik może zawsze pozwolić by odniesienie do uruchamiania przed wyjściem z analogowego komparatora lub ADC było używane. W celu redukcji zużywanej mocy w trybie obniżonego poboru mocy, użytkownik może pominąć trzy powyższe warunki aby zapewnić, że odniesienie jest wyłączone przed wejściem w tryb obniżonego poboru mocy.

Tabela20.

Symbol	Parametr	Minimum	Typ	Maksimum	Jednostka
V_{BG}	Napięcie odniesienia pasma zabronionego	1,15	1,23	1,35	V
t_{BG}	Czas uruchamiania odniesienia pasma zabronionego		40	70	µs
I_{BG}	Pobór prądu dla odniesienia w aśmie zabronionym		10		µA

- **Taktowanie Watchdog'a**

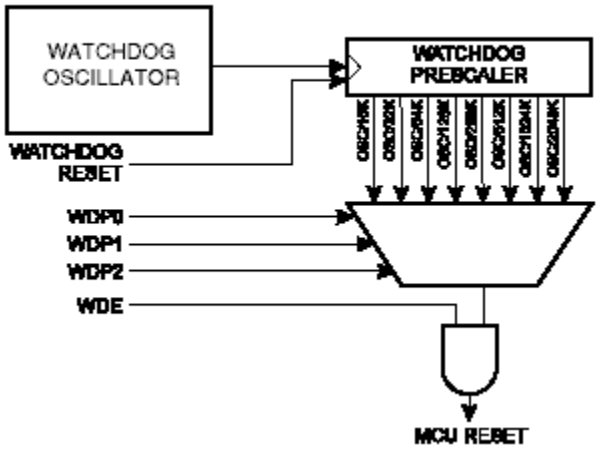
Watchdog jest taktowany przy użyciu oddzielnego wbudowanego oscylatora, który pracuje z częstotliwością 1MHz. Jest to typowa wartość dla V_{cc} = 5V. Przejrzyj charakterystykę danych dla typowych wartości przy innych poziomach napięcia V_{cc}. Poprzez kontrolowanie prescalera Watchdog'a, przerwa wywołania resetem Watchdog'a może być dopasowywana co pokazuje tabela 22 na stronie 52. Rozkaz WDR – Watchdog Reset – resetuje taktowanie Watchdog'a. Jest ono także resetowane kiedy Watchdog zostaje odcięty lub gdy zajdzie resetowanie modułu. Osiem różnych okresów cykli maszynowych może być wybranych aby zdeterminować okres resetu. Jeśli okres resetu trwa bez innego resetu Watchdog'a, ATmega128 resetuje się i rozpoczyna pracę od wektora resetu. Szczegóły czasowe na temat resetu Wwatchdog'a na stronie 49.

Aby zapobiec nie przewidzianym odłączeniom Watchdog'a lub zmiano okresu czasu przerwy, trzy różne poziomy bezpieczeństwa są wybierane przez bezpieczniki M103C i WDTON co pokazuje tabela 21. Poziom bezpieczeństwa 0 odpowiada ustawieniom ATmega103. Nie ma tutaj żadnych ograniczeń na odblokowywanie WDT w jakimkolwiek z trybów bezpieczeństwa, Szczegóły w „Sekwencje czasowe dla zmiany konfiguracji taktowania Watchdog'a" na stronie 53.

Tabela 21.

M103C	WDTON	Poziom bezpieczeństwa	Początkowy stan WDT	Jak zablokować WDT	Jak zmienić czas przerwy
Nieprogramowany	Nieprogramowany	1	Zablokowany	Sekwencja czasowa	Sekwencja czasowa
Nieprogramowany	Programowany	2	Odblokowany	Zawsze odblokowany	Sekwencja czasowa
Nieprogramowany	Nieprogramowany	0	Zablokowany	Sekwencja czasowa	Bez ograniczeń
Nieprogramowany	programowany	2	Odblokowany	Zawsze odblokowany	Sekwencja czasowa

Figure 27. Watchdog Timer



- **Rejestr kontroli taktowania Wathcdog'a WDTCR**

Bit	7	6	5	4	3	2	1	0	
	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

a: bity 7..5 – Res: bity zarezerwowane

Bity te są zarezerwowane i w ATmega128 zawsze będą zerem.

b: bit 4 – WDCE: odblokowanie zmian Watchdog'a

Bit ten musi zostać ustawiony kiedy bit WDE jest zapisany logiczną jedynką. W innym razie, Watchdog nie będzie zablokowany. Raz zapisany jedynką, sprzęt wyzeruje wszystkie bity po czterech cyklach zegarowych. Szczegóły w opisie bitu WDE dla procedury blokowania Watchdog'a. W poziomach bezpieczeństwa 1 i 2, bit ten musi także być ustawiony, kiedy zmieniany są bity prescalera. Szczegóły w „Sekwencje czasowe dla zmiany konfiguracji taktowania Watchdog'a" na stronie 53.

c: bit 3 – WDE: odblokowanie Watchdog'a

Kiedy do tego bitu jest wpisana logiczna jedynka, taktowanie Watchdog'a jest odblokowane i jeżeli WDE jest zapisane zerem, funkcje Watchdog'a są zablokowane. WDE może być wyzerowane tylko jeżeli bit WDCE jest ustawiony. Aby odblokowywać i zablokowywać taktowanie Watchdog'a następująca procedura musi być przestrzegana:

1. W jednej operacji należy wpisać logiczną jedynkę do WDCE i WDE. Logiczna jedynka musi być wpisana do WDE nawet jeżeli jest on już wcześniej ustawiony na jeden.
2. W ciągu następnych czterech cykli zegarowych należy wpisać zero do WDE. To blokuje Watchdog'a.

Na 2 poziomie bezpieczeństwa nie można zablokować taktowanie Watchdog'a nawet algorytmem opisanym powyżej. Szczegóły w „Sekwencje czasowe dla zmiany konfiguracji taktowania Watchdog'a" na stronie 53.

d: bit 2..0 – WDP2..0: prescaler taktowania Watchdog'a 2,1,0

Bity te determinują czy taktowanie Watchdog'a będzie przeliczane kiedy taktowanie Watchdog'a jest odblokowane. Różne wartości przelicznika i odpowiadające im okresy czasów przerw opisuje tabela 22.

WDP2	WDP1	WDP0	Liczba cykli oscylatoraWDT	Typowe wartości czasów przerwy dla V_{cc} = 3.0V	Typowe wartości czasów przerwy dla V_{cc} = 5.0V
0	0	0	16K (16,384)	17,1ms	16,3ms
0	0	1	32K (32,768)	34,3ms	32,5ms
0	1	0	64K (65,536)	68,5ms	65ms
0	1	1	128K (131,072)	0,14s	0,13s
1	0	0	256K (262,144)	0,27s	0,26s
1	0	1	512K (524,288)	0,55s	0,52s
1	1	0	1024K (1048,576)	1,1s	1,0s
1	1	1	2048K (2097,152)	2,2s	2,1s

Następujące przykłady kodu pokazują jak wyłączyć WDT. Przykłady te zakładają, że przerwania są kontrolowane (np., przez zablokowanie globalnego bitu przerwań) tak, że nie nastąpi przerwanie w czasie wykonywania tych procedur.

Assembly Code Example

```
WDT_off:
  ; Write logical one to WDCE and WDE
  ldi  r16, (1<<WDCE)|(1<<WDE)
  out  WDTCR, r16
  ; Turn off WDT
  ldi  r16, (0<<WDE)
  out  WDTCR, r16
  ret
```

C Code Example

```
void WDT_off(void)
{
  /* Write logical one to WDCE and WDE */
  WDTCR = (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCR = 0x00;
}
```

- **Sekwencje czasowe dla zmiany konfiguracji taktowania Watchdog'a**

Sekwencje zmiany konfiguracji różnią się nieznacznie pomiędzy trzema poziomami bezpieczeństwa. Osobne procedury zostały opisane dla poszczególnych poziomów.

- **Poziom bezpieczeństwa 0**

Ten tryb jest porównywalny z operacjami Watchdog'a znajdującymi się w Atemga103. Taktowanie Wathcdog'a jest początkowe zablokowane, ale może zostać odblokowane przez wpisanie jedynki do bitu WDE. Okres czasu przerwy może być zmieniany w każdym momencie bez ograniczeń. Aby zablokować taktowanie Watchdog'a, procedura opisana na stronie 51 (opis bitu WDE) musi być przestrzegana.

- **Poziom bezpieczeństwa 1**

W tym trybie taktowanie Watchdog'a jest początkowo zablokowane, ale może zostać odblokowane poprzez wpisanie jedynki do bitu WDE. Sekwencje czasowe są potrzebne podczas zmiany okresu czasu przerwy lub trzeba zablokować odblokowane taktowanie Watchdog'a. Aby zablokować odblokowane taktowanie Watchdog'a i/lub zmienić okres czasu przerwy należy wykonać następującą procedurę:

1. w tej samej operacji należy zapisań logiczną jedynkę do WDCE i WDE. Logiczna jedynka musi zostać zapisana do WDE niezależnie od wcześniejszej jego wartości.
2. w ciągu następnych czterech cykli maszynowych, w jednej operacji, należy zapisać WDE i WDP jak się chce ale należy wyzerować bit WDCE.

- **Poziom bezpieczeństwa 2**

W tym trybie taktowanie Wathcdog'a jest zawsze odblokowane i bit WDE jest zawsze odczytywany jako 1. Sekwencje czasowe sa potrzebne w celu zmienienia okresu czasu przerwy. Aby to zmienić należy postępować zgodnie z procedura:

1. W jednej operacji należy zapisać logiczną jedynkę do WDCE i WDE. Niezależnie od tego, że WDE zawsze jest ustawione należy ustawiać je od nowa.
2. W przeciągu następnych czterech cykli zegarowych, w jednej operacji, należy zapisać bit WDP jak się chce ale bit WDCE musi zostać wyzerowany. Wartość wpisana do bitu WDE jest nieistotna.

PRZERWANIA

Sekcja ta opisuje specyfikę obsługi przerwań jakie są obsługiwane w ATmega128. Ogólny opis jest w sekcji „Obsługa resetu i przerwań" na stronie 13.

- **Wektory przerwań w ATmega128**

Nr wektora	Adres programu(2)	Źródło	Definicja przerwania
1	$0000(1)	RESET	Wszystkie resety
2	$0002	INT0	Zewnętrzne żądanie przerwania 0
3	$0004	INT1	Zewnętrzne żądanie przerwania 1
4	$0006	INT2	Zewnętrzne żądanie przerwania 2
5	$0008	INT3	Zewnętrzne żądanie przerwania 3
6	$000A	INT4	Zewnętrzne żądanie przerwania 4
7	$000C	INT5	Zewnętrzne żądanie przerwania 5
8	$000E	INT6	Zewnętrzne żądanie przerwania 6
9	$0010	INT7	Zewnętrzne żądanie przerwania 7
10	$0012	TIMER2 COMP	Licznik-stoper2 porównanie
11	$0014	TIMER2 OVF	Licznik-stoper2 przepełnienie
12	$0016	TIMER1 CAPT	Licznik-stoper1 przechwyt zdarzenia
13	$0018	TIMER1 COMPA	Licznik-stoper1 porównanie A
14	$001A	TIMER1 COMPB	Licznik-stoper1 porównanie B
15	$001C	TIMER1 OVF	Licznik-stoper1 przepełnienie
16	$001E	TIMER0 COMP	Licznik-stoper0 porównanie
17	$0020	TIMER0 OVF	Licznik-stoper0 przepełnienie
18	$0022	SPI, STC	SPI ukończenie transmisji szeregowej
19	$0024	USART0, RX	USART0 zakończenie RX
20	$0026	USART0 UDRE	USART0 pusty rejestr danych
21	$0028	USART0 TX	USART0 zakończenie TX
22	$002A	ADC	zakończenie konwersji ADC
23	$002C	EE READY	EEPROM gotowe
24	$002E	ANALOG CMP	Analogowy komparator
25	$0030(3)	TIMER1 COMPC	Licznik-stoper1 porównanie C
26	$0032(3)	TIMER3 CAPT	Licznik-stoper3 przechwyt zdarzenia
27	$0034(3)	TIMER3 COMPA	Licznik-stoper3 porównanie A
28	$0036(3)	TIMER3 COMPB	Licznik-stoper3 porównanie B
29	$0038(3)	TIMER3 COMPC	Licznik-stoper3 porównanie C
30	$003A(3)	TIMER3 OVR	Licznik-stoper3 przepełnieni
31	$003C(3)	USART1 RX	USART1 zakończenie RX
32	$003E(3)	USART1 UDRE	USART1 pusty rejestr danych
33	$0040(3)	USART1 TX	USART1 zakończenie TX
34	$0042(3)	TWI	Interfejs dwuprzewodowy
35	$0044(3)	SPM READY	Pamięć przechowująca program gotowa

Uwaga: 1. Jeżeli bezpiecznik BOOTRST jest zaprogramowany, urządzenie przeskoczy do adresu ładowania programu przy resecie – „Wsparcie ładowania programu...” na stronie 269.
2. Kiedy bit IVSEL w MCUCR jest ustawiony, wektory przerwań zostaną przesunięte na początek sekcji ładowanej Flash. Adres każdego przerwania będzie dodany do początkowego adresu sekcji ładowanej we Flash.
3. Przerwania w adresach $0030 - $0044 nie istnieją w trybie kompatybilności z Atmegoa103.

Tabela 24 przedstawia umiejscowienie wektorów przerwań i resetu dla różnych kombinacji ustawień BOOTRST i IVSEL. Jeżeli program nigdy nie odblokuje źródeł przerwań, wektory te nie są wykorzystywane i kod programu może zostać umieszczony w tych adresach. Również kod może zająć te lokacje, kiedy wektory przerwań resetu są w części pamięci dla aplikacji a wektory przerwań w części ładowalnej i vice versa.

BOOTRST	IVSEL	Adres resetu	Adres początkowy wektorów przerwań
1	0	$0000	$0002
1	1	$0000	Adres ładowania resetu + $0002
0	0	Adres ładowania resetu	$0002
0	1	Adres ładowania resetu	Adres ładowania resetu + $0002

Uwaga: Adres ładowania resetu jest przedstawiony w tabeli 112 na stronie 280. Dla bezpiecznika BOOTRST jedynka oznacza nie zaprogramowanie, natomiast dla zera jest on zaprogramowany.

Najczęściej spotykana struktura adresów programu dla resetu i wektorów przerwań w ATmega128:

```
Address   Labels Code             Comments
$0000           jmp    RESET      ; Reset Handler
$0002           jmp    EXT_INT0   ; IRQ0 Handler
$0004           jmp    EXT_INT1   ; IRQ1 Handler
$0006           jmp    EXT_INT2   ; IRQ2 Handler
$0008           jmp    EXT_INT3   ; IRQ3 Handler
$000A           jmp    EXT_INT4   ; IRQ4 Handler
$000C           jmp    EXT_INT5   ; IRQ5 Handler
$000E           jmp    EXT_INT6   ; IRQ6 Handler
$0010           jmp    EXT_INT7   ; IRQ7 Handler
$0012           jmp    TIM2_COMP  ; Timer2 Compare Handler
$0014           jmp    TIM2_OVF   ; Timer2 Overflow Handler
$0016           jmp    TIM1_CAPT  ; Timer1 Capture Handler
$0018           jmp    TIM1_COMPA ; Timer1 CompareA Handler
$001A           jmp    TIM1_COMPB ; Timer1 CompareB Handler
$001C           jmp    TIM1_OVF   ; Timer1 Overflow Handler
$001E           jmp    TIM0_COMP  ; Timer0 Compare Handler
$0020           jmp    TIM0_OVF   ; Timer0 Overflow Handler
$0022           jmp    SPI_STC    ; SPI Transfer Complete Handler
$0024           jmp    USART0_RXC ; USART0 RX Complete Handler
$0026           jmp    USART0_DRE ; USART0,UDR Empty Handler
$0028           jmp    USART0_TXC ; USART0 TX Complete Handler
```

```
$002A          jmp     ADC         ; ADC Conversion Complete Handler
$002C          jmp     EE_RDY      ; EEPROM Ready Handler
$002E          jmp     ANA_COMP    ; Analog Comparator Handler
$0030          jmp     TIM1_COMPC  ; Timer1 CompareC Handler
$0032          jmp     TIM3_CAPT   ; Timer3 Capture Handler
$0034          jmp     TIM3_COMPA  ; Timer3 CompareA Handler
$0036          jmp     TIM3_COMPB  ; Timer3 CompareB Handler
$0038          jmp     TIM3_COMPC  ; Timer3 CompareC Handler
$003A          jmp     TIM3_OVF    ; Timer3 Overflow Handler
$003C          jmp     USART1_RXC  ; USART1 RX Complete Handler
$003E          jmp     USART1_DRE; USART1,UDR Empty Handler
$0040          jmp     USART1_TXC  ; USART1 TX Complete Handler
$0042          jmp     TWI         ; Two-wire Serial Interface Interrupt
Handler
$0044          jmp     SPM_RDY     ; SPM Ready Handler
;
$0046     RESET:ldir16, high(RAMEND); Main program start
$0047          out     SPH,r16     ; Set stack pointer to top of RAM
$0048          ldi     r16, low(RAMEND)
$0049          out     SPL,r16
$004A          sei                 ; Enable interrupts
$004B          <instr>  xxx
  ...      ...    ...    ...
```

Kiedy bezpiecznik BOOTRST nie jest zaprogramowany, sekcja ładowania programów jest ustawiona na 8KB i bit IVSEL w rejestrze MCUCR jest ustawiony przed odblokoaniem przerwań. Najczęściej spotykana struktura adresów programu dla resetu i wektorów przerwań są:

```
Address   LabelsCode              Comments
$0000     RESET:ldi     r16,high(RAMEND) ; Main program start
$0001           out     SPH,r16     ; Set stack pointer to top of RAM
$0002           ldi     r16,low(RAMEND)
$0003           out     SPL,r16
$0004           sei                 ; Enable interrupts
$0005           <instr>  xxx
;
.org $F002
$F002           jmp     EXT_INT0    ; IRQ0 Handler
$F004           jmp     EXT_INT1    ; IRQ1 Handler
...             ...     ...         ;
$F044           jmp     SPM_RDY     ; Store Program Memory Ready Handler
```

Kiedy bezpiecznik BOOTRST jest zaprogramowany, sekcja ładowania programów jest ustawiona na 8KB to najczęściej spotykana struktura adresów programu dla resetu i wektorów przerwań są:

```
Address     LabelsCode            Comments
.org $0002
$0002          jmp     EXT_INT0    ; IRQ0 Handler
$0004            jmp     EXT_INT1    ; IRQ1 Handler
...              ...     ...         ;
$0044            jmp     SPM_RDY     ; Store Program Memory Ready Handler
;
.org $F000
$F000     RESET: ldi     r16,high(RAMEND) ; Main program start
$F001            out     SPH,r16     ; Set stack pointer to top of RAM
$F002            ldi     r16,low(RAMEND)
$F003            out     SPL,r16
$F004            sei                 ; Enable interrupts
$F005            <instr>  xxx
```

Kiedy bezpiecznik BOOTRST jest zaprogramowany, sekcja ładowania programów jest ustawiona na 8KB i bit IVSEL w rejestrze MCUCR jest ustawiony przed odblokoaniem przerwań. Najczęściej spotykana struktura adresów programu dla resetu i wektorów przerwań są:

```
Address              Labels     Code              Comments
;
.org $F000
$F000            jmp     RESET       ; Reset handler
$F002            jmp     EXT_INT0    ; IRQ0 Handler
$F004            jmp     EXT_INT1    ; IRQ1 Handler
...              ...     ...         ;
$F044            jmp     SPM_RDY   ; Store Program Memory Ready Handler
$F046     RESET: ldi     r16,high(RAMEND) ; Main program start
$F047            out     SPH,r16     ; Set stack pointer to top of RAM
$F048            ldi     r16,low(RAMEND)
$F049            out     SPL,r16
$F04A            sei                 ; Enable interrupts
$F04B            <instr>  xxx
```

- **Przenoszenie przerwań miedzy programami i przestrzenią boot**

Główny rejestr kontroli kontroluje umieszczenie tablicy wektorów przerwań.

- **Rejestr kontroli MCU – MCUCR**

Bit	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUCR
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	

a: bit 1 – IVCE: wybór wektora przerwania

Kiedy bit ten jest wyzerowany, wektor przerwań umieszczony jest na początku pamięci Flash. Kiedy bit ten jest ustawiony, wektory przerwań zostają przeniesione na początek sekcji ładowania pamięci Flasz. Adres rozpoczynający ta sekcję jest zależny od bezpiecznika BOOTRST. Odnieś się do sekcji „Wsparcie programu inicjującego...”na stronie 269. W celu uniknięcia niezamierzonych zmian tablicy wektorów przerwań specjalna procedura zapisu musi być przestrzegana do zmiany ustawienia bitu IVSEL:

1. Zapisz bit odblokowujący zmianę wektora przerwań (IVCE) na jeden.
2. Podczas czterech cykli maszynowych zapisz żądaną wartość do IVSEL podczas zapisu zera do IVCE

Przerwania automatycznie zostaną zablokowane podczas wykonywania tej sekwencji. Przerwania są blokowane w cyklach gdzie IVCE jest ustawione i pozostają zablokowane dopóki nie zostanie wykonana instrukcja zapisu do IVSEL. Jeżeli IVSEL nie jest zapisane, przerwania zostają zablokowane na cztery cykle maszynowe. Automatyczne blokowanie przerwań nie ma wpływu na bit I w rejestrze statusu.

Uwaga: Jeżeli wektory przerwań są umieszczone w części inicjującej w pamięci Flash i bit odblokowujący ładowanie BLB02 jest zaprogramowany, przerwania są blokowane podczas wykonywania programu z części programowej pamięci. Jeżeli wektory przerwań są umieszczone w części programowej pamięci i bit BLB12 jest zaprogramowany przerwania są zablokowane podczas wykonywania programu z sekcji inicjującej pamięci. Zobacz w sekcji „Wsparcie programu inicjującego...” a stronie 269.

b: bit 0- IVCE: bit odblokowujący możliwość zmiany wektora przerwań

Bit ten musi być ustawiony na logiczną jedynkę, aby odblokować zmianę bitu IVSEL. IVCE jest zerowane przez sprzęt cztery cykle maszynowe po tym jak został zapisany, lub jak IVSEL został zapisane. Ustawienie bitu IVCE zablokuje przerwania co zostało wyjaśnione powyżej w opisie IVSEL. Przykłady kodu:

Assembly Code Example

```
Move_interrupts:
  ; Enable change of interrupt vectors
  ldi  r16, (1<<IVCE)
  out  MCUCR, r16
  ; Move interrupts to boot flash section
  ldi  r16, (1<<IVSEL)
  out  MCUCR, r16
  ret
```

C Code Example

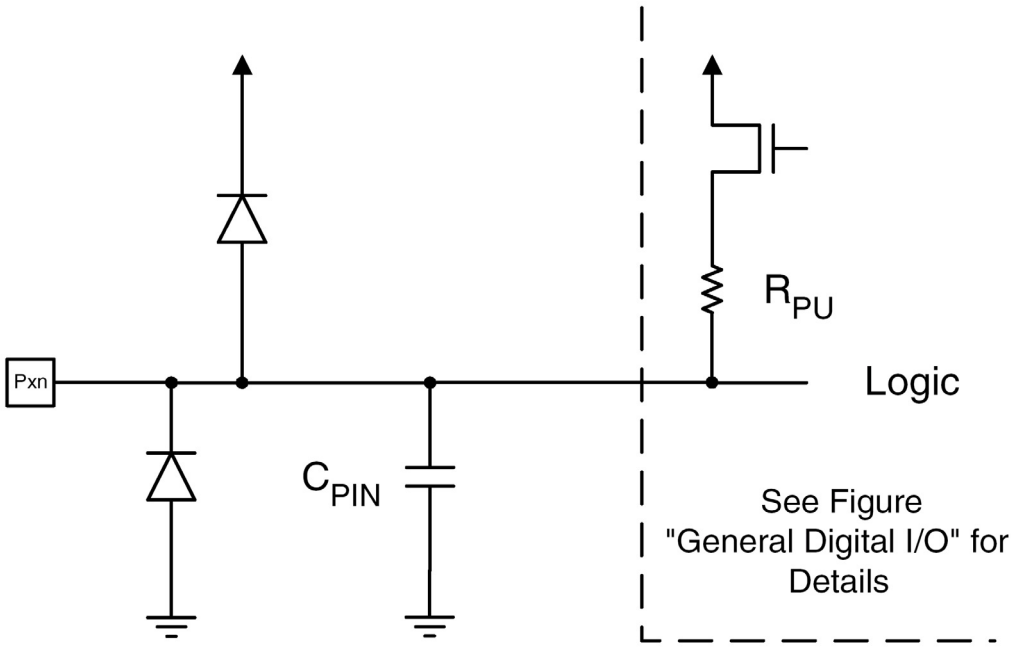
```
void Move_interrupts(void)
{
  /* Enable change of interrupt vectors */
  MCUCR = (1<<IVCE);
  /* Move interrupts to boot flash section */
  MCUCR = (1<<IVSEL);
}
```

PORTY WEJŚCIA WYJŚCIA

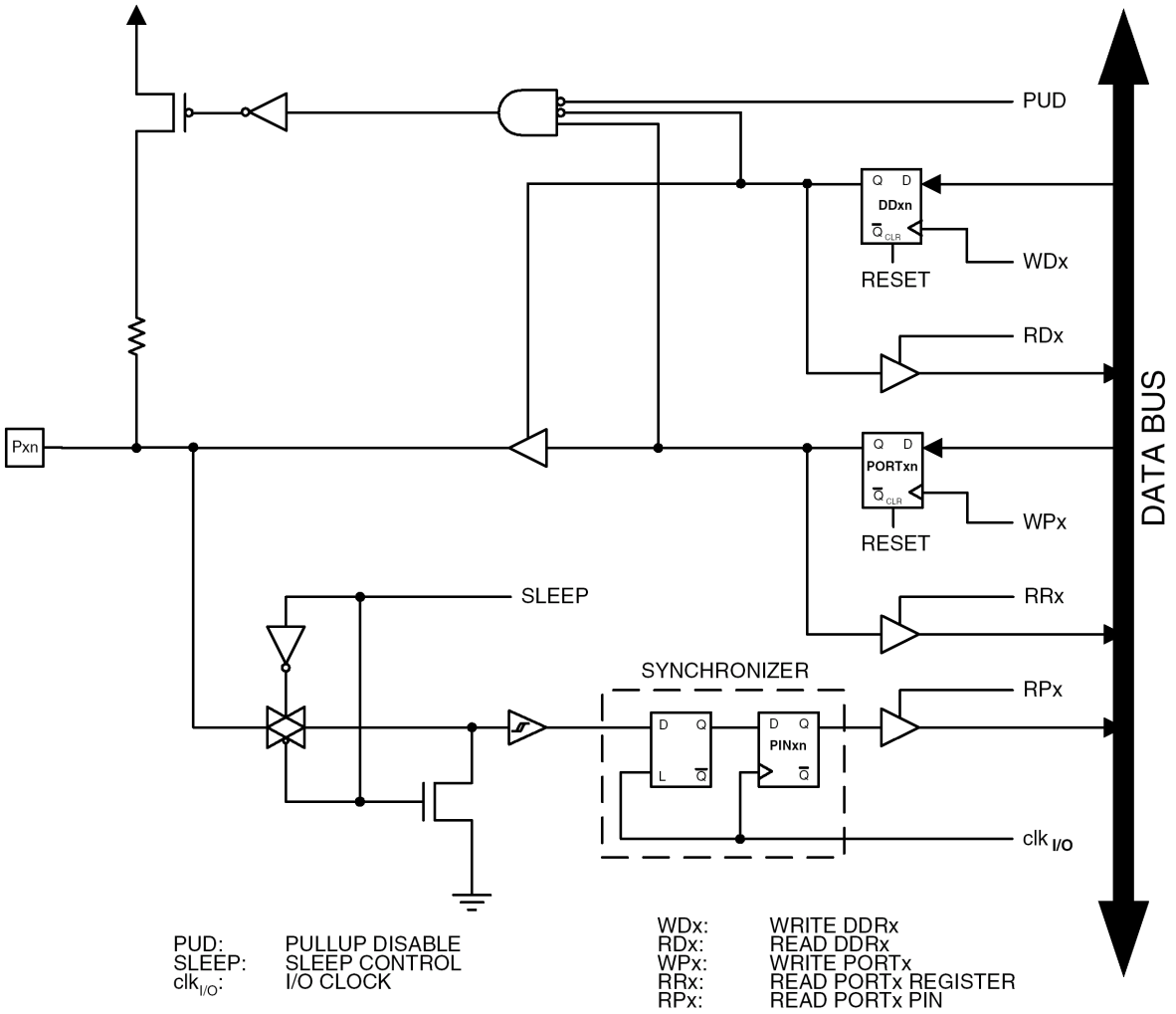
WPROWADZENIE

Wszystkie porty są portami dwukierunkowymi jeśli są używane jako cyfrowe porty we/wy ogólnego przeznaczenia. Oznacza to, że kierunkowość jakiegokolwiek wyprowadzenia może być zmieniana bez niezamierzonej zmiany kierunkowości któregokolwiek z pozostałych wyprowadzeń. Ta sama sytuacja ma miejsce w przypadku włączania/wyłączania rezystorów podciągających: mogą być włączane/wyłączane dla każdego wyprowadzenia niezależnie od innych wyprowadzeń. Obciążalność portów we/wy jest wystarczająca do podłączenia wprost wyświetlacza LED. Wszystkie wyprowadzenia mają niezależne rezystory podciągające oraz posiadają diody zabezpieczające (rys. 28.)

Rys. 28.



Rys 29. Schemat funkcjonalny wyprowadzenia portu we/wy



KONFIGURACJA WYPROWADZEŃ

Konfigurację każdego wyprowadzenia opisują trzy bity w odpowiednich rejestrach:

- DDxn (bit rejestru DDRx) ustawia kierunkowość portu. Jeśli wpisana jest wartość 1 wyprowadzenie Pxn pracuje jako wyjście, jeśli jest wpisane 0 pracuje jako wejście.
- PORTxn (bit rejestru PORTx) jeśli jest wpisana jedynka i pin jest skonfigurowany jako wejście : rezystor podciągający jest używany. Aby wyłączyć rezystor podciągający należy do tego bitu wpisać wartość zero lub skonfigurować wyprowadzenie jako wyjście. Jeśli bit PORTxn ma wartość 1 przy konfiguracji jako wyjście to na wyprowadzeniu jest stan wysoki. Jeśli POTRx = 0 i wyprowadzenie pełni funkcję wyjścia to na tym wyjściu jest stan niski.

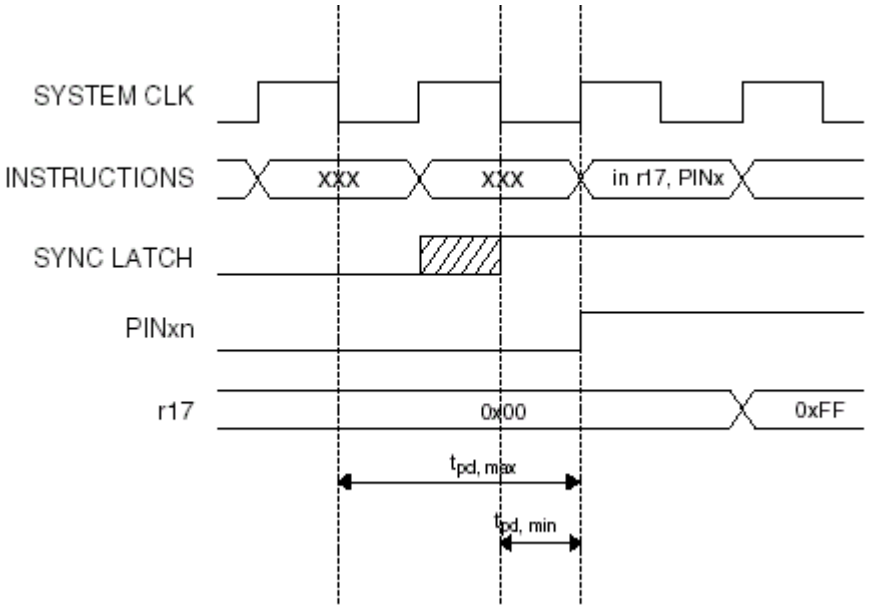
Przy przełączaniu między trzecim stanem ({DDxn, PORTxn} = 0b00) i stanem wysokim wyjścia musi wystąpić stan pośredni (z włączonym którymś z rezystorów podciągających) ({DDxn, PORTxn} = 0b01) lub stan niski wyjścia ({DDxn, PORTxn} = 0b10).
Przełączanie między wejściem z włączonymi rezystorami podciągającymi a wyjściem w stanie niskim również powoduje wystąpienie tego samego problemu. Jako stan pośredni musi wystąpić trzeci stan({DDxn, PORTxn} = 0b00) lub wysoki stan wyjścia
W celu wyłączenia rezystorów podciągających we wszystkich portach należy ustawić wartość bitu PUD (rejestr SFIOR) na 1.
Tabela 25. Konfiguracja wyprowadzeń portów.

DDxn	POTRxn	PUD	We/wy	Włączone rezystory podciągające
0	0	---	We	Nie
0	1	0	We	Tak
0	1	1	We	Nie
1	0	---	Wy	Nie
1	1	---	Wy	Nie

ODCZYT STANU WYPROWADZEŃ

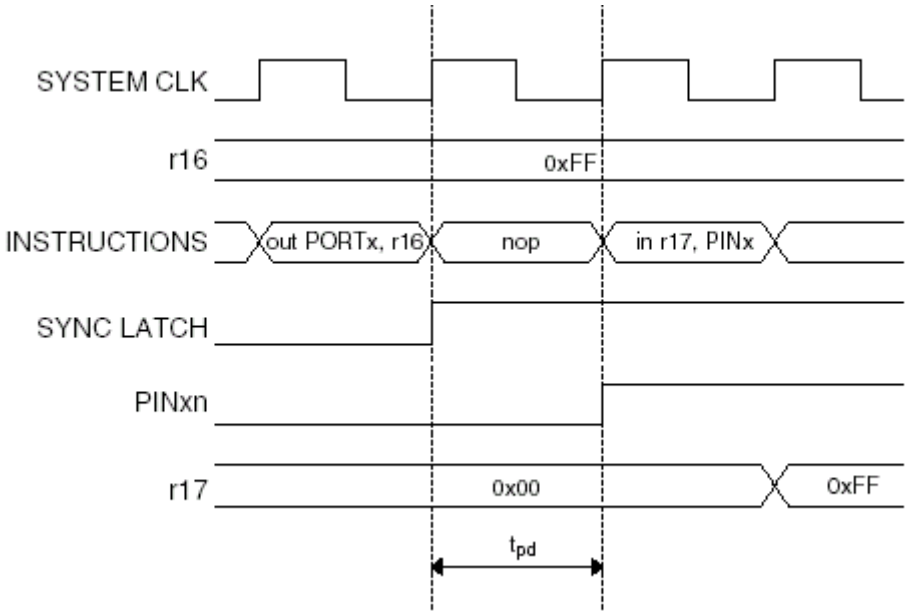
Niezależnie od ustawienia bitu DDxn, stan wyprowadzeń portu może być odczytany przez bity rejestru PINxn.

Rys. 30. Przebieg czasowy obrazujący synchronizacje przy czytaniu wartości na wyprowadzeniu. Maksymalne i minimalne czasy propagacji są oznaczone jako $t_{pd,max}$ i $t_{pd,min}$.



Cykl zegara rozpoczyna się krótko po pierwszym opadającym zboczu sygnału zegara systemowego. Przejście wyprowadzenia portu w nowy stan jest opóźnione od ½ do 1½ okresu zegara systemowego.
Podczas odczytu programowo ustawionej wartości wyprowadzenia w odpowiednim czasie należy wykonać instrukcję nop jak zaznaczono na rys. 31. Instrukcja out ustawia sygnał "SYNC LATCH" na 1 przy narastającym zboczu sygnału zegarowego. W tym przypadku czas opóźnienia t_{pd} potrzebny na synchronizację wynosi jeden okres zegara.

Rys 31.



Poniższy fragment kodu pokazuje jak ustawić wyprowadzenia

- 0 i 1 portu B na 1
- 4 do 7 jako wejście przy czym wyprowadzenia 6 i 7 mają włączone rezystory podciągające.

Kod assemblera:

```
..
ldi r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out PORTB,r16
out DDRB,r17
nop
in r16,PINB
...
```

Kod w języku C:

```
unsigned char i;
...
```

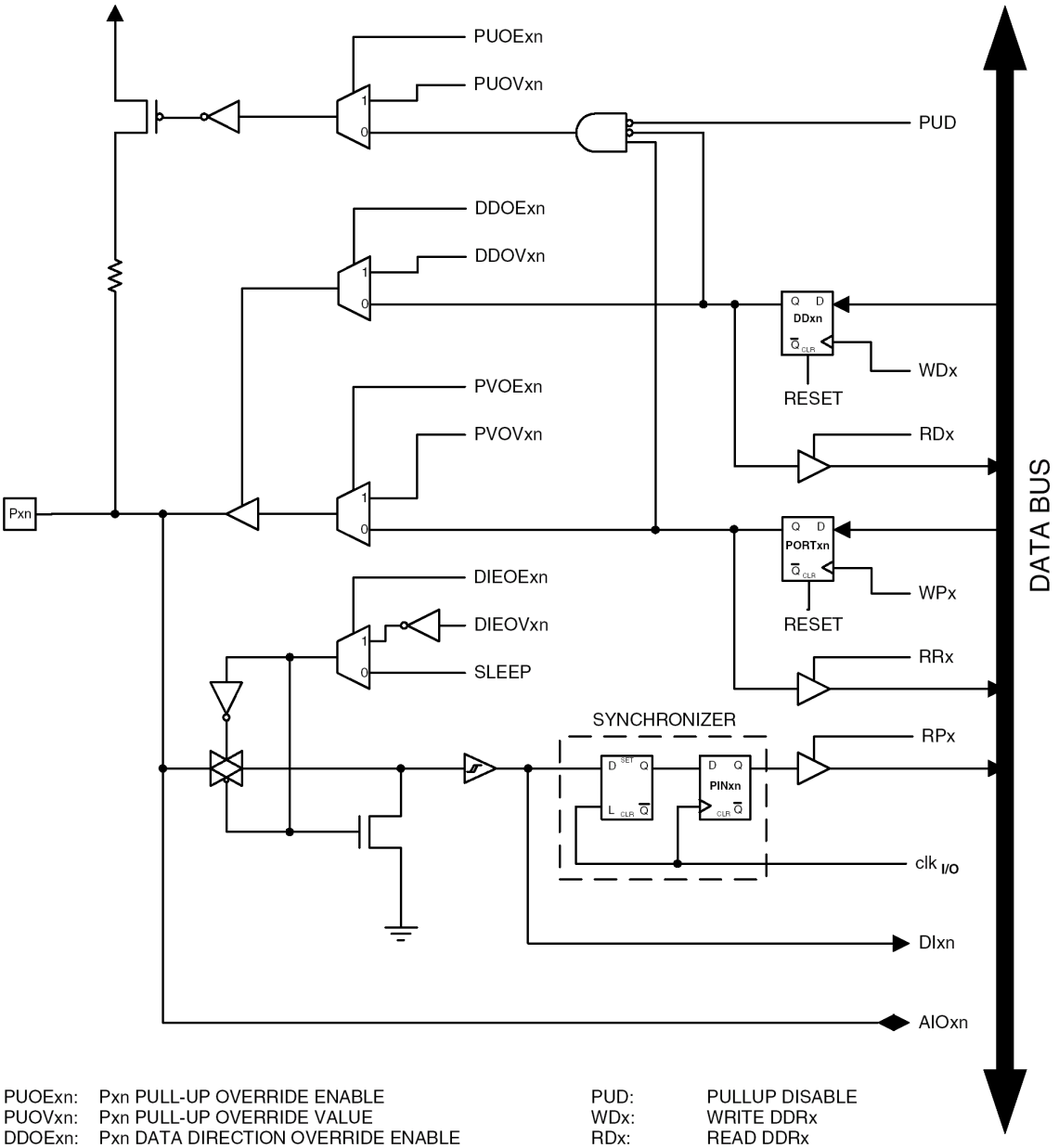
```
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
_NOP();
i = PINB;
...
```

Jak zaznaczono na rys. 29. sygnał cyfrowego wejścia może być zwarty do masy na wejściu przerzutnika Schmitta. Sygnał oznaczony jako SLEEP jest generowany przez MCU Sleep Controller w stanie wyłączenia(Power-down mode), w stanie ograniczonego poboru mocy(" Power-save mode") lub w stanie bezczynności ("Standby mode", „Extended Standby mode") w celu ograniczenia zużycia energii.
Sygnał SLEEP jest nieaktywny dla wyprowadzeń ustawionych jako „External Interrupt pins". SLEEP jest również „unieważniany" jeśli wyprowadzenie pełni alternatywną funkcję.przez

ALTERNATYWNE FUNKCJE WYPROWADZEŃ PORTÓW WE/WY

Większość wyprowadzeń może pełnić dodatkowe funkcje oprócz funkcji cyfrowego wejścia – wyjścia. Na rys.32. zaznaczono w jaki sposób sygnały sterowania wyprowadzeń nogą być znoszone przez dodatkowe funkcje. Sygnały dodatkowe mogą nie być dostępne dla wszystkich wyprowadzeń portów, jednak rysunek stanowi ogólny opis dodatkowych funkcji mikrokontrolerów AVR.

Rys. 32.



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULLUP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WPx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE	$clk_{I/O}$:	I/O CLOCK
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE	DIxn:	DIGITAL INPUT PIN n ON PORTx
SLEEP:	SLEEP CONTROL	AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

W tabeli 26 zostały przedstawione sygnały uaktywniające dodatkowe funkcje wyprowadzeń

Nazwa sygnału	Pełna nazwa sygnału	Opis
PUOE	Pull-up Override Enable	Jeśli sygnał jest aktywny, włączenie rezystorów podciągających jest kontrolowane przez sygnał PUOV. Jeśli sygnał jest nieaktywny Rezystory podciągające są włączone gdy {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	Przy włączonym PUOE rez. Podciągające są włączone(wyłączone) gdy PUOV jest aktywny(nieaktywny) bez względu na wartość bitów DDxn, PORTxn i PUD.
DDOE	Data Direction Override Enable	Jeśli DDOE jest aktywny włączenie wzmacniacza wyjścia jest zależne od sygnału DDOV. Jeśli DDOV jest nieaktywny wzmacniacz wyjścia jest wł/wył zależnie od wartości bitu DDxn.
DDOV	Data Direction Override Value	Przy aktywnym DDOE aktywny (nieaktywny) DDOV powoduje włączenie (wyłączenie) wzmacniacza wyjścia niezależnie od stanu bitu DDxn
PVOE	Port Value Override Enable	Jeśli ten sygnał jest aktywny i wzmacniacz wyjścia jest włączony, wartość portu jest kontrolowana przez sygnał PVOV. Jeśli PVOE jest nieaktywny a wzmacniacz wyjścia jest włączony wartość portu jest kontrolowana przez bit PORTxn
PVOV	Port Value Override Value	Jeśli PVOV jest aktywny wartość portu jest ustawiana tym sygnałem, niezależnie od wartości bitu PORTxn
DIEOE	Digital Input Enable Override Enable	Jeśli jest aktywny włączenie cyfrowego wejścia jest zależne od sygnału DIEOV. Jeśli DIEOE jest nieaktywny uaktywnienie wejścia cyfrowego jest zależne od stanu MCU (stan normalny, stany uśpienia(„sleep"))
DIEOV	Digital Input Enable Override Value	Jeśli DIEOE jest aktywny, wejście cyfrowe jest (włączone/wyłączone) jeśli DIEOV jest aktywny (nieaktywny) niezależnie od stanu MCU
DI	Digital Input	Na rysunku (rys 32) sygnał jest połączony z wyjściem przerzutnika Schmitta przed synchronizatorem. Niezależnie od wykorzystania wejścia cyfrowego jako źródła sygnału zegarowego moduł z funkcją alternatywną używa własnego synchronizatora.
AIO	Analog Input/output	Sygnał wejścia/wyjścia do/z dla dodatkowych funkcji.

DODATKOWE FUNKCJE PORTU A

Port A może być wykorzystany do adresowania linii danych lub adresowych interfejsu pamięci zewnętrznej.

Tabela 27.

Wyprowadzenie	Funkcja dodatkowa
PA7	interfejs pamięci zewnętrznej: 7 bit adresu lub danych

PA6	interfejs pamięci zewnętrznej: 6 bit adresu lub danych
PA5	interfejs pamięci zewnętrznej: 5 bit adresu lub danych
PA4	interfejs pamięci zewnętrznej: 4 bit adresu lub danych
PA3	interfejs pamięci zewnętrznej: 3 bit adresu lub danych
PA2	interfejs pamięci zewnętrznej: 2 bit adresu lub danych
PA1	interfejs pamięci zewnętrznej: 1 bit adresu lub danych
PA0	interfejs pamięci zewnętrznej: 0 bit adresu lub danych

Tabele 28 i 29 zawierają zależność między sygnałami a alternatywnymi funkcjami portu A.

Tabela 28. Alternatywne funkcje pinów PA7...PA4

Nazwa sygnału	PA7/AD7	PA6/AD6	PA5/AD5	PA4/AD4
PUOE	SRE	SRE	SRE	SRE
PUOV	~($\overline{WR}$ | ADA[1]) • PORTA7 • $\overline{PUD}$	~($\overline{WR}$ | ADA) • PORTA6 • $\overline{PUD}$	~($\overline{WR}$ | ADA) • PORTA5 • $\overline{PUD}$	~($\overline{WR}$ | ADA) • PORTA4 • $\overline{PUD}$
DDOE	SRE	SRE	SRE	SRE
DDOV	$\overline{WR}$ | ADA	$\overline{WR}$ | ADA	$\overline{WR}$ | ADA	$\overline{WR}$ | ADA
PVOE	SRE	SRE	SRE	SRE
PVOV	A7 • ADA | D7 OUTPUT • $\overline{WR}$	A6 • ADA | D6 OUTPUT • $\overline{WR}$	A5 • ADA | D5 OUTPUT • $\overline{WR}$	A4 • ADA | D4 OUTPUT • $\overline{WR}$
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	D7 INPUT	D6 INPUT	D5 INPUT	D4 INPUT
AIO	–	–	–	–

Tabela 29. Alternatywne funkcje pinów PA3...PA0

Nazwa sygnału	PA3/AD3	PA2/AD2	PA1/AD1	PA0/AD0
PUOE	SRE	SRE	SRE	SRE
PUOV	~(WR | ADA) • PORTA3 • PUD	~(WR | ADA) • PORTA2 • PUD	~(WR | ADA) • PORTA1 • PUD	~(WR | ADA) • PORTA0 • PUD
DDOE	SRE	SRE	SRE	SRE
DDOV	WR | ADA	WR | ADA	WR | ADA	WR | ADA
PVOE	SRE	SRE	SRE	SRE
PVOV	A3 • ADA | D3 OUTPUT • WR	A2• ADA | D2 OUTPUT • WR	A1 • ADA | D1 OUTPUT • WR	A0 • ADA | D0 OUTPUT • WR
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	D3 INPUT	D2 INPUT	D1 INPUT	D0 INPUT
AIO	–	–	–	–

DODATKOWE FUNKCJE PORTU B

Tabela 30

Wyprowadzenie	Funkcja dodatkowa
PB7	OC2/OC1C
PB6	OC1B
PB5	OC1A
PB4	OC0
PB3	MISO
PB2	MOSI
PB1	SCK
PB0	$\overline{SS}$

OC2/OC1C (bit 7)

OC2 :(Output Compare Match output) Wyprowadzenie PB7 można wykorzystać jako zewnętrzne wyjście dla licznika2 modulatora OCM1C2 **(**Timer/Counter2 Output Compare).
.W tym wypadku wyprowadzenie musi być skonfigurowane jako wyjście (bit DDB7 = =1).Wyprowadzenie OC2 może być również wyjściem dla licznika PWM.

OC1C (Output Compare Match C output) Wyprowadzenie PB7 można wykorzystać jako zewnętrzne wyjście dla licznika1 modulatora OCM1C2 (Timer/Counter1 Output Compare C).
. W tym wypadku wyprowadzenie musi być skonfigurowane jako wyjście (bit DDB7 = =1).Wyprowadzenie OC2 może być również wyjściem dla licznika PWM.

OC1B (bit 6)
OC1B (Output Compare Match B output) wyprowadzenie PB6 może być użyte jako zewnętrzne wyjście licznika1 OCM1C2 (Timer/Counter1 Output Compare B).Aby pełnić tą funkcje wyprowadzenie musi być skonfigurowane jako wyjście(DDB6 = 1)

OC1A (bit 5)
OC1A (Output Compare Match A output) wyprowadzenie PB65 może być użyte jako zewnętrzne wyjście licznika1 OCM1C2 (Timer/Counter1 Output Compare A).Aby pełnić tą funkcje wyprowadzenie musi być skonfigurowane jako wyjście(DDB5 = 1)

OC0 (bit 4)
OC0 (Output Compare Match output) pin PB4 może być wykorzystany jako wyjście dla licznika 0 OCM1C2 (Timer/Counter0 Output Compare), aby pełnić tą funkcję wyprowadzenie musi być skonfigurowane jako wejście.

MISO (bit 3)
MISO: (Master Data Input, Slave Data Output). Jeśli kanał SPI jest ustawiony jako master to wyprowadzenie PB3 pracuje jako wejście niezależnie od ustawienia bitu DDB3. Jeśli SPI jest ustawiony jako slave kierunkowość tego wyprowadzenia zależy od wartości wpisanej do bitu DDB3. W przypadku, gdy pin pracuje jako wejście włączenie rezystorów podciągających jest zależne od wartości bitu PORTB3.

MOSI (bit 2)
MOSI (SPI Master Data Output, Slave Data Input) Jeśli SPI jest ustawiony na slave to wyprowadzenie pracuje jako wejście niezależnie od wartości bitu DDB2. Jeśli kanał SPI pracuje jako master kierunkowość tego wyprowadzenia jest ustawiana bitem DDB2. Jeśli wyprowadzenie jest ustawione jako wejście aktywacja rezystorów podciągających zależy od wartości bitu PORTB2.

SCK (bit 1)
SCK (Master Clock output, Slave Clock input). W przypadku, gdy SPI jest ustawiony na slave to wyprowadzenie pracuje jako wejście niezależnie od wartości bitu DDB1. Jeśli kanał SPI pracuje jako master kierunkowość tego wyprowadzenia jest ustawiana bitem DDB1. Jeśli wyprowadzenie jest ustawione jako wejście aktywacja rezystorów podciągających zależy od wartości bitu PORTB1.

SS (bit 0)
SS (Slave Port Select Input)) Jeśli SPI jest ustawiony na slave to wyprowadzenie pracuje jako wejście niezależnie od wartości bitu DDB0. Spi jest ustawiony na slawe jeśli na tym wyprowadzeniu jest wartość niska. Jeśli kanał SPI pracuje jako master kierunkowość tego wyprowadzenia jest ustawiana bitem DDB0. Jeśli wyprowadzenie jest ustawione jako wejście aktywacja rezystorów podciągających zależy od wartości bitu PORTB1.

Tabela 31. Alternatywne funkcje pinów PB7...PB4

Nazwa sygnału	PB7/OC2/OC1C	PB6/OC1B	PB5/OC1A	PB4/OC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2/OC1C ENABLE(1)	OC1B ENABLE	OC1A ENABLE	OC0 ENABLE
PVOV	OC2/OC1C(1)	OC1B	OC1A	OC0B
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	–	–	–	–

Tabela 32. Alternatywne funkcje wyprowadzeń PB3...PB0

Nazwa sygnału	PB3/MISO	PB2/MOSI	PB1/SCK	PB0/$\overline{SS}$
PUOE	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$
PUOV	PORTB3 • $\overline{PUD}$	PORTB2 • $\overline{PUD}$	PORTB1 • $\overline{PUD}$	PORTB0 • $\overline{PUD}$
DDOE	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$
DDOV	0	0	0	0
PVOE	SPE • $\overline{MSTR}$	SPE • MSTR	SPE • MSTR	0
PVOV	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	SCK OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SPI MSTR INPUT	SPI SLAVE INPUT	SCK INPUT	SPI $\overline{SS}$
AIO	–	–	–	–

DODATKOWE FUNKCJE PORTU C

W trybie zgodności z ATmega 103 Port C może być używany tylko jako wyjście. Port C może pełnić dodatkową funkcję adresowania starszego bajtu interfejsu pamięci zewnętrznej.

Tabela 33. Alternatywne funkcje portu C.

Wyprowadzenie	Funkcja dodatkowa
PC7	A15
PC6	A14
PC5	A13
PC4	A12
PC3	A11
PC2	A10
PC1	A9
PC0	A8

Tabela 34. Alternatywne funkcje portu C.

Nazwa sygnału	PC7/A15	PC6/A14	PC5/A13	PC4/A12
PUOE	SRE • (XMM[(1)]<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
PUOV	0	0	0	0
DDOE	SRE • (XMM<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
DDOV	1	1	1	1
PVOE	SRE • (XMM<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
PVOV	A11	A10	A9	A8
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	–	–	–	–

Tabela 35. Alternatywne funkcje portu C.

Nazwa sygnału	PC3/A11	PC2/A10	PC1/A9	PC0/A8
PUOE	SRE • (XMM<5)	SRE • (XMM<6)	SRE • (XMM<7)	SRE • (XMM<7)
PUOV	0	0	0	0
DDOE	SRE • (XMM<5)	SRE • (XMM<6)	SRE • (XMM<7)	SRE • (XMM<7)
DDOV	1	1	1	1
PVOE	SRE • (XMM<5)	SRE • (XMM<6)	SRE • (XMM<7)	SRE • (XMM<7)
PVOV	A11	A10	A9	A8
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	–	–	–	–

DODATKOWE FUNKCJE PORTU D

T2 (bit7) wejście licznika 2

T2 (bit6) wejście licznika 1

XCK1 (bit 5) (USART1 External clock) Wartość bitu DDD4 decyduje, czy wyprowadzenie jest wejściem (DDD4 = 1) czy wyjściem(DDD4 = 0). Pin XCK1 jest aktywny jedynie gdy USART1 pracuje synchronicznie.

IC1 (bit 4) może działać jako wejście

INT3/TXD1 (bit 3)
INT3 Wyprowadzenie PD3 może działać jako wejście przerwań zewnętrznych dla MCU.
TXD1 Wyjście danych dla USART1: jeśli transmiter USART jest włączony wyprowadzenie PD3 jest skonfigurowane jako wyjście niezależnie od wartości bitu DDD3

INT2/RXD1 (bit 2)
INT2 Pin PD2 może pracować jako wejście przerwań zewnętrznych dla MCU
TXD1 Wyjście danych dla USART1: kiedy odbiornik USART1 jest aktywny wyprowadzenie PD2 jest skonfigurowane jako wejście niezależnie od wartości bitu DDD2, włączenie rezystorów podciągających jest nadal kontrolowane przez bit PORTD2

INT1/SDA (bit 1)
INT1 Wyprowadzenie PD1 może działać jako wejście przerwań zewnętrznych dla MCU.
SDA (Two-wire Serial Interface Data) Kiedy dwuprzewodowy szeregowy interfejs jest włączony, bit TWEN w rejestrze TWCR =1 co powoduje, że wyprowadzenie PD1 przestaje pełnić rolę wyprowadzenia portu D i staje się wyprowadzeniem danych dwuprzewodowego interfejsu szeregowego. W tym trybie działa filtr impulsów tłumiący impulsy krótsze od 50 ns.

INT0/SCL (bit 0)
INT0 Wyprowadzenie PD0 może działać jako wejście przerwań zewnętrznych dla MCU.

SCL (Two-wire Serial Interface Clock:). Jeśli w rejestrze TWCR bit TWEN ma wartość 1 (dwuprzewodowy szeregowy interfejs jest włączony) Pin PD0 przestaje być wyprowadzeniem portu D i pełni funkcje zegara dla dwuprzewodowego interfejsu szeregowego W tym trybie działa filtr impulsów tłumiący impulsy krótsze od 50 ns.

Tabela 37 i Tabela 38 Alternatywne funkcje portu D

Tabela 37.Dodatkowe funkcje wyprowadzeń PD7...PD4

Nazwa sygnału	PD7/T2	PD6/T1	PD5/XCK1	PD4/IC1
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	UMSEL1	0
PVOV	0	0	XCK1 OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	T2 INPUT	T1 INPUT	XCK1 INPUT	IC1 INPUT
AIO	–	–	–	–

Tabela 38. Dodatkowe funkcje wyprowadzeń PD3...PD0

Nazwa sygn.	PD3/INT3/TXD1	PD2/INT2/RXD1	PD1/INT1/SDA	PD0/INT0/SCL
PUOE	TXEN1	RXEN1	TWEN	TWEN
PUOV	0	PORTD2 • $\overline{PUD}$	PORTD1 • $\overline{PUD}$	PORTD0 • $\overline{PUD}$
DDOE	TXEN1	RXEN1	TWEN	TWEN
DDOV	1	0	SDA_OUT	SCL_OUT
PVOE	TXEN1	0	TWEN	TWEN
PVOV	TXD1	0	0	0
DIEOE	INT3 ENABLE	INT2 ENABLE	INT1 ENABLE	INT0 ENABLE
DIEOV	1	1	1	1
DI	INT3 INPUT	INT2 INPUT/RXD1	INT1 INPUT	INT0 INPUT
AIO	–	–	SDA INPUT	SCL INPUT

DODATKOWE FUNKCJE PORTU E

INT7/IC3 (bit 7)
INT7 : Wyprowadzenie PE7 może służyć jako wejście przerwania zewnętrznego
IC3 : Pin PE7 może być użyty jako wejście dla licznika3

INT6/T3 (bit 6)
INT6 : Wyprowadzenie PE6 może służyć jako wejście przerwania zewnętrznego
T3 : wejście dla licznika3

NT5/OC3C (bit 5)
INT5 : Wyprowadzenie PE5 może służyć jako wejście przerwania zewnętrznego
OC3C (Output Compare Match C output) wyprowadzenie PE5 może być użyte jako zewnętrzne wyjście licznika3 OCM1C2 (Timer/Counter3 Output Compare C).Aby pełnić tą funkcje wyprowadzenie musi być skonfigurowane jako wyjście(DDE5 = 1). Wyprowadzenie OC3C może także pełnić funkcję wyjścia dla licznika z PVM.

INT4/OC3B (bit 4)
INT4 : Wyprowadzenie PE4 może służyć jako wejście przerwania zewnętrznego
OC3B (Output Compare Match B output) wyprowadzenie PE4 może być użyte jako wyjście zewnętrzne licznika3 OCM1C2 (Timer/Counter3 Output Compare B).Aby pełnić tą funkcje wyprowadzenie musi być skonfigurowane jako wyjście(DDE4 = 1). Wyprowadzenie OC3B może także pełnić funkcję wyjścia dla licznika z PVM.

AIN1/OC3A (bit 3)
AIN1 : ujemne wejście komparatora analogowego. To wyprowadzenie jest podłączone wprost do ujemnego wejścia komparatora analogowego.
OC3B (Output Compare Match A output) wyprowadzenie PE3 może być użyte jako wyjście zewnętrzne licznika3 OCM1C2 (Timer/Counter3 Output Compare A).Aby pełnić tą funkcje wyprowadzenie musi być skonfigurowane jako wyjście(DDE3 = 1). Wyprowadzenie OC3B może także pełnić funkcję wyjścia dla licznika z PVM.

AIN0/XCK0 (bit 2)
AIN0 dodatnie wejście komparatora analogowego. Wyprowadzenie jest podłączone wprost do dodatniego wejścia komparatora analogowego.
XCK0 zegar zewnętrzny. Bit DDE2 decyduje o tym, czy jest to wejście (DDE2 = 1) czy wyjście (DDE2 = 0). Wyprowadzenie XCK0 jest aktywne tylko wtedy, kiedy USART jest w trybie synchronicznym.

PDO/TXD0 (bit 1)
PDO (SPI Serial Programming Data Output) podczas szeregowego wprowadzania programu to wyprowadzenie pracuje jako linia wyjścia danych dla Atmega 128.
TXD0 wyprowadzenie transmisji dla USART0

PDI/RXD0 (bit 0)
PDO (SPI Serial Programming Data Input) podczas szeregowego wprowadzania programu to wyprowadzenie pracuje jako linia wejścia danych dla Atmega 128.
RXD0 Wejście danych dla USART0. Kiedy odbiornik USART0 jest aktywny, wyprowadzenie PE0 jest skonfigurowane jako wejście niezależnie od wartości bitu DDRE0. Jeśli USART0 wymusza prace tego trybu jako wejścia włączenie rezystorów podciągających jest możliwe przez wpisanie „1” do bitu PORTE0.

Tabela 40 i 41 Alternatywne funkcje portu E
Tabela 40 Dodatkowe funkcje wyprowadzeń PE7...PE4

Nazwa sygnału	PE7/INT7/IC3	PE6/INT6/T3	PE5/INT5/OC3C	PE4/INT4/OC3B
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	OC3C ENABLE	OC3B ENABLE
PVOV	0	0	OC3C	OC3B
DIEOE	INT7 ENABLE	INT6 ENABLE	INT5 ENABLE	INT4 ENABLE
DIEOV	1	1	1	1
DI	INT7 INPUT/IC3 INPUT	INT7 INPUT/T3 INPUT	INT5 INPUT	INT4 INPUT
AIO	–	–	–	–

Tabela 41 Dodatkowe funkcje wyprowadzeń PD3...PD0

Nazwa sygnału	PE3/AIN1/OC3A	PE2/AIN0/XCK0	PE1/PDO/TXD0	PE0/PDI/RXD0
PUOE	0	0	TXEN0	RXEN0
PUOV	0	0	0	PORTE0 • $\overline{PUD}$
DDOE	0	0	TXEN0	RXEN0
DDOV	0	0	1	0
PVOE	OC3B ENABLE	UMSEL0	TXEN0	0
PVOV	OC3B	XCK0 OUTPUT	TXD0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	0	XCK0 INPUT	–	RXD0
AIO	AIN1 INPUT	AIN0 INPUT	–	–

DODATKOWE FUNKCJE PORTU F

Port F może służyć jako wejście analogowe konwertera analogowo cyfrowego(ADC). W przypadku, gdy któreś z wyprowadzeń portu są skonfigurowane jako wyjścia jest niezwykle istotne to, że nie można ich przełączyć na wejścia w czasie konwersji. Nie jest to możliwe ponieważ mogłoby spowodować błędy konwersji. W trybie zgodności z Atmega 103 port F może pracować jedynie jako wejście. Jeśli interfejs JTAG jest aktywny rezystory podciągające wyprowadzeń PF7, PF5, i PF4 będą włączone nawet jeśli nastąpi reset.

TDI, ADC7 (bit 7)
ADC7 : 7. kanał ADC

TDI Szeregowe wejście danych JTAG, jeśli interfejs JTAG jest aktywny to wyprowadzenie nie może pracować jako pin portu we/wy.

TCK, ADC6 (bit 6)
ADC6 : 6. kanał ADC
Szeregowe wyjście danych z rejestrów instrukcji lub danych. Jeśli interfejs JTAG jest aktywny to wyprowadzenie nie może pracować jako pin portu we/wy.

TMS, ADC5 – Port F, Bit 5
ADC5 : 5. kanał ADC
TMS (JTAG Test Mode Select) :Jeśli interfejs JTAG jest aktywny to wyprowadzenie nie może pracować jako pin portu we/wy.

TDO, ADC4 – Port F, Bit 4
ADC4 : 4. kanał ADC
TCK (JTAG Test Clock): Operacje JTAG są synchronizowane z TCK Jeśli interfejs JTAG jest aktywny to wyprowadzenie nie może pracować jako pin portu we/wy.

ADC3 – ADC0 – Port F, Bit 3..0
ADC0-3 : kanały 0 do 3 ADC

Tabela 43 i 44 Alternatywne funkcje portu F
Tabela 43 Dodatkowe funkcje wyprowadzeń PF7...PF4

Nazwa sygnału	PF7/ADC7/TDI	PF6/ADC6/TDO	PF5/ADC5/TMS	PF4/ADC4/TCK
PUOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
PUOV	1	0	1	1
DDOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
DDOV	0	SHIFT_IR + SHIFT_DR	0	0
PVOE	0	JTAGEN	0	0
PVOV	0	TDO	0	0
DIEOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	TDI/ADC7 INPUT	ADC6 INPUT	TMS/ADC5 INPUT	TCKADC4 INPUT

Tabela 44 Dodatkowe funkcje wyprowadzeń PF3...PF0

Nazwa sygnału	PF3/ADC3	PF2/ADC2	PF1/ADC1	PF0/ADC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

DODATKOWE FUNKCJE PORTU G

W trybie zgodności z Atmega 103 port G nie może być używany jako port we/wy. W tym trybie mogą być użyte jedynie funkcje alternatywne tego portu

TOSC1 (bit 4) Jeśli bit AS0 w rejestrze ASSR = 1 w celu uaktywnienia asynchronicznego taktowania licznika0 wyprowadzenie PG4 jest odłączane od portu G i staje się wejściem wzmacniacza odwracającego oscylatora. W tym trybie, jeśli oscylator jest podłączony do tego wyprowadzenia wyprowadzenie nie może być używane jako pin we/wy.

TOSC2 (bit 3) (bit 4) Jeśli bit AS0 w rejestrze ASSR = 1 w celu uaktywnienia asynchronicznego taktowania licznika0 wyprowadzenie PG3 jest odłączane od portu i staje się wyjściem wzmacniacza odwracającego oscylatora. W tym trybie, jeśli oscylator jest podłączony do tego wyprowadzenia wyprowadzenie nie może być używane jako pin we/wy.

ALE (bit 2) Sygnał umożliwiający stosowanie pamięci zewnętrznej

RD (bit 1)Sygnał strobujący odczyt pamięci zewnętrznej

WR (bit 0) Sygnał strobujący zapis pamięci zewnętrznej

Tabela 46 i 47 Alternatywne funkcje portu G

Tabela 46 Dodatkowe funkcje wyprowadzeń PG4...PG1

Nazwa sygnału	PG4/TOSC1	PG3/TOSC2	PG2/ALE	PG1/$\overline{RD}$
PUOE	AS0	AS0	SRE	SRE
PUOV	0	0	0	0
DDOE	AS0	AS0	SRE	SRE
DDOV	0	0	1	1
PVOE	0	0	SRE	SRE
PVOV	0	0	ALE	RD
DIEOE	AS0	AS0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	T/C0 OSC INPUT	T/C0 OSC OUTPUT	–	–

Tabela 46 Dodatkowe funkcje wyprowadzenia PG0

Nazwa sygnału	PG0/$\overline{WR}$
PUOE	SRE
PUOV	0
DDOE	SRE
DDOV	1
PVOE	SRE
PVOV	WR
DIEOE	0
DIEOV	0
DI	–
AIO	–

OPIS REJESTRÓW PORTÓW WEJŚCIA/WYJŚCIA

Rejestr PORTA

BIT	7	6	5	4	3	2	1	0
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr DDRA

BIT	7	6	5	4	3	2	1	0
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr PINA

BIT	7	6	5	4	3	2	1	0
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
ODCZYT/ZAPIS	R	R	R	R	R	R	R	R
WARTOŚĆ POCZ.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Rejestr PORTB

BIT	7	6	5	4	3	2	1	0
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr DDRB

BIT	7	6	5	4	3	2	1	0
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr PINB

BIT	7	6	5	4	3	2	1	0
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
ODCZYT/ZAPIS	O	O	O	O	O	O	O	O
WARTOŚĆ POCZ.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Rejestr PORTC

BIT	7	6	5	4	3	2	1	0
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr DDRC

BIT	7	6	5	4	3	2	1	0
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr PINC

BIT	7	6	5	4	3	2	1	0
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
ODCZYT/ZAPIS	0	0	0	0	0	0	0	0
WARTOŚĆ POCZ.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Rejestr PORTD

BIT	7	6	5	4	3	2	1	0
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr DDRD

BIT	7	6	5	4	3	2	1	0
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr PIND

BIT	7	6	5	4	3	2	1	0
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
ODCZYT/ZAPIS	0	0	0	0	0	0	0	0
WARTOŚĆ POCZ.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Rejestr PORTE

BIT	7	6	5	4	3	2	1	0
	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr DDRE

BIT	7	6	5	4	3	2	1	0
	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr PINE

BIT	7	6	5	4	3	2	1	0
	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0
ODCZYT/ZAPIS	0	0	0	0	0	0	0	0
WARTOŚĆ POCZ.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Rejestr PORTF

BIT	7	6	5	4	3	2	1	0
	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr DDRF

BIT	7	6	5	4	3	2	1	0
	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
ODCZYT/ZAPIS	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr PINF

BIT	7	6	5	4	3	2	1	0
	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0
ODCZYT/ZAPIS	0	0	0	0	0	0	0	0
WARTOŚĆ POCZ.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Rejestr PORTG

BIT	7	6	5	4	3	2	1	0
	–	–	–	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0
ODCZYT/ZAPIS	0	0	0	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr DDRG

BIT	7	6	5	4	3	2	1	0
	–	–	–	DDG4	DDG3	DDG2	DDG1	DDG0
ODCZYT/ZAPIS	0	0	0	O/Z	O/Z	O/Z	O/Z	O/Z
WARTOŚĆ POCZ.	0	0	0	0	0	0	0	0

Rejestr PING

BIT	7	6	5	4	3	2	1	0
	–	–	–	PING4	PING3	PING2	PING1	PING0
ODCZYT/ZAPIS	0	0	0	0	0	0	0	0
WARTOŚĆ POCZ.	0	0	0	N/A	N/A	N/A	N/A	N/A

Uwaga : Rejestry PORTF, DDRF, PINF, PORTG, DDRG, PING nie są dostępne w trybie zgodności z Atmega103

PRZERWANIA ZEWNĘTRZNE

Zewnętrzne przerwania są wywoływane przez wejścia INT7..0. Warto zauważyć, że jeżeli przerwania te są odblokowane będą wywoływane nawet jeśli nóżki INT7.0 sa skonfigurowane jako wyjścia. Cecha ta zapewnia możliwość na generowania przerwań programowych. Zewnętrzne przerwania mogą być wywoływane zboczem opadającym lub narastającym lub niskim poziomem. T to ustawiane zgodnie z opisem rejestru kontroli przerwań zewnętrznych EICRA (INT3..0 i EICRB (INT7..4). Kiedy zewnętrzne przerwania są odblokowane i są wywoływane poziomem, przerwanie będzie zgłaszane przez cały czas utrzymywania niskiego poziomu na wejściu. Rozpoznanie opadającego lub narastającego zbocza przerwania na nóżkach INT7..4 wymaga obecności zegara I/O opisanego w sekcji systemy zegarowe i ich dystrybucja" na stronie 33. Przerwania powodowane niskim poziomem i zboczami na nóżkach INT3..0 są wykrywane asynchronicznie. Powoduje to, że przerwania te mogą być wykorzystywane do wybudzania urządzenia z trybów snu innych niż IDLE . Zegar I/O jest wstrzymywany we wszystkich trybach snu za wyjątkiem trybu IDYL.

Należy zauważyć, że jeżeli przerwania zgłaszane poziomem są wykorzystywane do wybudzenia z trybu obniżonego poboru mocy. Zmieniony poziom napięcia musi być utrzymywany przez jakiś czas, aby uruchomić MCU. To powoduje, że MCU jest mniej czułe na zakłócenia. Zmieniony poziom jest próbkowany dwa razy przez oscylator Watchdog'a. Okres Watchdog'a wynosi 1mikros dla 5.0V w temperaturze 25 stopni. Częstotliwość Wathcdog's zależy od napięcia co pokazano w sekcji „Charakterystyki elektryczne" na stronie 314. MCU wybudzi się, jeżeli wejście ma wymagany poziom podczas próbkowania lub jest ono utrzymywane do końca czasu uruchamiania. Czas uruchamiania jest zdefiniowany przez bezpiecznik SUT co opisana w sekcji „Zegary systemowe i ich dystrybucja" na stronie 33. Jeżeli poziom jest próbkowany dwa razy przez oscylator Watchdog'a ale zaniknie przed upływem czasu uruchamiania , MCU wybudzi się, ale żadne przerwanie nie zostanie wygenerowane. Wymagany poziom musi być utrzymywany wystarczająco długo, aby MCU mogło ukończyć wybudzanie i wywołać poziom przerwania.

- **Rejestr kontroli zewnętrznych przerwań A – EICRA**

Bit	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
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	

Rejestr ten jest nie dostępny w trybie kompatybilności z ATmega103, ale początkowe wartości INT#..0 zdefiniowane jako niski poziom przerwań są takie jak w ATmega103.

a: bity 7..0 – ISC31, ISC30 – ISC00, ISC00: zewnętrzne przerwania 3..0 bity kontroli

Zewnętrzne przerwania 3..0 są aktywowane przez zewnętrzne wejścia INT3..0 jeśli flaga I w rejestrze SREG i odpowiadająca jej maska przerwania w EIMSK są ustawione. Poziom i zbocza na zewnętrznych wyjściach, które aktywują przerwania są zdefiniowane w tabeli 48. Zbocza na wejściach INT3..0 są rejestrowane asynchronicznie. Impulsy na wejściach INT3..0 szersze od minimalnej szerokości impulsu są podane w tabeli 49. Krótsze impulsy nie gwarantują

wygenerowania przerwania. Jeżeli przerwanie niskopoziomowe jest wybrane, niski poziom musi być utrzymywany do ukończenia wykonywanej instrukcji aby wygenerować przerwanie. Jeśli odblokowane, przerwania zgłaszane poziomem będą generować żądania przerwania tak długo jak na danym wyjściu jest stan niski. Podczas zmian bitu ISCn może wystąpić przerwanie. Dlatego, zaleca się najpierw zablokować INTn poprzez wyzerowanie bitu umożliwiającego przerwania w rejestrze EIMSK. Następnie bit ISCn może zostać zmieniony. Wreszcie, flagi przerwań INTn powinny zostać wyzerowane poprzez wpisanie logicznej jedynki do bitu flagi przerwań w rejestrze EIFR nim przerwania zostaną z powrotem odblokowane.

ISCn1	ISCn0	Opis
0	0	Niski poziom INT n wygeneruje żądanie przerwania
0	1	Zarezerwowane
1	0	Opadające zbocze INTn wygeneruje asynchroniczne żądanie przerwania
1	1	Narastające zbocze INTn wygeneruje asynchroniczne żądanie przerwania

Uwaga: 1. n = 3..0
Podczas zmian ISCn1/ISCn0 przerwanie musi być zablokowane poprzez wyzerowanie bitu umożliwiającego przerwania w rejestrze EIMSK. W przeciwnym wypadku przerwanie może wystąpić podczas zmiany tych bitów.

Symbol	Parametr	Warunki	Min	Typ	Max	Jednostki
t_{INT}	Minimalna szerokość impulsu dla asynchronicznego zewnętrznego przerwania			50		ns

- **Rejestr kontroli zewnętrznych przerwań B – EICRB**

Bit	7	6	5	4	3	2	1	0	
	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
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	

a: bit 7..0 – ISC71, ISC70 – ISC41m ISC40: zewnętrzne przerwania 7-4 bity kontroli

Zewnętrzne przerwania 7 – 4 są aktywowane przez zewnętrzne nóżki INT7..4 jeżeli flaga I rejestru SREG i odpowiednie maski przerwań są ustawione. Poziom i zbocza na wyjściach zewnętrznych, które aktywują przerwanie są zdefiniowane w tabeli 50. Wartości na wyjściach INT7..4 są próbkowane przed wykryciem zbocza. Jeśli zbocze lub przełącznik przerwań jest wybrany impulsy trwające dłużej niż jeden cykl maszynowy wygenerują przerwanie. Krótsze impulsy nie gwarantują generacji przerwania. Zauważ, że częstotliwość zegara CPU może być mniejsza niż częstotliwość XTAL jeżeli dzielnik XTAL jest odblokowany. Jeżeli wybrana jest przerwanie wyzwalane niskim poziomem niski poziom musi być utrzymywany aż skończy się wykonywana instrukcja w celu wygenerowania przerwania. Jeśli jest odblokowane, poziomowe zgłoszenie przerwania będzie generować żądanie przerwania tak długo ja na wyjściach jest niski poziom.

ISCn1	ISCn0	Opis
0	0	Niski poziom INTn generuje żądanie przerwania

0	1	Jakakolwiek logiczna zmiana na INTn generuje przerwanie
1	0	Opadające zbocze między próbkowaniem INTn generuje przerwanie
1	1	Narastające zbocze między próbkowaniem INTn generuje przerwanie

Uwaga: n = 7..4

Podczas zmian ISCn1/ISCn0 przerwanie musi być zablokowane poprzez wyzerowanie bitu umożliwiającego przerwania w rejestrze EIMSK. W przeciwnym wypadku przerwanie może wystąpić podczas zmiany tych bitów.

- **Rejestr maskujący zewnętrzne przerwania –EIMSK**

Bit	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
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	

a: bity 7..0 – INT7 – INT 0: Odblokowanie zewnętrznego żądania przerwania

Kiedy na tych bitach są zapisane jedynki i bit I w rejestrze statusu (SREG) jest ustawiony (!), odpowiadające bitom zewnętrzne przerwania są odblokowane. Bity kontroli przerwań w rejestrach EICGA i EICGB definiują czy przerwanie zewnętrzne jest aktywowane narastającym, opadającym zboczem, czy poziomem. Aktywność na którymkolwiek z tych wyjść spowoduje żądanie przerwania nawet jeśli dana nóżka jest wyjściem. Zapewnia to możliwość generowania przerwań za pomocą oprogramowania.

- **Rejestr flag przerwań zewnętrznych EIFR**

Bit	7	6	5	4	3	2	1	0	
	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	EIFR
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	

a: bity 7..0 – INTF7 - INTF0: flagi przerwań zewnętrznych 7..0

Kiedy zbocze albo poziom spowoduje żądanie przerwania na wejściach INT7..0 flaga INTF7..0 zostanie ustawiona na jeden. Jeżeli bit I w rejestrze SREG i odpowiednie bity przerwań w EIMSK są odblokowane (1) MCU przeskoczy do wektora przerwań. Flaga jest zerowana po wykonaniu procedury obsługi przerwania. Lub może zostać wyczyszczona poprzez wpisanie logicznej jedynki do niej. Flagi INTF7..0 są zawsze czyszczone w przypadku wywoływania przerwań poziomem. Wchodząc w stan snu z zablokowanymi flagami INT3..0 bufory wejściowe na tych nóżkach będą zablokowane. Może to powodować logiczną zmianę w wewnętrznych sygnałach, które będą ustawiały flagi INTF3..0. Przejrzyj „Odblokowanie binarnego wejścia i tryby snu”.

8-BITOWY LICZNIK Z PWM I OPERACJE ASYNCHRONICZNE

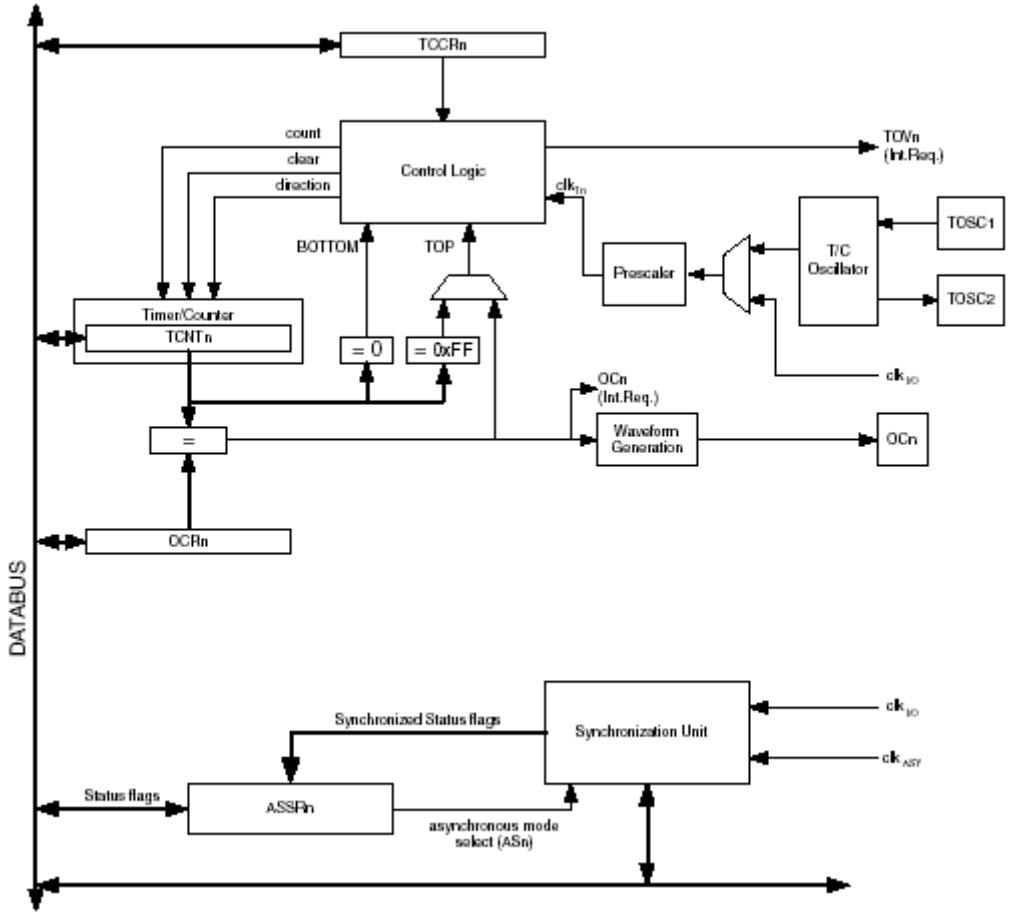
Główne cechy:

• jednokanałowy licznik
• kasowanie licznika podczas dopasowania
• wolny od zakłóceń, Modulacji Szerokości Impulsu z Korekcją Fazy (PWM)
• generator częstotliwości
• 10-bitowy prescaler licznika
• źródła przerwań przepełnienia i dopasowania (TOV0 i OCF0)
• możliwość zegarowania z zewnętrznego generatora 32 kHz niezależnie od zegara we/wy

Przegląd

Uproszczony diagram blokowy 8-bitowega zegara/licznika przedstawiony jest na rys. 33.
Rzeczywiste rozmieszczenie styków we/wy przedstawione jest w “Konfiguracji styków” na stronie 2.
Rejestry we/wy dostępne dla CPU , włączając bity we/wy I styki we/wy zostały pogrubione.
Opis specyficznych rejestrów we/wy oraz rozmieszczenie bitów znajduje się na stronie 98.

Rys. 33. Diagram blokowy 8-bitowego zegara/licznika



Rejestry

Zegar/Licznik (TCNT0) oraz zewnętrzny rejestr porównawczy (OCR0) są 8-bitowymi rejestrami.
Sygnały żądania przerwania (skrót: Int.Req.) są widoczne w rejestrze flagi przerwania zegara (TIFR).
Wszystkie przerwania są osobno maskowane w rejestrze maski przerwania zegara (TIMSK).
TIFR oraz TIMSK nie są ukazane na rysunkach , jeśli rejestry są współdzielone przez różne urządzenia zegarowe.
Zegar/Licznik może być taktowany wewnętrznie za pomocą prescalera lub asynchronicznie
Ze styków TOSC1/2. Asynchroniczne operacja jest kontrolowana przez rejestr asynchronicznego stanu (ASSR).
Zegar/Licznik jest nieaktywny gdy nie ma wybranego żadnego źródła zegarowego. Podwójnie buforowany zewnętrzny rejestr porównawczy (OCR0) jest zawsze porównywany z wartością Zegar/Licznik. Rezultat porównania może być użyty przez generator przebiegów falowych
W celu generacji PWM lub zmiennej częstotliwości wyjściowej na styku porównania wyjścia (OC0).
Szczegóły znajdują się na stronie 89.
Zdarzenie zgodności porównania ustawi flagę porównania (OCF0), która może zostać użyta do generacji żądania przerwania porównania wyjścia.

Pojęcia
Mała litera “n” zastępuje numer Zegar/Licznik, w tym przypadku 0.
Pojęcia przedstawione w tabeli 51 są często używane w tym dokumencie.

Tabela 51 **Pojęcia**

BOTTOM	Licznik osiąga BOTTOM gdy jego stan wynosi 0 (0x00).
MAX	Licznik osiąga MAXimum gdy jego stan wynosi 0xFF(dziesiętnie 255).
TOP	Licznik osiąga MAXimum gdy jego stan osiąga największą wartość w licznej sekwencji. Wartość TOP Mozę być przypisana do stałej wartości 0xFF(MAX) lub do wartości znajdującej się w rejestrze OCR0. Przypisanie to jest zależne od trybu operacji.

ŹRÓDŁA ZEGAROWE DLA ZEGAR/LICZNIK (TIMER/COUNTER)

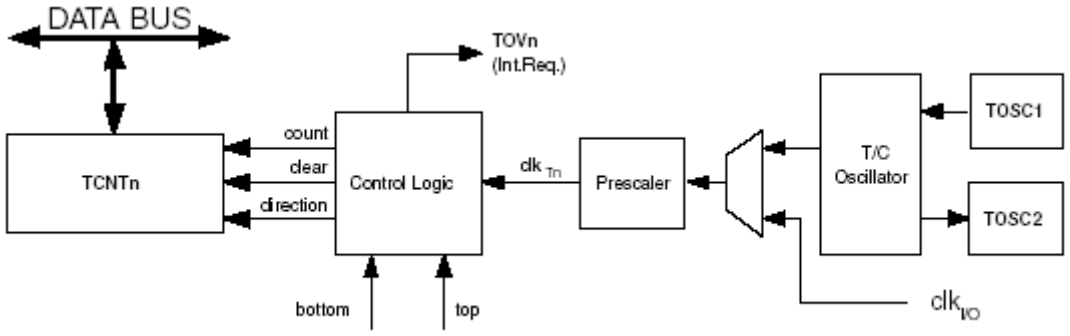
Źródła

Zegar/Licznik może być taktowany przez wewnętrzne synchroniczne źródło taktowania lub zewnętrzne asynchroniczne. Źródło zegarowe clkT0 jest domyślnie równe zegarowi MCU, clkI/O.
Gdy bit AS0 w rejestrze ASSR jest ustawiany na jedynkę logiczną, źródło zegarowe jest brane z Zegar/Licznik oscylatora podłączonego do TOSC1 i TOSC2.

Jednostka licznika

Główną częścią 8-bitowego Zegara/Licznika jest dwukierunkowy programowalny licznik.
Rysunek 34 prezentuje diagram blokowy licznika oraz jego otoczenie.

Rys. 34. Diagram blokowy licznika



Opis sygnałów (sygnały wewnętrzne):
Mount inkrementuje lub dekrementuje TCNT0 .

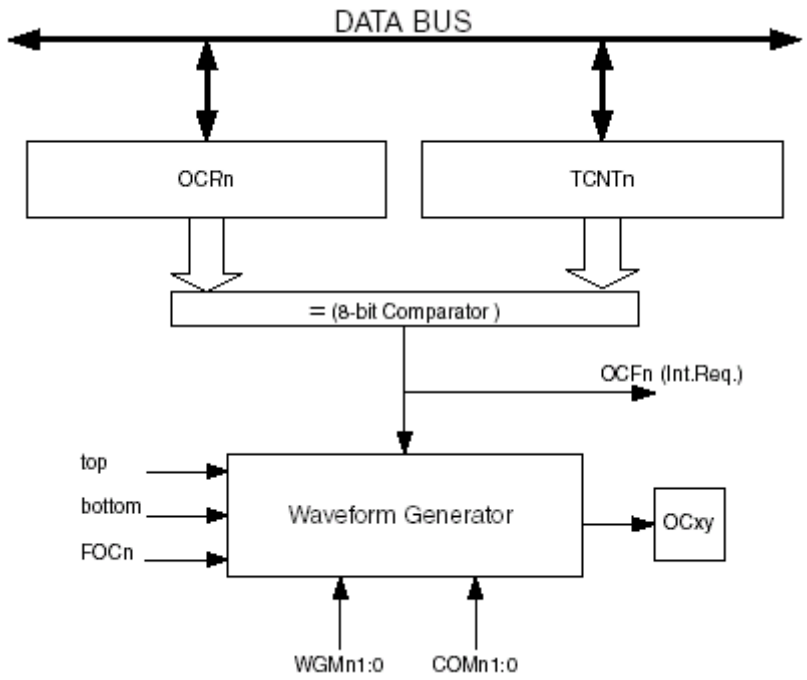
direction Wybiera inkrementację lub dekrementację.
clear resetuje TCNT0 (ustawia wszystkie bity na zero).
clk$_{T0}$ Zegar/Licznik .
top sygnalizuje, że TCNT0 osiągnął maksymalną wartość.
bottom sygnalizuje, że TCNT0 osiągnął minimalną wartość(zero).

W zależności od trybu używanych operacji, licznik jest resetowany, inkrementowany lub dekrementowany w trakcie każdego taktu zegarowego (clk$_{T0}$). clk$_{T0}$ może być generowany przez zewnętrzne lub wewnętrzne źródło zegarowe , wybierane przez bity clock select (CS02:0). W przypadku, gdy nie jest wybrane żadne źródło zegarowe (CS02:0 = 0) zegar jest zatrzymywany. Jednakże wartość TCNT0 może być odczytana przez CPU, niezależnie od tego czy clk$_{T0}$ jest obecny.
Zapis CPU ma wyższy priorytet od wszystkich resetów liczników oraz operacji obliczeniowych.
Sekwencja zliczania zależna jest od ustawień bitów WGM01 oraz WGM00 znajdujących się w rejestrze sterowania Zegar/Licznik (TCCR0). Istnieją ścisłe powiązania pomiędzy sposobem zliczania licznika i generacja przebiegu falowego na OC0. Więcej szczegółów nt. sekwencji zliczania znajdziesz na stronie 92.
Flaga przepełnienia Zegara/Licznika (TOV0) jest ustawiana w zależności od trybu operacji wybranego przez bity WGM01:0. TOV0 może zostać użyty do generacji przerwania CPU.

Komparator wyjściowy

8-bitowy komparator stale porównuje TCNT0 z zewnętrznym rejestrem porównawczym (CR0). Ilekroć TCNT0 równa się OCR0, komparator o tym sygnalizuje. Jeśli wynik porównania jest pozytywny flaga OCF0 zostaje ustawiona w trakcie następnego cyklu zegarowego. Jeśli OCIE0 = 1, flaga OCF generuje przerwanie output compare.
Flaga OCF0 flag jest automatycznie resetowana gdy przerwanie zostaje wykonane. Flaga OCF0 może zostać zresetowana programowo przez zapis jedynki logicznej w miejsce odpowiedniego bitu we/wy.

Rys. 35. **Komparator wyjściowy , diagram blokowy**



Rejestr OCR0 jest podwójnie buforowany e czasie trybu modulacji szerokości impulsu(PWM).

W trybach normalnym i CTC , podwójne buforowanie jest wyłączone. Podwójne buforowanie synchronizuje uaktualnienie rejestru porównawczego OCR0 do wartości górnej lub dolnej zliczanej sekwencji. Synchronizacja zapobiega występowaniu impulsów niesymetrycznych, nieparzystej długości, dzięki czemu wyjście jest wolne od zakłóceń.
Dostęp do rejestru OCR0 może wydawać się złożony , ale to nie jest przypadek. Jeśli podwójne buforowanie jest włączone , CPU ma dostęp do bufora rejestru OCR0, w przeciwnym wypadku CPU ma bezpośredni dostęp do OCR0.

Wymuszenie dopasowania na wyjściu

W trybach generacji przebiegów falowych bez PWM , wyjście porównania komparatora może być wymuszone przez zapis jedynki do bitu FOC0. Wymuszenie pozytywnego wyniku porównania nie ustawi flagę OCF0 i nie zresetuje zegara. Styk OC0 zostanie zaktualizowany tylko gdy zaszła rzeczywiście zaszła taka zgodność (ustawienie bitu COM01:0 określa stan styku OC0).

Blokowanie komparatora poprzez zapis TCNT0

Wszystkie operacje zapisu CPU do rejestru TCNT0 blokują każdy wynik porównania, który wystąpi podczas następnego cyklu zegarowego, nawet gdy zegar jest wyłączony. Ta cecha pozwala zainicjalizować OCR0 tą samą wartością co TCNT0 bez wymuszania przerwania gdy Zegar/Licznik jest włączony.

Wykorzystanie komparatora wyjściowego

Odkąd zapis TCNT0 w każdym trybie operacji blokuje na jeden cykl zegarowy wszystkie wyniki porównania , istnieje ryzyko podczas zmiany TCNT0 w trakcie używania kanału porównania wyjścia, niezależnie od tego czy Zegar/Licznik pracuje. Jeśli wartość zapisana do TCNT0 równa się wartości CR0 , wynik porównania zostanie

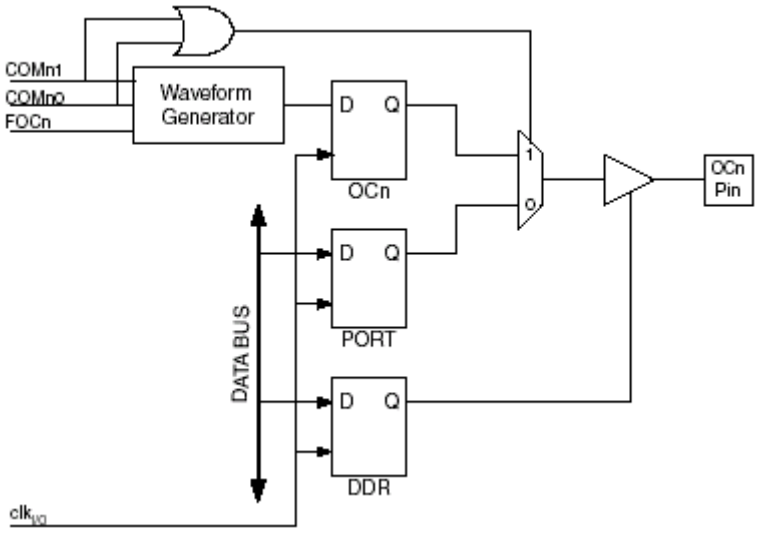
pominięty, skutkiem czego zostanie wygenerowany nieprawidłowy kształt fali. Nie zapisuj również wartości TCNT0 równej BOTTOM gdy licznik zlicza w dół.

Najłatwiejszą metodą ustawienia wartości OC0 jest wymuszenie bitu FOC0 w normalnym trybie.
Rejestr OC0 trzyma swoją wartość nawet podczas zmiany trybów generacji fali.
Bądź świadom tego, że bity COM01:0 nie są podwójnie buforowane razem z wartością porównania. Zmiana bitów COM01:0 następuje natychmiastowo.

JEDNOSTKA KOMPARACJI WYJŚCIA

Bity COM01:0 mają dwa zastosowania. Generator przebiegu falowego używa bitów COM01:0 w celu zdefiniowania stanu OC0 podczas następnego pozytywnego wyniku porównania. Bity COM01:0 sterują źródłem styku OC0. Rys. 36 pokazuje uproszczony schemat logiczny afektowany przez ustawienie bitu COM01:0. Rejestry we/wy, bity we/wy , oraz styki we/wy zostały pogrubione.

Rys 36. Jednostka komparacji wyjścia, schemat



Ogólną funkcja portu we/wy jest zamiana OC0 z generatora przebiegu falowego jeśli oba bity COM01:0 są ustawione. Jednakże kierunek styku OC0 (we lub wy) jest nadal sterowany przez rejestr kierunku danych DDR dla styku port. Bit rejestru kierunku danych dla styku OC0 (DDR_OC0) musi zostać ustawiony jako wyjście przed pojawieniem się wartości OC0 na styku. Funkcja ta nie jest zależna od trybu generacji przebiegu falowego. Budowa styku output compare pozwala na inicjalizację stanu OC0 przed włączeniem wyjścia. Niektóre ustawienia bitów OM01:0 bit są zarezerwowane dla pewnych trybów operacji (patrz strona 98).

Generator fal używa bitów COM01:0 w odmienny sposób w trybach: normalnym, CTC, oraz PWM.
Dla wszystkich trybów ustawienie COM01:0 = 0 mówi generatorowi, że żadna operacja nie zostanie wykonana na rejestrze OC0 w ciągu następnego udanego porównania. Więcej informacji znajdziesz na stronach 99 (tabela 53, tabela 54) oraz 100(tabela 55). Zmiana stanu bitów COM01:0 odniesie skutek w trakcie pierwszego udanego porównania po zapisie bitów. Dla trybów innych niż PWM , można wymusić natychmiastową zmianę stanu bitów poprzez użycie bitów strobujących FOC0.

Tryby operacji

Tryb operacji ,np. zachowanie Zegar/Licznik oraz wyjściowych styków porównania jest zdefiniowany przez kombinację trybu generacji fali (WGM01:0) oraz trybu bitów wyjścia porównania (COM01:0).
Bity wyjścia porównania nie wpływają na sekwencję zliczania, ale tryb generacji fali ma taki wpływ.
Bity COM01:0 sterują odwróceniem lub nie generowanego wyjścia PWM. Dla trybów innych niż PWM bity COM01:0 decydują czy wyjście powinno być ustawione, zresetowane, czy przełączone na stan pozytywnego wyniku porównania.

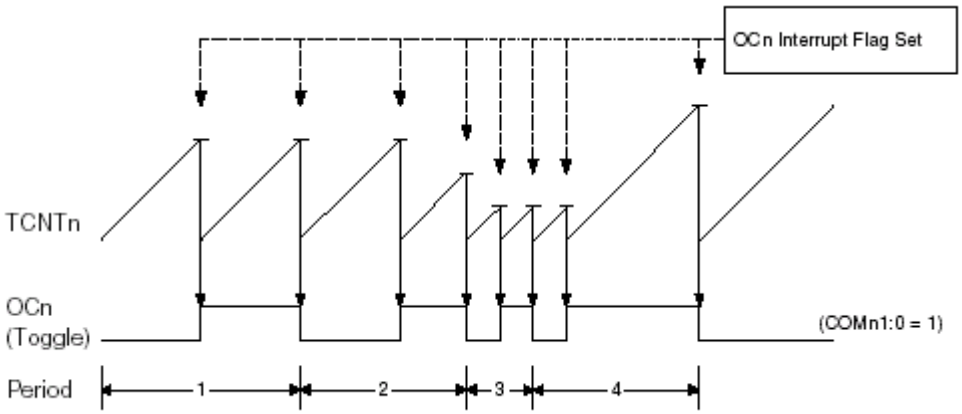
Tryb Normalny

Najprostszym trybem operacji jest tryb normalny (WGM01:0 = 0). W tym trybie licznik zlicza zawsze w górę (inkrementacja), nie jest wykonywany reset licznika. Licznik po osiągnięciu maksymalnego stanu 8-bit (TOP = 0xFF) zaczyna zliczać od początku (bottom =0x00). W normalnych operacjach flaga przepełnienia Zegar/Licznik (TOV0) zostanie ustawiona w cyklu zegarowym, w którym wartością TCNT0 będzie zero. W tym przypadku flaga TOV0 flag zachowuje się jak dziewiąty bit , przy czym ona jest ustawiona a nie zresetowana. Jednakże, w połączeniu z przerwaniem przepełnienia zegara, które automatycznie kasuje flagę TOV0 flag, rozdzielczość zegara może zostać zwiększona programowo. Nowy stan licznika może być zapisany w dowolnym momencie. Zewnętrzna jednostka porównawcza może zostać użyta w celu generacji przerwania. W trybie normalnym nie powinno się używać tej jednostki do generacji przebiegów falowych ponieważ w taka operacja zabiera zbyt wiele czasu procesora.

Tryb CTC (kasowanie zegara, przy zgodności wyjścia)

W tym trybie (WGM01:0 = 2), rejestr OCR0 jest używany do zmiany rozdzielczości licznika. W trybie CTC licznik jest ustawiany na zero gdy jego wartość (TCNT0) wynosi OCR0. OCR0 definiuje górną wartość(top) licznika, a co za ty idzie rozdzielczość. Ten tryb upraszcza operację zliczania zewnętrznych zdarzeń. Diagram czasowy trybu CTC znajduje się na rys. 37. Stan licznika (TCNT0)jest zwiększany aż do zajścia równości między TCNT0 a OCR0, wówczas licznik (TCNT0) jest kasowany.

Rys. 37. Tryb CTC, diagram czasowy



Przerwanie może być generowane za każdym razem gdy wartość licznika osiągnie TOP, poprzez użycie flagi OCF0. Jeśli przerwanie wystąpiło, rutyna obsługująca przerwanie może zostać użyta w celu uaktualnienia wartości TOP. Jednakże zmiana wartości TOP na wartość bliską BOTTOM, gdy licznik zlicza bez prescalera lub z małą wartością prescalera musi zostać wykonana uważnie, ponieważ tryb CTC nie obsługuje podwójnego buforowania.
Jeśli nowa wartość zapisana do OCR0 jest mniejsza niż obecna wartość TCNT0, licznik pominie pozytywny wynik porównania.
Częstotliwość generowanej fali w trybie CTC opisuje poniższe równanie:

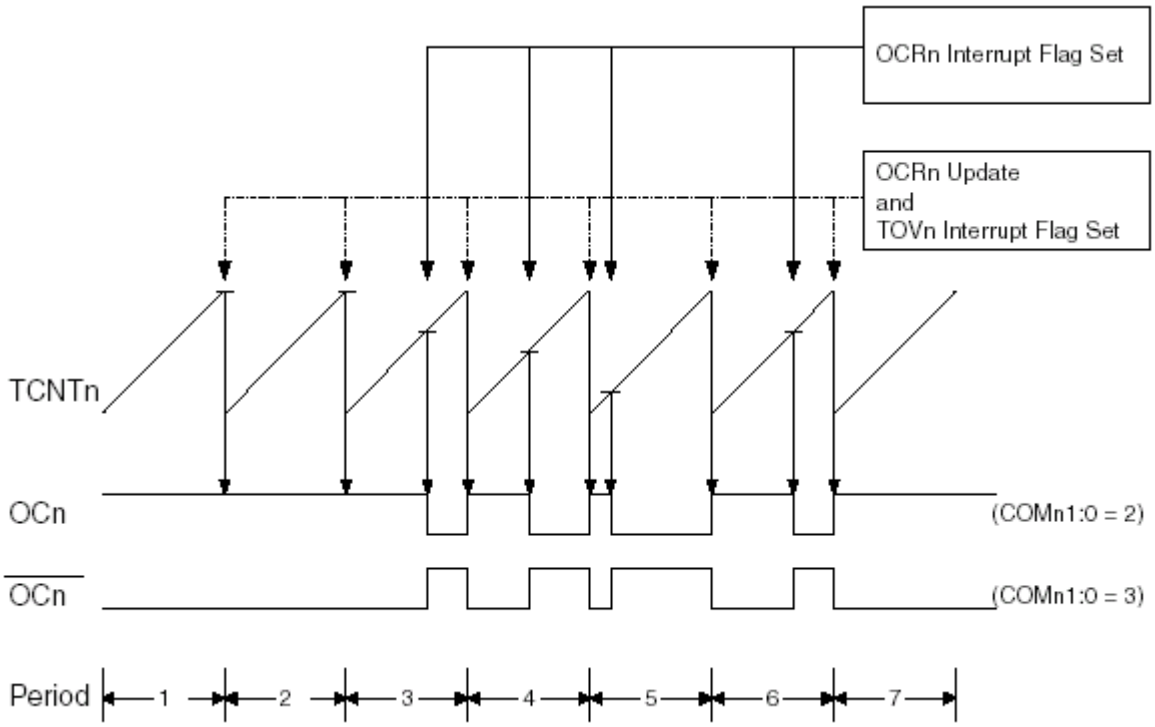
$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

.
,gdzie:
N – współczynnik prescalera (1, 8, 32, 64, 128, 256, 1024).

Szybki tryb Modulacji Szerokości Impulsu (Szybki tryb PWM)

Szybki tryb PWM (WGM01:0 = 1) zapewnia opcję generacji fal wysokiej częstotliwości. Szybki tryb różni się od innych trybów PWM operacją na pojedynczym zboczu. Licznik zlicza cyklicznie od dołu(BOTTOM) do góry (MAX). W trybie nieodwracającym wyjścia porównującego, OC0 jest resetowany gdy zajdzie równość wartości TCNT0 i OCR0, i ustawiany na wartość BOTTOM. W trybie odwracającym, wyjście jest ustawiane gdy zajdzie powyższa równość i resetowane przy wartości minimalnej (BOTTOM). Przez operacje wykonywane na pojedynczym zboczu częstotliwość pracy w szybkim trybie PWM może być dwa razy większa niż przy operacjach wykonywanych na podwójnym zboczu. Wysoka częstotliwość czyni szybki tryb PWM odpowiedni do regulacji zasilania, prostowania, przetworników analogowo-cyfrowych. Szybkim trybie PWM wartość licznika jest zwiększana aż do momentu osiągnięcia wartości maksymalnej(MAX). Licznik jest kasowany w trakcie następnego cyklu zegarowego.

Rys 38. Szybki tryb PWM, diagram czasowy



Flaga przepełnienia licznika TOV0 jest ustawiana za każdym razem gdy licznik osiąga wartość MAX. Jeśli jedno z przerwań jest włączone, podprogram obsługi przerwań może być użyty do aktualizowania wartości porównania. W tym trybie jednostka porównawcza pozwala na generację fal na styku OC0.
Ustawienie bitów COM01:0 na 2 uruchomi tryb nieodwrócony PWM, a ustawienie tych bitów na 3 uruchomi tryb odwrócony. Rzeczywista wartość OC0 będzie widoczna na styku portu jeśli kierunek danych portu będzie ustawiony jako wyjście. Fala PWM jest generowana poprzez ustawienie(lub skasowanie) rejestru OC0 w momencie zgodności wartości OCR0 i TCNT0, ustawienie i skasowanie rejestru OC0 w trakcie cyklu zegarowego skasuje licznik(zmiana z wartości MAX na BOTTOM).

Częstotliwość PWM dla wyjścia może być obliczona wg następującego wzoru:

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

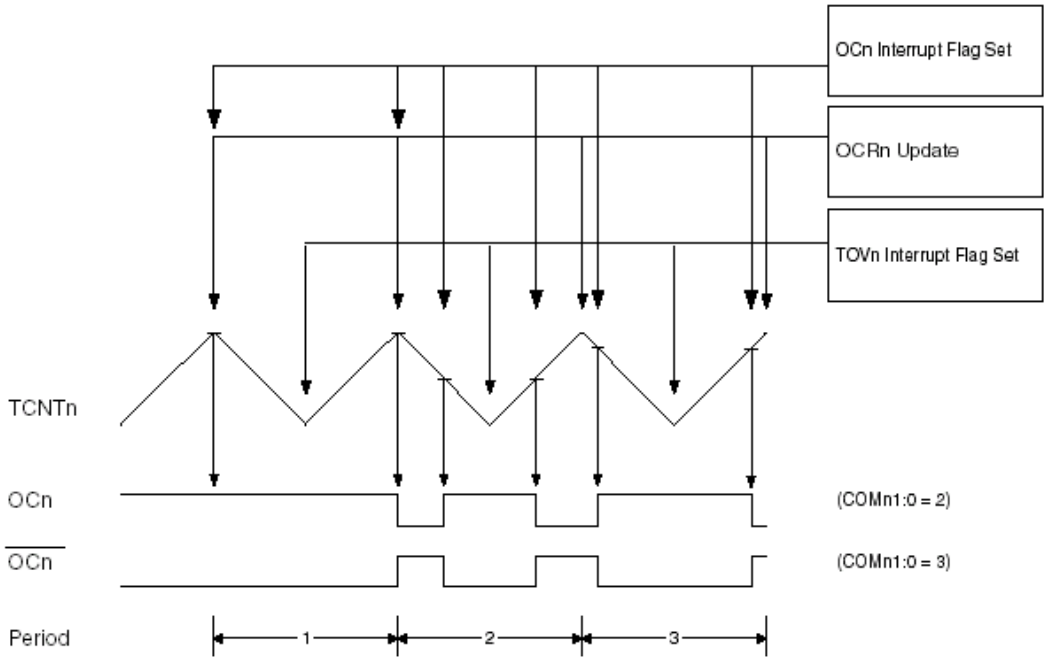
,gdzie:
N – reprezentuje współczynnik prescalera (1, 8, 32, 64, 128, 256, lub 1024).

Skrajne wartości rejestru OCR0 reprezentują specjalne przypadki generacji fali PWM na wyjściu w trakcie szybkiego trybu PWM

Tryb Modulacji Szerokości Impulsu z Korekcją Fazy

Modulacja szerokości impulsu z korekcją fazy (WGM01:0 = 3) posiada opcję generacji przebiegów PWM z korekcją fazy. Tryb PWM z korekcją fazy jest oparty o operacje dwuzboczowe. Licznik liczy kolejno od BOTTOM do MAX i potem od MAX do BOTTOM W trybie porównania wyjścia bez odwracania, porównanie wyjścia (OC0) jest zerowane gdy porównanie TCNT0 i OCR0 jest pozytywne w trakcie zliczania w górę, i ustawiane w trakcie liczenia w dół. W trybie porównania wyjścia z odwracaniem wykonywana jest przeciwna operacja. Działanie w oparciu o dwa zbocza powoduje ograniczenie maksymalnej częstotliwości w porównaniu do działania na jednym zboczu. Jednakże, z powodu symetrii trybu PWM pracującego na dwóch zboczach, są one rekomendowane do działania w zastosowaniach związanych z kontrolą silników. Rozdzielczość PWM dla trybu PWM z korekcją fazy

Rys. 39. Tryb PWM z korekcją fazy, diagram czasowy



Flaga przepełnienia licznika (TOV0) jest ustawiana za każdym razem, gdy stan licznika jest równy BOTTOM. Flaga przerwania może zostać użyta w celu generacji przerwania w momencie osiągnięcia przez licznik wartości BOTTOM. W trybie PWM z korekcją fazy, jednostka komparatora zezwala na generacje przebiegów na styku OC0. Tryb PWM bez odwracania wyjścia można osiągnąć przez wystawienie bitów COM01:0 na 2, zaś tryb PWM z odwróconym wyjściem poprzez ustawienie COM01:0 na 3. Rzeczywista wartość OC0 będzie widoczna na nóżce portu, jeśli nóżka ta ustawiona jest jako wyjście. Przebiegi PWM są generowane poprzez kasowanie(lub ustawianie) rejestru OC0 w trakcie zajścia równości w czasie porównania OC0 i TCNT0, gdy licznik zlicza w górę, i w sposób przeciwny gdy licznik zlicza w dół. Częstotliwość na wyjściu trybie PWM z korekcją fazy można obliczyć według wzoru:

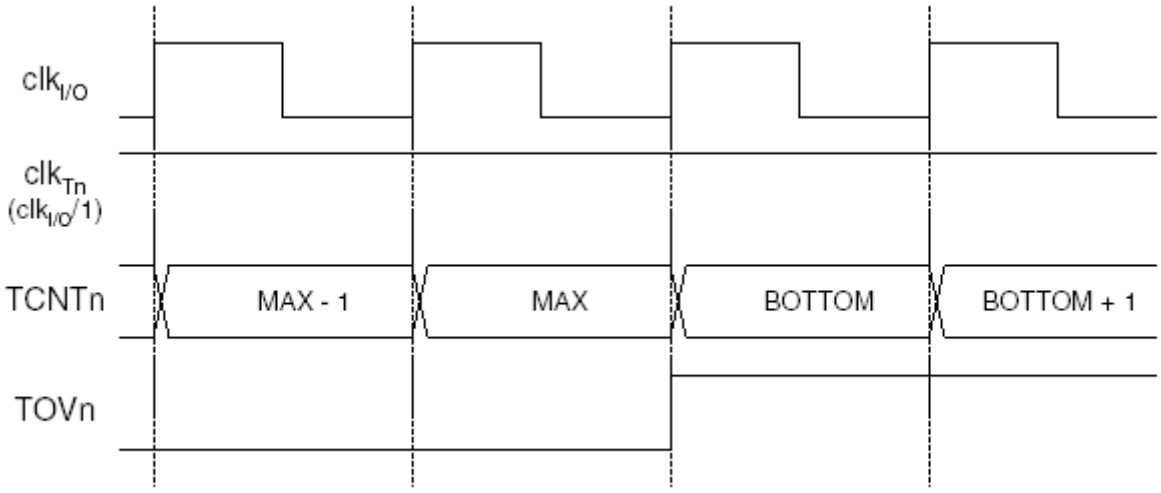
$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

gdzie:
N – współczynnik prescalera (1, 8, 32, 64, 128, 256, lub 1024).

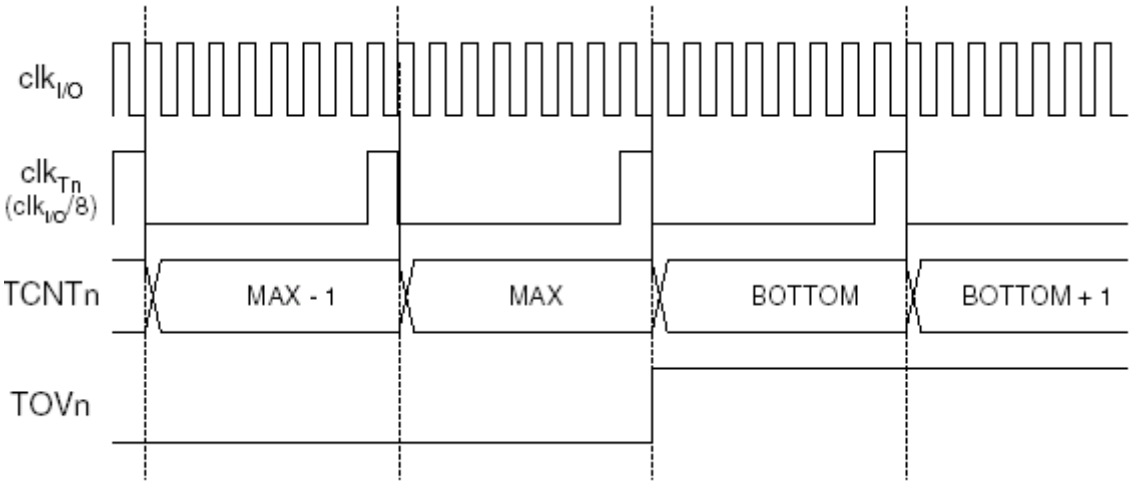
Skrajne wartości rejestru OCR0 reprezentują specjalne przypadki generacji fali PWM na wyjściu w trakcie trybu PWM z korekcją fazy. Jeśli OCR0 jest równy BOTTOM, wyjście będzie ciągle w stanie niskim, gdy OCR0 wyniesie MAX wyjście będzie stale w stanie wysokim dla trybu PWM bez odwracania. W odwróconym trybie PWM wyjście będzie miało przeciwne wartości logiczne.

Przebiegi czasowe

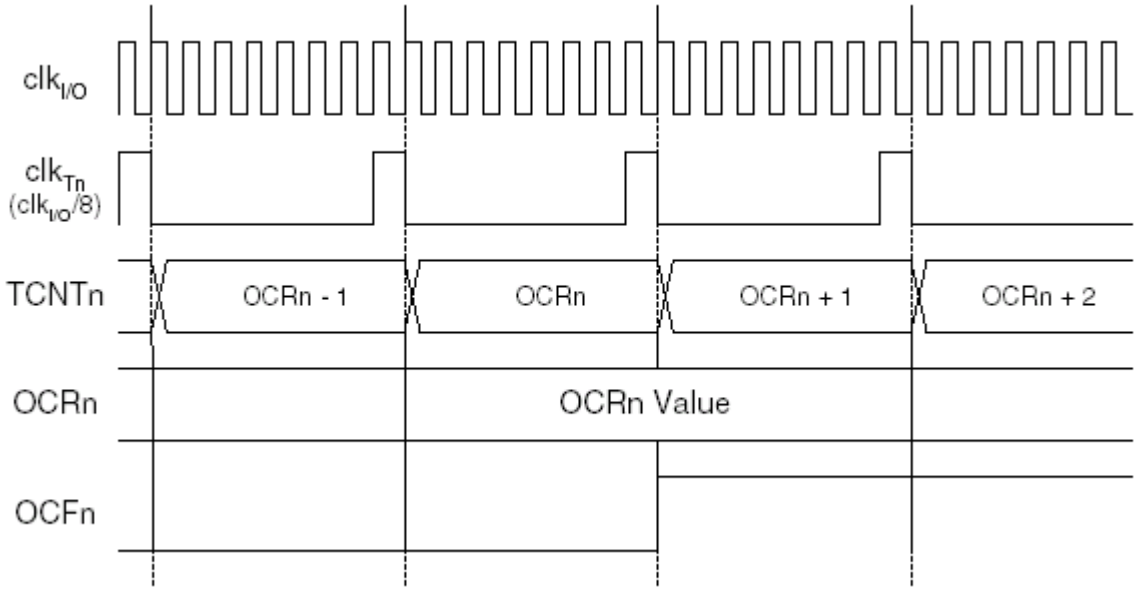
Rys. 40. Zegar/Licznik – diagram czasowy, bez prescalowania



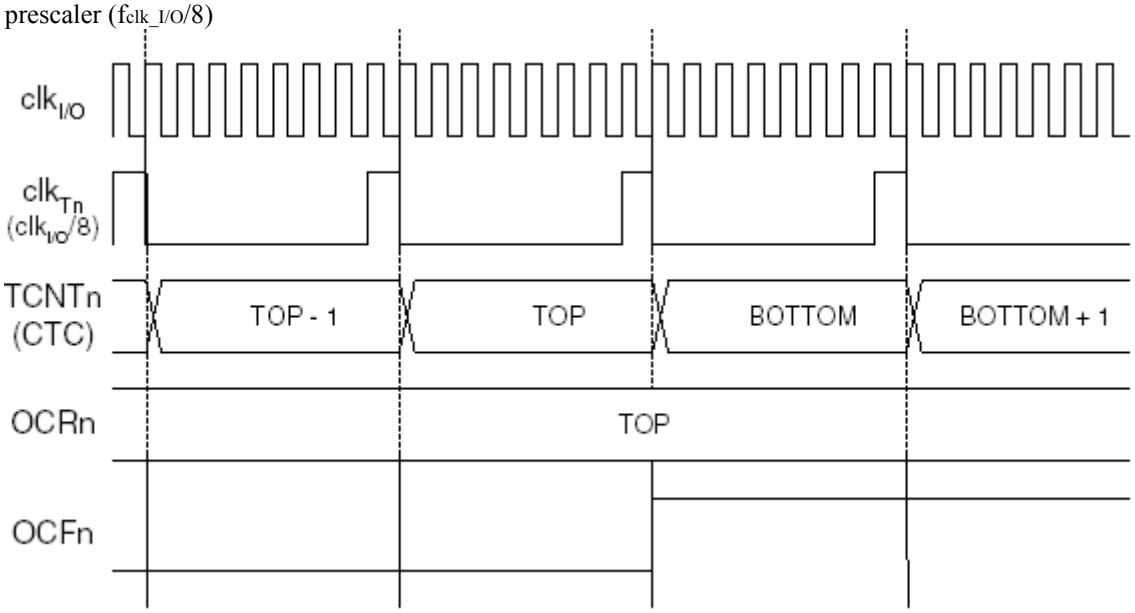
Rys. 41. Zegar/Licznik – diagram czasowy, prescaler ($f_{clk\_I/O}/8$)



Rys. 42. Zegar/Licznik – diagram czasowy, ustawianie OCF0, prescaler ($f_{clk\_I/O}/8$)



Rys. 43 Zegar/Licznik – diagram czasowy, kasowanie zegara w trybie zgodności porównania, prescaler ($f_{clk\_I/O}/8$)



8-BITOWY ZEGAR/LICZNIK - OPIS REJESTRÓW

REJESTR TCCR0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Odczyt/Zapis(R/W)	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

• Bit 7 – FOC0: wymuszenie komparacji wyjścia

Bit FOC0 jest nieaktywny tylko w trybie PWM. Jednakże dla zachowania kompatybilności z przyszłymi urządzeniami, bit ten musi być ustawiony na zero podczas zapisu TCCR0 w trybie PWM. Podczas zapisu jedynki logicznej do bitu FOC0, zostaje wymuszane dopasowanie na generatorze przebiegów falowych. Wyjście OC0 zmienia się zgodnie z ustawieniami bitów COM01:0. Bit FOC0 jest zaimplementowany jako strobujący. FOC0 nie może generować przerwania, ani kasować licznika w trybie CTC w którym OCR0 jest wartością TOP. Bit FOC0 jest odczytywany jako zero.

• Bit 6, 3 – WGM01:0: Tryb generacji przebiegów falowych

Bity te sterują kolejnością zliczania licznika, źródłem maksymalnej (TOP) wartości licznika, wybierają typ generowanego przebiegu falowego. Tryby działania jednostki Zegar/Licznik: normalny, CTC, dwa tryby PWM.

Tabela 52. Tryb generacji przebiegów falowych – opis bitów

Tryb	WGM01 (CTC0)	WGM00 (PWM0)	Tryb Operacji Zegara/Licznika	TOP	Aktualizacja OCR0 przy	Ustawienie flagi TOV0 przy:
0	0	0	Normalny	0xFF	Natychmiastowo	MAX
1	0	1	PWM, korekcja fazy	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Natychmiastowo	MAX
3	1	1	Szybki PWM	0xFF	TOP	MAX

• **Bit 5:4 – COM01:0:** Tryb zgodności komparacji wyjścia

Bity te sterują pinem porównania wyjścia (OC0). Jeśli co najmniej jeden bit z bitów COM01:0 jest ustawiony, wyjście OC0 unieważnia normalne działanie pinu I/O do którego jest ono podłączone.
Gdy OC0 jest podłączony do styku, funkcja bitów COM01:0 jest zależna od ustawień WGM01:0.

Tabela 53. Compare Output Mode, non-PWM Mode

COM01	COM00	Opis
0	0	Normalne działanie portu, OC0 odłączony
0	1	Zmiana OC0 przy dopasowaniu
1	0	Kasowanie OC0 przy dopasowaniu
1	1	Ustawienie OC0 przy dopasowaniu

Tabela 54. Compare Output Mode, Fast PWM Mode(1)

COM01	COM00	Opis
0	0	Normalne działanie portu, OC0 odłączony
0	1	Zarezerwowane
1	0	Kasowanie OC0 przy dopasowaniu, ustawienie OC0 przy TOP
1	1	Ustawienie OC0 przy dopasowaniu, kasowanie OC0 przy TOP

Tabela 55. Compare Output Mode, Phase Correct PWM Mode(1)

COM01	COM00	Opis
0	0	Normalne działanie portu, OC0 odłączony
0	1	Zarezerwowane

1	0	Kasowanie OC0 przy dopasowaniu gdy licznik zlicza w górę. Ustawienie OC0 przy dopasowaniu gdy licznik zlicza w dół.
1	1	Ustawienie OC0 przy dopasowaniu gdy licznik zlicza w górę. Kasowanie OC0 przy dopasowaniu gdy licznik zlicza w dół.

.

• Bit 2:0 – CS02:0: Wybór zegara

Tabela 56. Opis bitu wyboru zegara

CS02	CS01	CS00	Opis
0	0	0	Brak źródła zegarowego (Zegar/Licznik zatrzymany)
0	0	1	clk_{T0S}/(brak prescalera)
0	1	0	clk_{T0S}/8 (z prescalera)
0	1	1	clk_{T0S}/32 (z prescalera)
1	0	0	clk_{T0S}/64 (z prescalera)
1	0	1	clk_{T0S}/128 (z prescalera)
1	1	0	clk_{T0S}/256 (z prescalera)
1	1	1	clk_{T0S}/1024 (z prescalera)

Timer/Counter Register – TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCN0[7:0]								TCN0
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Rejestr jednostki Zegar/Licznik umożliwia bezpośredni dostęp dla operacji zapisu i odczytu.
Zapis do rejestru TCNT0 usuwa dopasowanie podczas następnego cyklu zegarowego. Modyfikacja licznika (TCNT0) podczas jego pracy stwarza ryzyko utraty dopasowania między TCNT0 i rejestrem OCR0.

Output Compare Register – OCR0

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Rejestr porównania wyjścia zawiera 8-bitową wartość, która jest stale porównywana z wartością licznika (TCNT0). Dopasowanie może być wykorzystane do generacji przerwania lub przebiegu falowego na styku OC0.

Zegar/Licznik - Asynchroniczne Operacje

Asynchronous Status Register – ASSR

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	AS0	TCN0UB	OCR0UB	TCR0UB	ASSR
Odczyt/Zapis (R/W)	R	R	R	R	R/W	R	R	R	
Wartość początkowa	0	0	0	0	0	0	0	0	

• Bit 3 – AS0: Asynchroniczny Zegar/Licznik0
Jeśli AS0 ma zapisaną wartość zero, Zegar/Licznik0 jest taktowany z zegara We/Wy, $clk_{I/O}$. W przeciwny przypadku (1 w AS0), Zegar/Licznik jest taktowany z oscylatora podłączonego do TOSC1. Po zmianie wartości AS0, zawartość TCNT0, OCR0, oraz TCCR0 może być niepoprawna.

• Bit 2 – TCN0UB: Zegar/Licznik Update Busy
Bit ten zostaje ustawiony gdy Zegar/Licznik pracuje asynchronicznie i TCNT0 jest zapisywany.
Bit ten zostaje skasowany sprzętowo, po uaktualnieniu TCNT0 z rejestru tymczasowych wartości. Zero logiczne w tym bicie wskazuje gotowość TCNT0 do zapisania nowej wartości.

• Bit 1 – OCR0UB: Rejestr0 porównania wyjścia Update Busy
Bit ten zostaje ustawiony gdy Zegar/Licznik pracuje asynchronicznie i OCR0 jest zapisywany.
Bit ten zostaje skasowany sprzętowo, po uaktualnieniu OCR0 z rejestru tymczasowych wartości.
Zero logiczne w tym bicie wskazuje gotowość OCR0 do zapisania nowej wartości.

• Bit 0 – TCR0UB: Rejestr0 sterujący Zegar/Licznik Update Busy

Bit ten zostaje ustawiony gdy Zegar/Licznik pracuje asynchronicznie i TCCR0 jest zapisywany.
Bit ten zostaje skasowany sprzętowo, po uaktualnieniu TCCR0 z rejestru tymczasowych wartości.
Zero logiczne w tym bicie wskazuje gotowość TCCR0 do zapisania nowej wartości.

Podczas odczytu TCNT0, pobierana jest rzeczywista wartość zegara. Podczas odczytu OCR0 i TCCR0, pobierana jest zawartość rejestru tymczasowych wartości

Rejestr maski przerwań jednostki Zegar/Licznik – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

• Bit 1 – OCIE0: Włączenie przerwania dopasowania Zegar/Licznik0

Gdy bit OCIE0 jest zapisany jedynką, i bit I rejestru stanu jest ustawiony (jedynka), zostaje włączone przerwanie dopasowania Zegar/Licznik0. Odpowiednie przerwanie jest wykonywane jeśli zajdzie dopasowanie w Zegar/Licznik0.

• Bit 0 – TOIE0: Zegar/Licznik0 Overflow Interrupt Enable

Gdy bit TOIE0 jest zapisany jedynką, i bit I rejestru stanu jest ustawiony (jedynka), zostaje włączone przerwanie przepełnienia Zegar/Licznk0. Odpowiednie przerwanie jest wykonywane jeśli nastąpi przepełnienie w Zegar/Licznik0.

REJESTR FLAG PRZERWAŃ LICZNIKA – TIFR

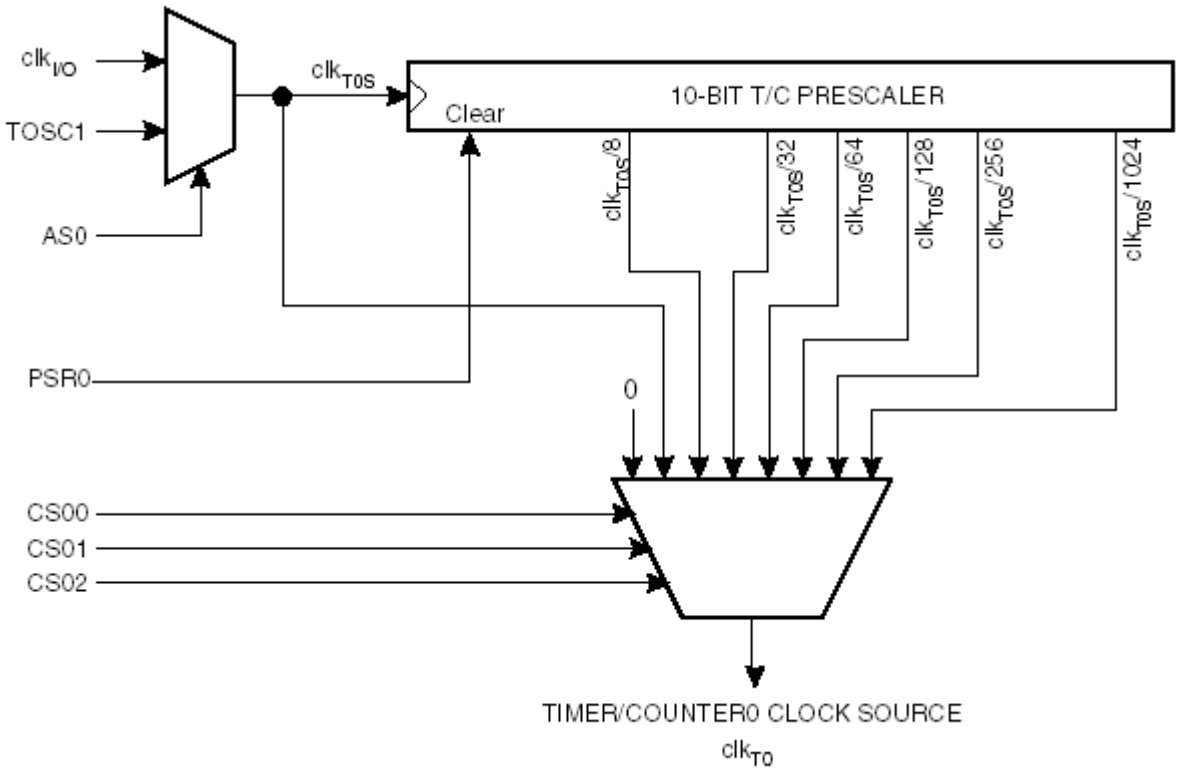
Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OC1B	TOV1	OCF0	TOV0	TIFR
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

• Bit 1 – OCF0: Porównanie wyjścia - Flaga 0
Bit OCF0 jest ustawiany (jedynka) gdy zajdzie dopasowanie pomiędzy Zegar/Licznik0 i danymi w OCR0. OCF0 jest kasowany sprzętowo po wykonaniu odpowiedniego podprogramu obsługi przerwania. OCF0 można także skasować przez zapis jedynki logicznej do flagi. Gdy bit I w SREG, OCIE0, i OCF0 są ustawione (jedynka), zostaje obsłużone przerwanie dopasowania Zegar/Licznik0.

• Bit 0 – TOV0: Timer/Counter0 Overflow Flag
Bit TOV0 jest ustawiany gdy nastąpi przepełnienie w Zegar/Licznik0. jest kasowany sprzętowo po wykonaniu odpowiedniego podprogramu obsługi przerwania. TOV0 można także skasować poprzez zapis jedynki logicznej. Gdy bit I w SREG, TOIE0, i TOV0 są ustawione (jedynka), zostaje obsłużone przerwanie przepełnienia Zegar/Licznik0. W trybie PWM, bit ten jest ustawiony gdy Zegar/Licznik0 zmienia kierunek przy $00.

Zegar/Licznik Prescaler

Rys. 44. Prescaler dla Zegar/Licznik0



Źródło zegarowe dla Zegar/Licznik0 oznaczane jest jako clkT0. clkT0 jest domyślnie podłączone do głównego zegara clkI/O. Poprzez ustawienie bitu AS0 w ASSR, następuje asynchroniczne taktowanie Zegar/Licznik0 ze styku TOSC1. W ten oto sposób Zegar/Licznik0 pracuje jak licznik czasu rzeczywistego (RTC). Gdy bit AS0 jest ustawiony, styki TOSC1 i TOSC2 zostają odłączone od portu C. Możliwe konfiguracje dla Zegar/Licznik0 z prescalerem: clkT0S/8, clkT0S/32, clkT0S/64, clkT0S/128, clkT0S/256, i clkT0S/1024. Dodatkowo można wybrać clkT0S jako 0 (stop). Ustawienie bitu PSR0 w SFIOR kasuje prescaler.

Specjalne funkcje rejestru IO – SFIOR

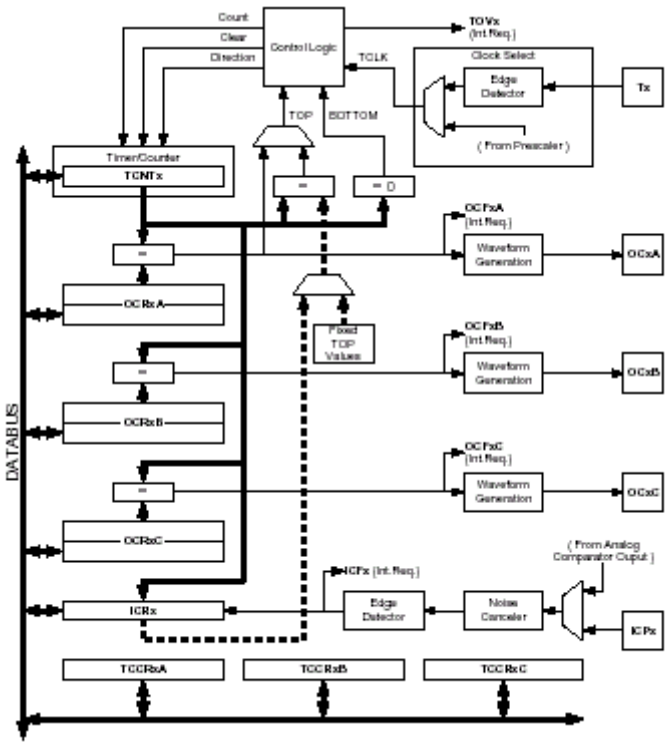
Bit	7	6	5	4	3	2	1	0	
	TSM	-	-	ADHSM	ACME	PUD	PSR0	PSR231	SFIOR
Odczyt/Zapis (R/W)	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

• Bit 7 – TSM: Tryb synchronizacji jednostki Zegar/Licznik
Zapis jedynki do TSM, spowoduje trzymanie wartości rejestrów PSR0 i PSR321 aż do momentu ponownego zapisania , lub zapisu zera w TSM. Tryb ten jest użyteczny dla synchronizacji liczników.

• Bit 1 – PSR0: Kasowanie prescaler Zegar/Licznik0
Prescaler zostaje zresetowany, gdy bit 1 jest zapisywany jedynką. Zapis zera do tego bitu nie spowoduje żadnej akcji. Bit ten zostanie zawsze odczytany jako zero jeśli Zegar/Licznik0 jest taktowany z wewnętrznego zegara CPU. Jeśli ten bit jest zapisywany podczas asynchronicznej pracy jednostki Zegar/Licznik0, pozostanie on jedynką aż do zresetowania prescalera.

16-BIT TIMER/LICZNIK (TIMER/LICZNIK1 I TIMER/LICZNIK3)

Szesnastobitowa jednostka timera/licznika pozwala na dokładną kontrolę parametrów czasowych związanych z wykonywaniem instrukcji (zarządzanie zdarzeniami), generację przebiegów oraz pomiary parametrów czasowych sygnałów. Główne jej cechy to:
- Pełna zgodność z architekturami 16-bitowymi
- Trzy niezależne jednostki porównywania wyjść
- Podwójnie buforowane rejestry porównywania wyjść
- Jedna jednostka kontrolna wejścia
- Reduktor szumów jednostki kontrolnej wejścia
- Timer z zerowaniem przy zgodności z zadanym warunkiem (samoprzeładowanie)
- Bezhazardowy modulator szerokości impulsu z poprawną fazą
- Zmienny okres PWM
- Generator
- Licznik zdarzeń zewnętrznych
- 10 niezależnych źródeł przerwań



REJESTRY

Timer/Licznik (TCNTn), wyjściowe rejestry porównania (OCRnA/B/C), oraz Wejściowy rejestr zdarzeń (ICRn) są rejestrami 16-bitowymi. Rejestry kontrolne licznika (TCCRnA/B/C) są rejestrami 8-bitowymi i nie nakładają żadnych ograniczeń dostępu na procesor. Wszystkie sygnały żądania przerwań są widoczne w rejestrze przerwań TIFR w rozszerzonym rejestrze przerwań ETIFR.
Podwójnie buforowane wyjściowe rejestry porównania (OCRnA/B/C) są cały czas porównywane z wartością licznika. Wynik porównania może być użyty przez generator fal.

KOMPATYBILNOŚĆ

Omawiany licznik jest nową ulepszoną wersją 16-bitowego licznika AVR. Jest on w pełni kompatybilny z poprzednią wersją jeśli chodzi o:
- Wszystkie lokacje adresowe 16-bitowego rejestru I/O, włącznie z rejestrem przewań
- Lokacje bitów we wszystkich 16-bitowych rejestrach
- Wektory przerwań

Następujące kontrolne bity zmieniły nazwę, lecz posiadają te same funkcje i lokacje:
- PWMn0 zmieniony na WGMn0
- PWMn1 zmieniony na WGMn1
- CTCn zmieniony na WGMn2

Następujące rejestry zostały dodane:
- Rejestr kontrolny C (TCCRnC)
- Rejestry OCRnCH, OCRnCL

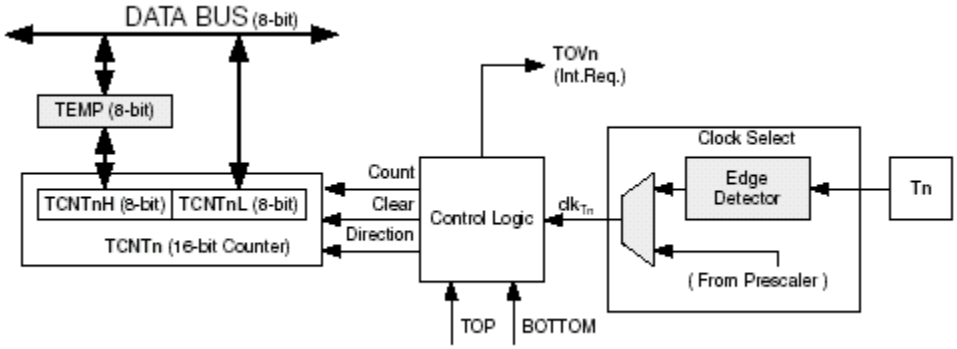
Następujące bity zostały dodane:
- COM1C1:0 dodany do rejestru TCCR1A
- FOCnA, FOCnB, FOCnC dodane do nowego rejestru TCCRnC.

ŹRÓDŁA TAKTOWANIA

Timer/Licznik może być taktowany przez wewnętrzne lub zewnętrzne źródło. Wyboru dokonuje się poprzez ustawienie bitów CSn2:0 zlokalizowanych w rejestrze kontrolnym B (TCCRnB).

Jednostka Licząca

Główną częścią 16-bitowego Licznika/Zegara jest programowalna 16-bitowa dwukierunkowa jednostka licząca.



Count zwiększenie lub zmniejszenie TCNTn o 1.
Direction Wybór pomiędzy inkrementacją i dekrementacją.
Clear Wyzerowanie TCNTn (ustawienie wszystkich bitów na zero).
clkTn Impuls zegara/licznika.
TOP Sygnalizuje że TCNTn osiągnął wartość maksymalną.
BOTTOM Sygnalizuje że TCNTn osiągnął wartość minimalną.

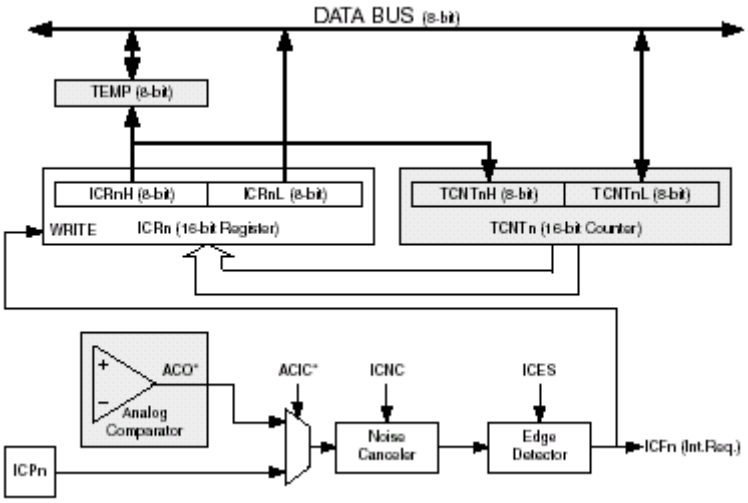
16-bitowy licznik jest odwzorowany w dwie 8-bitowe I/O lokacje pamięci: TCNTnH zawiera wyższe 8 bitów licznika, TCNTnL zawiera 8 bitów niższych. Procesor ma pośredni dostęp do rejestru TCNTnH. Kiedy zawartość TCNTnL jest czytana to do rejestru tymczasowego (TEMP) zapisywana jest zawartość TCNTnH natomiast zawartość TCNTnH jest aktualizowana zawartością rejestru tymczasowego kiedy wartość TCNTnL jest zapisywana. To pozwala procesorowi na zapisywanie i odczytywanie całego 16-bitowego licznika w jednym cyklu zegara. Należy odnotować, że występują szczególne przypadki zapisywania do rejestru TCNTn kiedy licznik jest w trakcie liczenia. Daje to nieprzewidywalne wyniki.

W zależności od wybranego trybu licznik jest czyszczony, inkrementowany lub dekrementowany za każdym impulsem zegara. Impulsy mogą być generowane z zewnętrznego lub zewnętrznego źródła w zależności od ustawienia bitów wyboru zegara(Clock Select, CSn2:0). Jeżeli żadne źródło nie jest wybrane (CSn2:0=0) licznik jest zatrzymany. Procesor w takim wypadku ma dostęp do zawartości TCNTn ponieważ operacja zapisu przez procesor ma wyższy priorytet niż operacje liczenia i czyszczenia licznika.

Sekwencja liczenia jest określona przez ustawiania odpowiednich bitów (Waveform Generation, WGMn3:0) zlokalizowanych w rejestrach kontrolnych A i B (TCCRnA , TCCRnB).
Istnieje bliski związek między tym jak licznik się zachowuje (jak liczy) a tym jakie kształty fal generowane są na wyjściu OCnx.

Jednostka Przechwytująca Sygnały Wejściowe

Licznik zawiera wejściową jednostkę, która potrafi przechwytywać zewnętrzne zdarzenia i nadawać im znacznik czasowy wskazujący czas wystąpienia. Zewnętrzny sygnał wskazujący na zdarzenie, lub grupę zdarzeń może być stosowany przez pin ICPn lub alternatywnie przez Analogowy Komparator. Znacznik czasowy może być potem użyty do zliczenia częstotliwości i innych właściwości sygnału. Alternatywnie znacznik czasowy może być wykorzystany do stworzenia dziennika zdarzeń.



Jeżeli na wejściu powyższego układu (ICPn) lub na wyjściu analogowego komparatora (ACO) występuje zmiana poziomu logicznego i ta zmiana jest zgodna z ustawieniami wykrywacza zbocza, dane zostaną zatrzaśnięte. W takim przypadku 16-bitowa wartość licznika (TCNTn) jest zapisywana do wejściowego rejestru (ICRn). Wejściowy wskaźnik stanu (ICFn) jest ustawiany w tym samym takcie w którym zawartość TCNTn jest kopiowana do rejestru ICRn. Wejściowy wskaźnik stanu generuje przerwanie. Po wygenerowaniu przerwania wskaźnik jest automatycznie czyszczony.

Czytanie 16-bitowej wartości rejestru wejściowego (ICRn) odbywa się w dwóch etapach. Najpierw czytane są młodsze bajty (ICRnL) a następnie starsze (ICRnH). Gdy młodsze bajty są czytane, starsze są kopiowane do rejestru tymczasowego (TEMP).

Rejestr ICRn może być czytany tylko gdy tryb generacji fali jest włączony. Tryb ten wykorzystuje rejestr ICRn dla zdefiniowania maksymalnej wartości licznika. W tym przypadku bit trybu generacji fali (WGM3:0) musi być ustawiony zanim maksymalna wartość licznika zostanie zapisana do rejestru ICRn. Kiedy rejestr ICRn jest zapisywany, to starszy bajt musi zostać zapisany do lokacji wejścia/wyjścia rejestru ICRnH zanim młodszy bajt zostanie zapisany do rejestru ICRnL.

Źródło wyzwalania dla jednostki kontrolnej wejścia

Głównym źródłem wyzwalania dla jednostki kontrolnej wejścia jest pin ICPn. Timer/Licznik1 może używać w zamian także wyjścia komparatora analogowego jako alternatywnego źródła wyzwalania. Komparator analogowy jest źródłem sygnału wyzwalania gdy ustawiony jest bit ACIC w rejestrze ACSR. Istnieje możliwość niekontrolowanego wyzwolenia i przypadkowego zatrzaśnięcia danych wejściowych w

jednostce kontrolnej wejścia. Flaga zatrzaśnięcia wejścia musi być więc wyzerowana po takiej zmianie. Oba piny- ICPn i ACO są próbkowane przy użyciu tym samym sposobem jak i piny Tn . Detektor zbocza jest więc identyczny. Jednakże, gdy włączony jest reduktor szumów dołączana jest dodatkowa ilość logiki przed detektorem zbocza która powoduje wzrost opóźnienia o 4 cykle zegara systemowego. Należy zauważyć, że wejście reduktora szumu i detektor zbocza są zawsze włączone dopóki Timer/Licznik jest ustawiony w trybie generacji przebiegów który używa ICRn do definiowania TOP. Zatrzaśnięcie wejścia może być wyzwolone programowo za pomocą portu związanego z pinem ICPn.

Reduktor Zakłóceń

Reduktor Zakłóceń zwiększa odporność układu na zakłócenia, poprzez zastosowanie prostego układu filtracji cyfrowej. Wejście reduktora jest monitorowane przez cztery próbki, i wszystkie cztery muszą być jednakowe żeby nastąpiła zmiana wyjścia.
Reduktor Zakłóceń jest włączany przez ustawienie wejściowego bitu redukcji szumów (ICNCn) w rejestrze kontrolnym licznika (TCCRnB). Kiedy redukcja szumów jest włączona wprowadza dodatkowe cztery cykle zegarowe. Reduktor używa zegara systemowego i dlatego nie ma na niego wpływu przelicznik wstępny.

Używanie jednostki przechwytywania sygnałów wejściowych

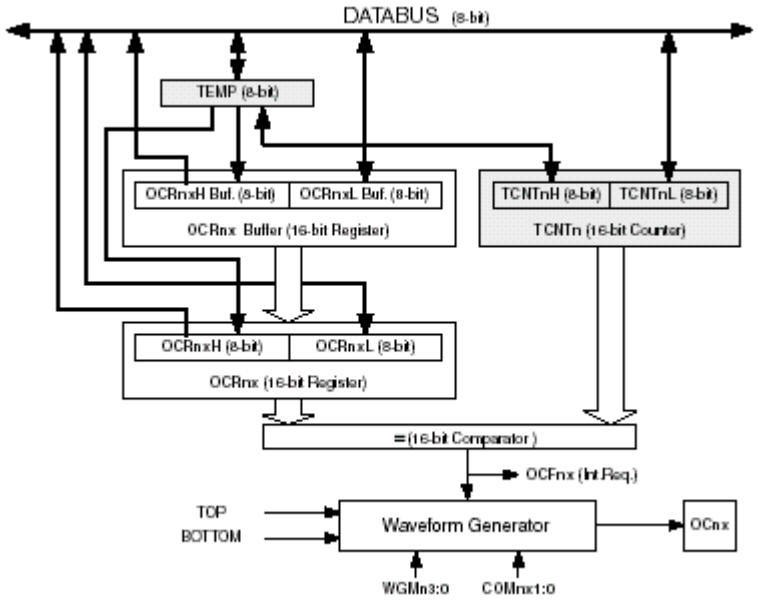
Głównym zadaniem przy używaniu jednostki przechwytywania sygnałów wejściowych jest określenie wystarczającej mocy obliczeniowej procesora, żeby był on w stanie obsłużyć napływające zdarzenia. Czas pomiędzy dwoma zdarzeniami jest krytyczny. Jeśli procesor nie przeczytał przechwyconej wartości z rejestru ICRn przed następnym zdarzeniem, zawartość rejestru ICRn zostanie zamazana przez nową wartość. W tym przypadku wynik operacji będzie nieprawidłowy.

Kiedy występuje przerwanie (po wychwyceniu zdarzenia na wejściu)**,** rejestr ICRn powinien zostać przeczytany przez podprogram obsługi przerwań, tak szybko jak to tylko możliwe. Chociaż to przerwanie posiada względnie wysoki priorytet, to maksymalny czas reakcji jest uzależniony od maksymalnej liczby cykli zegarowych potrzebnych do obsługi każdego innego zgłoszenia przerwania.

Używanie omawianej jednostki w dowolnym trybie, kiedy wartość maksymalna licznika (TOP) jest zmieniana na bieżąco podczas operacji nie jest wskazane.
Pomiar zewnętrznego sygnału cyklu pracy wymaga, aby zbocze wyzwalające było zmieniane po każdym wychwyceniu zdarzenia. Zmiana odczytu zbocza musi być wykonana tak wcześnie jak to możliwe, po tym jak rejestr ICRn został odczytany. Po zmianie zbocza, flaga (ICFn) musi zostać wyczyszczona programowo (zapisanie logicznej jedynki to bitu lokacji I/O). Dla pomiaru tylko częstotliwości, czyszczenie flagi ICFn nie jest konieczne.

KOMPARATOR WYJŚCIOWY

16-btowy komparator porównuje zawartość licznika (TCNTn) z zawartością rejestru OCRnx. Jeżeli zawartości są takie same to komparator sygnalizuje to poprzez ustawienie odpowiedniej flagi (OCFnx) w następnym cyklu zegarowym. powoduje to wygenerowanie przerwania. Flaga OCFnx jest automatycznie czyszczona po wykonaniu przerwania. Generator falowy używa sygnału porównania do generowania sygnału wyjściowego zgodnie z trybem współpracy określonym przez bity tryb generacji fali (WGMn3:0) i porównania (COMnx1:0).
Specjalna cecha jednostki A pozwala na zdefiniowanie maksymalnej wartości (TOP) licznika. Wartość ta określa okres generowanych fali.



Rejestr OCRnx jest podwójnie buforowany kiedy używany jest dowolny z dwunastu trybów modulacji szerokości impulsu (PWM). Dla normalnego trybu operacji i dla trybu czyszczenia zegara przy porównaniu (CTC) podwójne buforowanie jest wyłączone. Podwójne buforowanie synchronizuje aktualizacje zawartości rejestru OCRnx każdej górnej (TOP) albo dolnej (BOTTOM) sekwencji liczenia. Synchronizacja zapobiega występowania niesymetrycznych impulsów PWM, czyniąc wyjście wolne od zakłóceń.

Dostęp do rejestru OCRnx jest skomplikowany, ale nie przez przypadek. Jeżeli podwójne buforowanie jest włączone, procesor ma dostęp do rejestru buforowego OCRnx. Gdy podwójne buforowanie jest wyłączone, procesor ma bezpośredni dostęp do rejestru OCRnx. Zawartość rejestru OCR1x może być zmieniona tylko przez operację zapisu (Licznik nie aktualizuje rejestru automatycznie jak rejestrów TCNTn oraz ICRn). Rejestr OCRnx nie jest czytany za pośrednictwem rejestru tymczasowego (TEMP). Jednak dobrym zwyczajem jest odczytanie najpierw niskich bajtów, tak jak przy dostępie do innych 16-bitowych rejestrów. Zapisywanie rejestru OCRnx musi odbywać się przez rejestr tymczasowy (TEMP). Wyższe bajty (OCRnxH) muszą zostać zapisane najpierw. Kiedy lokacja I/O jest zapisana przez procesor, rejestr TEMP zostanie zaktualizowany zapisaną wartością. Następnie gdy zawartość OCRnxL jest zapisana do niskich 8 bitów, wysokie bajty zostaną skopiowane do wyższych 8 bitów bufora lub rejestru OCRnx w tym samym cyklu zegarowym.

Wymuszenie dopasowania na wyjściu

W trybie generacji fal przy wyłączonej modulacji szerokości impulsu, dopasowanie na wyjściu komparatora może zostać wymuszone przez zapisanie jedynki do bitu wymuszenia porównania na wyjściu (FOCnx). Nie spowoduje to ustawienia flagi OCFnx ani przeładowania czy wyczyszczenia zegara. Wszystkie procesory zapisując do rejestru TCNTn będą blokować każde sprawdzenie dopasowania, które nastąpi w następnym cyklu zegarowym, nawet gdy zegar jest zatrzymany. Ta cech pozwala rejestrowi OCRnx być inicjalizowanym tą samą wartością co licznik TCNTn bez wyzwolenia przerwania.

Zastosowanie Komparatora

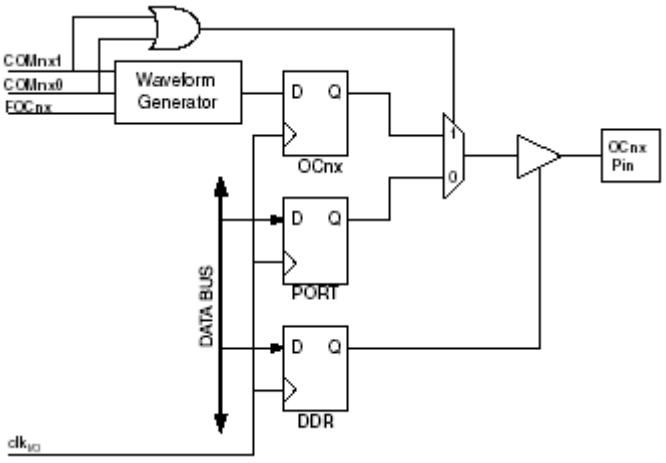
Fakt, że zapisywanie do licznika (TCNTn) blokuje wszystkie operacje porównania na czas jednego cyklu zegarowego, sprawia że występuje ryzyko wystąpienia błędu. Jeżeli wartość zapisana do TCNTn jest równa wartości OCRnx, porównanie nie nastąpi i w rezultacie wystąpi nieprawidłowa generacja fali.

Nie wolno zapisywać jednocześnie zawartości licznika (TCNTn) oraz wartości maksymalnej (TOP) w trybie PWM. Sprawdzenie dopasowania dla wartości maksymalnej zostanie zignorowane i licznik będzie kontynuował pracę do wartości 0xFFFF. Podobnie nie można zapisywać jednocześnie TCNTn oraz wartości minimalnej (BOTTOM) kiedy licznik liczy w dół.

Trzeba zwrócić uwagę na to, że bity COMnx1:0 nie są podwójnie buforowane. Efekt Zmiany tych bitów będzie natychmiastowy.

Jednostka porównująca dopasowanie na wyjściu

Bity trybu porównania wyjścia (COMnx1:0) posiadają dwie funkcje. Generator falowy używa bitów COMnx1:0 dla zdefiniowania stanu porównania wyjścia (OCnx) przy następnym porównaniu dopasowania. Po drugie bity COMnx1:0 kontrolują źródło wyjściowe OCnx.



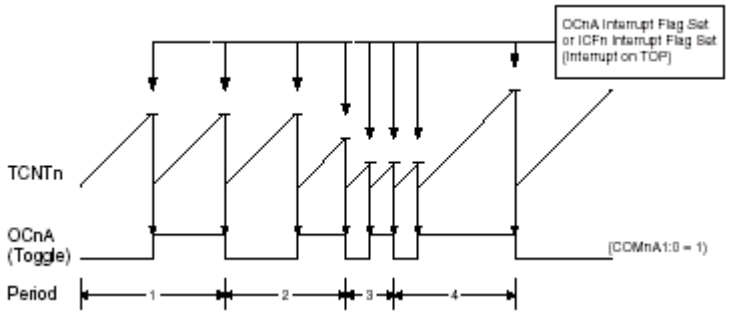
Generator falowy używa bitów COMnx1:0 różnie w trybie normalnym, CTC i PWM. We wszystkich trybach ustawienie COMnx1:0=0 informuje generator, że żadne działanie na rejestrze OCnx nie będzie podjęte przy następnym dopasowaniu.
Zmiana bitów COMnx1:0 będzie miała efekt przy pierwszym dopasowaniu po zapisaniu bitów.
W trybie CTC i normalnym można wymusić natychmiastowy efekt przez użycie strobujących bitów FOCnx.

Tryby Operacji

Tryb operacji, czyli zachowanie Licznika i pinów porównania wyjścia, jest zdefiniowane przez kombinację bitów trybu generacji fal (WGM3:0) i trybu porównania wyjścia (COMnx1:0). Bity trybu porównania wyjścia nie mają wpływu na sekwencję liczenia, dopóki określają to bity trybu generacji fali. Bity COMnx1:0 określają, czy generowany sygnał wyjściowy powinien być odwrócony czy nie. Dla trybów różnych od PWM bity COMnx1:0 czy wyjście powinno być ustawione, czyszczone czy przełączone przy porównaniu.

Tryb Normalny
Najprostszym trybem operacji jest tryb normalny (WGM3:0=0). W tym trybie licznik zawsze liczy w górę, i nie jest wykonywane czyszczenie licznika. Licznik gdy osiąga swoją maksymalną wartość (MAX=0xFFFF) zaczyna liczyć od wartości minimalnej (BOTTOM=0x0000). W tym trybie flaga przekroczenia zakresu (*Timer/Counter Overflow Flag*- TOVn) zostanie ustawiona w tym samym cyklu zegarowym, w którym wartość TCNTn staje się zerem. W tym przypadku flaga TOVn zachowuje się jak siedemnasty bit, z wyjątkiem tego że jest tylko ustawiana, nie czyszczona.
Nie ma w tym trybie żadnych specjalnych przypadków do rozpatrywania. Nowa wartość licznika może być zapisana w każdej chwili. Jednostka przechwytująca sygnały wejściowe jest łatwa do użycia w trybie normalnym. Komparator wyjściowy może być użyty do generowania przerwań, nie zaleca się stosowania do generacji fal ponieważ wtedy za bardzo obciążony jest procesor.

Tryb CTC (czyszczenia zegara)
W trybie CTC (WGMn3:0=4 lub 12) rejestr OCRnA lub ICRn jest używany do manipulowania rozdzielczością licznika. W trybie CTC licznik jest czyszczony do zera kiedy wartość licznika (TCNTn) jest zgodna z rejestrem OCRnA (WGM3:0=4) lub z rejestrem ICRn (WGMn3:0=12). Rejestry te mogą określać maksymalną wartość licznika, a zatem i jego rozdzielczość. Ten tryb pozwala na większą kontrolę nad dopasowaniem częstotliwości na wyjściu. Upraszcza on także operacje liczenia zewnętrznych zdarzeń.
Na rysunku pokazano wykres czasowy dla trybu CTC. Wartość licznika (TCNTn) jest zwiększana dopóki nie wystąpi zgodność z OCRnA lub ICRn. Wtedy następuje czyszczenie licznika.



Przerwanie może zostać wygenerowane, za każdym razem gdy licznik osiągnie wartość maksymalną (TOP) używając flagi ICFn lub OCFnA zgodnie z rejestrem używanym do zdefiniowania wartości maksymalnej. Jeżeli przerwanie jest dostępne, podprogram obsługi przerwań może zostać użyty do odświeżania wartości TOP. Jednak zmienianie wartości maksymalnej na bliską wartości minimalnej podczas pracy licznika należy robić bardzo ostrożnie jeżeli tryb CTC nie jest podwójnie buforowany. Jeżeli nowa wartość maksymalna jest mniejsza od bieżącej wartości licznika, to licznik tego nie wychwyci i będzie liczył do swojej maksymalnej wartości (0xFFFF) i zacznie liczyć od zera. W wielu przypadkach jest to niepożądane. Rozwiązaniem jest użycie szybkiego trybu PWM, używając OCRnA dla zdefiniowania wartości TOP (WGMn3:0=15). OCRnA będzie wtedy podwójnie buforowany.

Dla generowania fali na wyjściu w trybie CTC, wyjście OCnA może być ustawione na zmianę swojego stanu logicznego przy każdym dopasowaniu. Robi się to ustawiając bity trybu porównania wyjścia na tryb zmiany (COMnA1:0=1). Generowana fala będzie miała maksymalną wartość częstotliwości gdy OCRnA jest ustawiony na zero (0x0000):

$$f_{OCnA} = f_{clk\_I/O}/2$$

Częstotliwość generowanej fali jest określona następującym wzorem:

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

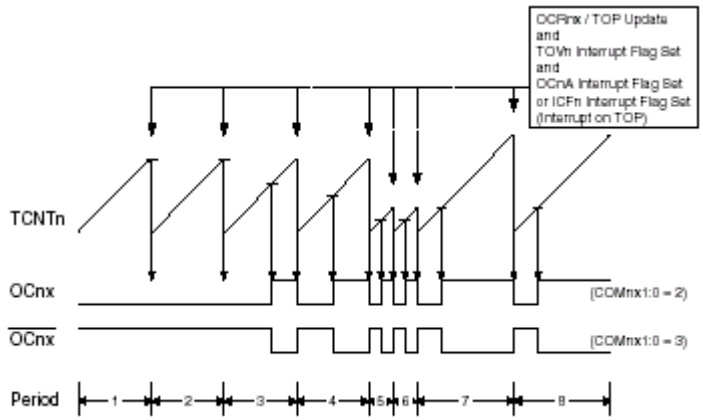
Współczynnik N przyjmuje wartości 1,8,64,256,1024.
Tak jak w trybie normalnym flaga TOV jest ustawiana w tym samym cyklu zegarowym, w którym licznik przechodzi z wartości MAX do 0x0000.

Szybki tryb Modulacji Szerokości Impulsu(Szybki tryb PWM)
Szybki tryb PWM (WGMn3:0=5,6,7,14,15) zapewnia opcję generacji fal wysokiej częstotliwości. Szybki tryb różni się od innych trybów PWM operacją na pojedynczym zboczu.
Licznik liczy w pętli od wartości BOTTOM do wartości TOP. W nieodwracającym trybie porównania wyjścia, wyjście OCnx jest ustawiane przy porównaniu rejestru OCRnx i TCNTn, i jest czyszczone przy wartości maksymalnej (TOP). W trybie odwracającym jest czyszczone przy porównaniu i ustawiane przy TOP. Przez operacje wykonywane na pojedynczym zboczu częstotliwość pracy w szybkim trybie PWM może być dwa razy większa niż przy operacjach wykonywanych na podwójnym zboczu. Wysoka częstotliwość czyni szybki tryb PWM odpowiedni do regulacji zasilania, prostowania, przetworników analogowo-cyfrowych.

Minimalna rozdzielczość trybu PWM to 2-bity (ICRn lub OCRnA ustawione na 0x0003), a maksymalna rozdzielczość to 16-bitów (ICRn lub OCRnA ustawione na MAX). Według poniższego wzoru można określić rozdzielczość w bitach:

$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

W szybkim trybie PWM licznik jest inkrementowany dopóki wartość licznika nie jest zgodna z jedną z ustalonych wartości: 0x00FF, 0x01FF, 0x03FF (WGMn3:0=5,6,7) lub z wartością rejestru ICRn (WGMn3:0=14), lub też z wartością rejestru OCRnA (WGMn3:0=15). Licznik jest wtedy czyszczony w następnym cyklu zegarowym. Na rysunku pokazano przebieg czasowy dla szybkiego trybu PWM.



Flaga przepełnienia licznika TOVn jest ustawiana za każdym razem gdy licznik osiąga wartość TOP. W dodatku flaga ICnA lub ICFn jest ustawiana w tym samym cyklu zegarowym.
Jeśli jedno z przerwań jest włączone, podprogram obsługi przerwań może być użyty do aktualizowania wartości TOP.

Podczas zmiany wartości TOP program musi być pewny, że nowa wartość jest równa lub większa od zawartości wszystkich rejestrów OCRnx. Jeśli wartość TOP jest od któregoś rejestru mniejsza, wtedy nigdy nie nastąpi przypadek w którym licznik TCNTn i rejestr OCRnx będą miały tą samą wartość.

Rejestr ICRn nie jest podwójnie buforowany. Znaczy to, że jeśli zawartość ICRn jest zmieniona na niską wartość podczas pracy licznika bez przeliczania wstępnego, istnieje ryzyko, że nowa wartość ICRn jest niższa od aktualnej wartości TCNTn.
W rezultacie licznik nie wykona porównania przy wartości maksymalnej(TOP). Licznik będzie wtedy musiał doliczyć do wartości 0xFFFF i zacząć od zera.
Rejestr OCRnA jest podwójnie buforowany. Pozwala to na zapisanie lokacji I/O rejestru OCRnA w dowolnym momencie. Kiedy tak się stanie, to zapisana wartość zostanie umieszczona w buforowym rejestrze OCRnA. Rejestr OCRnA zostanie zaktualizowany nową wartością z rejestru buforowego w następnym cyklu zegarowym gdy TCNTn osiągnie wartość TOP. Aktualizacja nastąpi w tym samym cyklu, w którym TCNTn jest czyszczony i flaga TOVn jest ustawiona.
Użycie rejestru ICRn dla zdefiniowania wartości TOP działa dobrze dla ustalonych wartości. Jednak użycie rejestru OCRnA jako wartości TOP jest lepszym rozwiązaniem ze względu na podwójne buforowanie.
W szybkim trybie PWM, możliwe jest generowanie fal na pinie OCnx. Ustawienie bitów COMnx1:0 na wartość 2 spowoduje generacje fal nieodwróconych. Generacja fal odwróconych nastąpi przy ustawieniu COMnx1:0 na wartość 3. Częstotliwość można obliczyć według następującej zależności:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

Współczynnik N przyjmuje wartości 1,8,64,256,1024.

Graniczne wartości rejestru OCRnx określają specjalne zdarzenia podczas generacji w szybkim trybie PWM. Jeśli OCRnx jest ustawiony na wartość BOTTOM, to na wyjściu pojawi się ostry impuls dla każdego TOP+1-szego cyklu zegara. Ustawienie rejestru OCRnx na wartość TOP, spowoduje pojawienie

się na wyjściu stanu wysokiego lub niskiego (w zależności od polaryzacji wyjścia ustawionej na bitach COMnx1:0).

Generowane fale mają maksymalną częstotliwość:

$$f_{OCnA}=f_{clk\_I/O}/2$$

gdy rejestr OCRnA jest ustawiony na zero (0x0000).

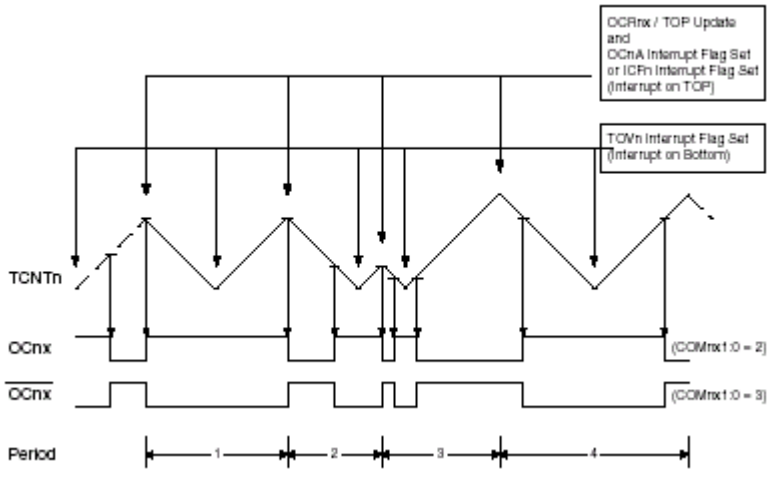
Tryb Modulacji Szerokości Impulsu z Korekcją Fazy

Tryb modulacji szerokości impulsu z korekcją fazy (WGMn3:0=1,2,3,10,11) zapewnia opcje generowania fal o wysokiej rozdzielczości. Ten tryb podobnie jak tryb PWM z korekcją fazy i częstotliwości bazuje na operacjach wykonywanych na podwójnym zboczu. Licznik liczy od wartości BOTTOM (0x0000) do wartości TOP, i następnie od TOP do BOTTOM. W nieodwracającym trybie porównania wyjścia, OCnx jest czyszczone przy dopasowaniu pomiędzy TCNTn i OCRnx podczas liczenia w górę. OCnx jest ustawiane przy dopasowaniu podczas liczenia w dół.
Rozdzielczość modulacji szerokości impulsu w tym trybie może być ustawiona na 8-,9-,10-bit, lub może być określona przez ICRn lub OCRnA. Minimalna rozdzielczość to 2 bity (ICRn lub OCRnA ustawione na 0x0003), a maksymalna to 16 bit (ICRn lub OCRnA ustawione na MAX).
Rozdzielczość w bitach wyrażona jest następującym wzorem:

$$R_{PCPWM} = \frac{\log(TOP+1)}{\log(2)}$$

W omawianym trybie licznik jest inkrementowany dopóki nie osiągnie jednej z wartości: 0x00FF, 0x01FF, 0x02FF (WGMn3:0=1,2,3), wartości w rejestrze ICRn(WGMn3:0=10) lub wartości OCRnA (WGMn3:0=11). Wtedy licznik osiągnął wartość TOP i zmienia kierunek liczenia. Wartość TCNTn będzie równa TOP na jeden cykl zegarowy.

Poniższy przebieg przedstawia tryb PWM z korekcją fazy w sytuacji gdy OCRnA oraz ICRn są używane do zdefiniowania wartości TOP.



Flaga przepełnienia licznika TOVn jest ustawiana za każdym razem gdy licznik osiąga wartość minimalną. Gdy rejestr OCRnA lub ICRn jest używany do zdefiniowania wartości TOP, flaga ICF lub OCnA jest ustawiana odpowiednio w tym samym cyklu w którym rejestr OCRnx jest aktualizowany. Flaga przerwania może zostać wykorzystana do wygenerowania przerwania za każdym razem gdy licznik osiągnie wartość TOP lub BOTTOM.

Podczas zmiany wartości TOP program musi być pewny, że nowa wartość jest równa lub większa od zawartości wszystkich rejestrów OCRnx. Jeśli wartość TOP jest od któregoś rejestru mniejsza, wtedy nigdy nie nastąpi przypadek w którym licznik TCNTn i rejestr OCRnx będą miały tą samą wartość.

Zalecane jest używanie trybu z korekcją fazy i częstotliwości zamiast trybu z korekcją fazy gdy wartość TOP jest zmieniana w czasie pracy licznika. Kiedy używamy stałej wartości TOP nie praktycznie różnicy między tymi trybami.

W trybie PWM z korekcją fazy możliwe jest generowanie fali na pinie OCnx. Częstotliwość na wyjściu w tym trybie można obliczyć według wzoru:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

Współczynnik N przyjmuje wartości 1,8,64,256,1024.

Graniczne wartości rejestru OCRnx reprezentują specjalne przypadki podczas generacji w trybie PWM z korekcją fazy. Jeśli OCRnx jest ustawiona na wartość BOTTOM to na wyjściu pojawi się stan niski, a jeśli rejestr jest ustawiony na TOP to na wyjściu pojawi się stan wysoki. W trybie inwersyjnym wyjście będzie miało przeciwne logicznie wartości.

Tryb modulacji szerokości impulsu z korekcją fazy i częstotliwości
Tryb modulacji szerokości impulsu z korekcją fazy i częstotliwości (WGMn3:0=8, 9) zapewnia opcję generowania fali wysokiej częstotliwości. Ten tryb podobnie jak tryb PWM z korekcją fazy bazuje na operacjach wykonywanych na podwójnym zboczu. Licznik liczy od wartości BOTTOM (0x0000) do wartości TOP, i następnie od TOP do BOTTOM. W nieodwracającym trybie porównania wyjścia, OCnx jest czyszczone przy dopasowaniu pomiędzy TCNTn i OCRnx podczas liczenia w górę. OCnx jest ustawiane przy dopasowaniu podczas liczenia w dół.

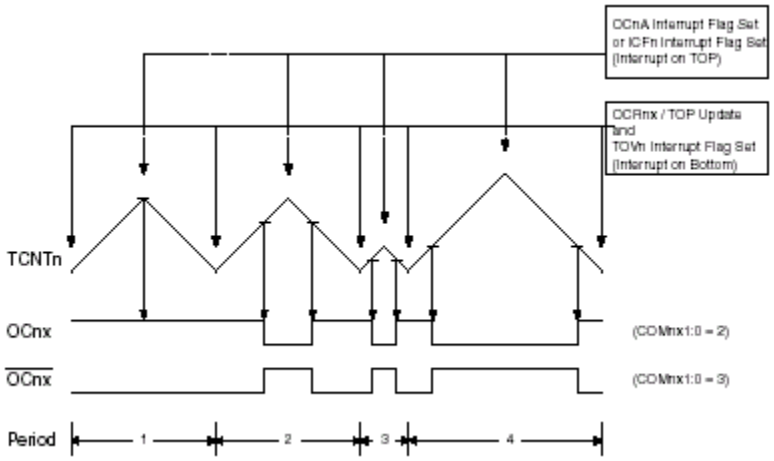
Główną różnicą pomiędzy trybem PWM z korekcją fazy a trybem PWM z korekcją fazy i częstotliwości jest czas w którym rejestr OCRnx jest aktualizowany przez rejestr buforowy.

Rozdzielczość modulacji szerokości impulsu w tym trybie może być określona przez ICRn lub OCRnA. Minimalna rozdzielczość to 2 bity (ICRn lub OCRnA ustawione na 0x0003), a maksymalna to 16 bit (ICRn lub OCRnA ustawione na MAX). Rozdzielczość w bitach wyrażona jest następującym wzorem:

$$R_{PFCPWM} = \frac{\log(TOP+1)}{\log(2)}$$

W szybkim trybie PWM licznik jest inkrementowany dopóki wartość licznika nie jest zgodna z jedną z wartości rejestru ICRn (WGMn3:0=8), lub też z wartością rejestru OCRnA (WGMn3:0=9). Licznik osiąga wtedy wartość TOP i zmienia kierunek liczenia. Wartość TCNTn będzie równa TOP na czas jednego cyklu.

Poniższy przebieg przedstawia sytuację gdy OCRnA lub ICRn są użyte do określenia wartości TOP:



Flaga przepełnienia licznika TOVn jest ustawiana w tym samym cyklu, w którym rejestr OCRnx jest aktualizowany. Gdy OCRnA lub ICRn są używane do zdefiniowania wartości TOP, flagi OCnA lub ICFn są ustawiane gdy TCNTn osiągnie wartość TOP. Flaga przerwania może zostać wykorzystana do wygenerowania przerwania za każdym razem gdy licznik osiągnie wartość TOP lub BOTTOM.

Podczas zmiany wartości TOP program musi być pewny, że nowa wartość jest równa lub większa od zawartości wszystkich rejestrów OCRnx. Jeśli wartość TOP jest od któregoś rejestru mniejsza, wtedy nigdy nie nastąpi przypadek w którym licznik TCNTn i rejestr OCRnx będą miały tą samą wartość. Częstotliwość PWM w trybie korekcji fazy i częstotliwości jest określona wzorem:

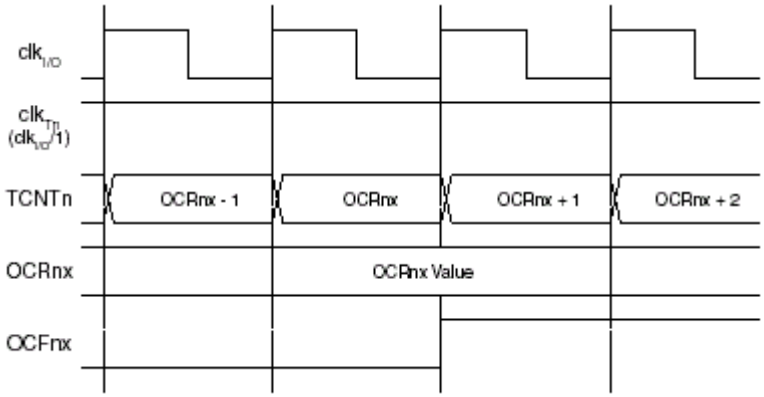
$$f_{OCnxPFCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

Współczynnik N przyjmuje wartości 1,8,64,256,1024.

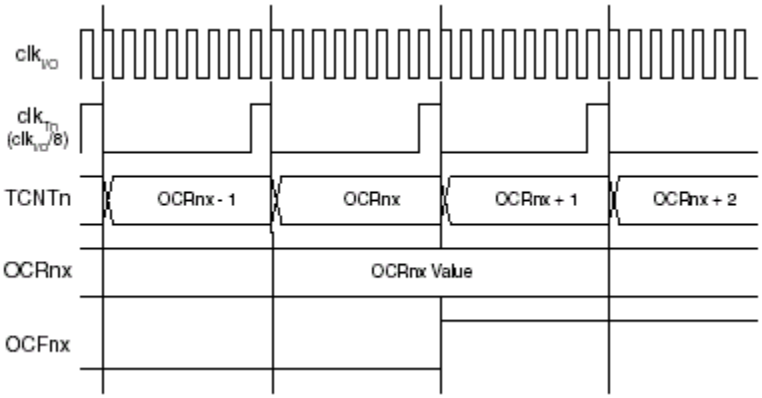
Graniczne wartości rejestru OCRnx reprezentują specjalne przypadki podczas generacji w trybie PWM z korekcją fazy i częstotliwości. Jeśli OCRnx jest ustawiona na wartość BOTTOM to na wyjściu pojawi się stan niski, a jeśli rejestr jest ustawiony na TOP to na wyjściu pojawi się stan wysoki. W trybie inwersyjnym wyjście będzie miało przeciwne logicznie wartości.

PRZEBIEGI CZASOWE

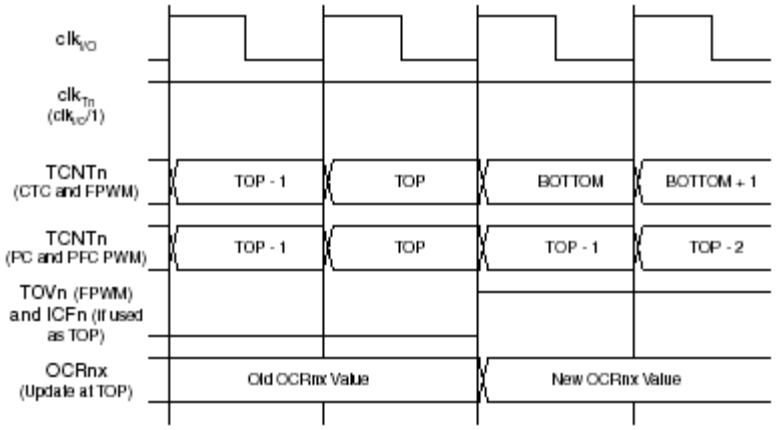
Poniższy wykres przedstawia przebieg czasowy ustawiania rejestru OCRnx.



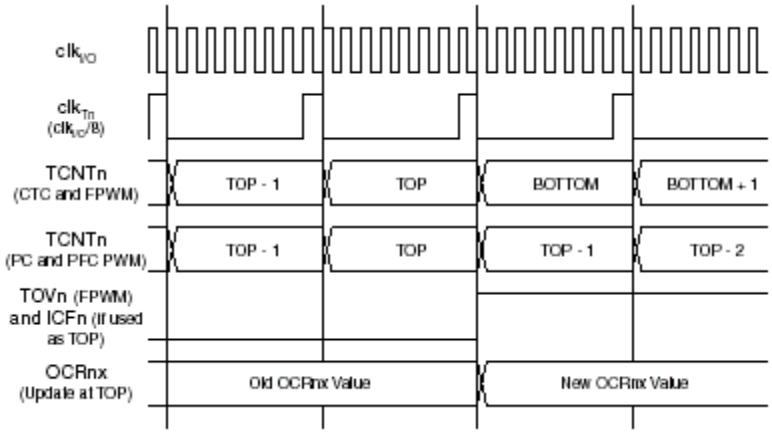
Poniższy wykres przedstawia ten sam przebieg, lecz z włączoną opcją przelicznika wstępnego:



Kolejny przebieg pokazuje sekwencję liczenia blisko wartości maksymalnej (TOP) dla różnych trybów:



Ten wykres również pokazuje sekwencję liczenia, lecz z włączoną opcją przelicznika wstępnego:



Opis wybranych rejestrów 16-bitowego Zegara/Licznika

Rejestr kontrolny A Zegara/Licznika1 – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Rejestr kontrolny A Zegara/Licznika3 – TCCR3A

Bit	7	6	5	4	3	2	1	0	
	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

• Bit 7:6 – COMnA1:0: Tryb porównania wyjścia dla Kanału A
• Bit 5:4 – COMnB1:0: Tryb porównania wyjścia dla Kanału B
• Bit 3:2 – COMnC1:0: Tryb porównania wyjścia dla Kanału C

Poniższa tabela pokazuje funkcje bitu COMnx1:0, gdy bity WGMn3:0 są ustawione na tryb normalny lub CTC:

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Opis
0	0	Normalne działanie portu, OCnA/OCnB/OCnC - odłączone
0	1	Przełączenie OCnA/OCnB/OCnC przy dopasowaniu
1	0	Wyczyszczenie OCnA/OCnB/OCnC przy dopasowaniu (ustawienie wyjścia na niski poziom)
1	1	Ustawienie OCnA/OCnB/OCnC przy dopasowaniu (Ustawienie wyjścia na poziom wysoki)

Kolejna tabela pokazuje funkcje bitu COMnx1:0 gdy bity WGMn3:0 są ustawione na tryb szybki PWM:

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Opis
0	0	Normalne działanie portu, OCnA/OCnB/OCnC - odłączone
0	1	WGMn3=0 Normalne działanie portu, OCnA/OCnB/OCnC – odłączone. WGMn3=1: Przełączenie OCnA przy dopasowaniu, OCnB/OCnC – zarezerwowane.
1	0	Wyczyszczenie OCnA/OCnB/OCnC przy dopasowaniu. Ustawienie OCnA/OCnB/OCnC przy wartości TOP.
1	1	Ustawienie OCnA/OCnB/OCnC przy dopasowaniu. Czyszczenie OCnA/OCnB/OCnC przy wartości TOP.

Następna tabela pokazuje funkcje bitu COMnx1:0 gdy bity WGMn3:0 są ustawione na tryb PWM z korekcją fazy i częstotliwości:

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Opis
0	0	Normalne działanie portu, OCnA/OCnB/OCnC - odłączone
0	1	WGMn3=0 Normalne działanie portu, OCnA/OCnB/OCnC – odłączone. WGMn3=1: Przełączenie OCnA przy dopasowaniu, OCnB/OCnC – zarezerwowane.
1	0	Wyczyszczenie OCnA/OCnB/OCnC przy dopasowaniu podczas liczenia w górę. Ustawienie OCnA/OCnB/OCnC przy dopasowaniu podczas liczenia w dół.
1	1	Ustawienie OCnA/OCnB/OCnC przy liczeniu w górę. Czyszczenie OCnA/OCnB/OCnC przy dopasowaniu podczas liczenia w dół.

• Bit 1:0 – WGMn1:0: Tryb Generacji Fali
Te bity kontrolują sekwencję liczenia, źródło wartości maksymalnej (TOP) licznika, oraz określają jaki typ fal zostanie wygenerowany.

Opis bitów trybu generacji fali:

Tryb	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Tryb Operacji Zegara/Licznika	TOP	Aktualizacja OCRnx przy:	Ustawienie flagi TOVn przy:
0	0	0	0	0	Normalny	0Xfff	Natychmiast	MAX
1	0	0	0	1	PWM, korekcja fazy, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, korekcja fazy, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, korekcja fazy, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Natychmiast	MAX
5	0	1	0	1	Szybki PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Szybki PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Szybki PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, korekcja fazy i częstotliwości	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, korekcja fazy i częstotliwości	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, korekcja fazy	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, korekcja fazy	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Natychmiast	MAX
13	1	1	0	1	(zarezerwowany)	-	-	-
14	1	1	1	0	Szybki PWM	ICRn	TOP	TOP
15	1	1	1	1	Szybki PWM	OCRnA	TOP	TOP

Rejestr kontrolny B Zegara/Licznika1 – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	INCN1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Odczyt/Zapis (R/W)	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Rejestr kontrolny B Zegara/Licznika3 – TCCR3B

Bit	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	-	WGM33	WGM32	CS32	CS31	CS31	TCCR3B
Odczyt/Zapis (R/W)	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

• Bit 7 – ICNCn: Reduktor Zakłóceń Wejściowych
Ustawienie tych bitów (na wartość 1) aktywuje reduktor zakłóceń, co powoduje że sygnał wejściowy na pinie przechwytywania wejścia ICPn jest filtrowany. Przechwytywanie sygnałów na wejściu jest wtedy opóźnione.

• Bit 6 – ICESn: Wybór Zbocza
Ten bit określa które zbocze będzie użyte na pinie ICPn do wywołania przechwyconego zdarzenia. Kiedy bit ICESn jest ustawiony na zero, to zostanie użyte opadające zbocze. Jeśli ustawimy ICESn na jeden, narastające zbocze wywoła zdarzenia.

Jeżeli rejestr ICRn jest używany jako wartość TOP dla licznika, to pin ICPn zostaje odłączony.

• Bit 5 – Bit Zarezerwowany

• Bit 4:3 – WGMn3:2: Tryb Generacji Fali

• Bit 2:0 – CSn2:0: Wybór źródła taktowania
Trzy bity określają źródło taktowania licznika. Opis bitów:

CSn2	CSn1	CSn0	Opis
0	0	0	żadne źródło nie zostało wybrane
0	0	1	$clk_{I/O}/1$
0	1	0	$clk_{I/O}/8$
0	1	1	$clk_{I/O}/64$
1	0	0	$clk_{I/O}/256$
1	0	1	$clk_{I/O}/1024$
1	1	0	zewnętrzne źródło podłączone do Tn (taktowanie opadającym zboczem)
1	1	1	zewnętrzne źródło podłączone do Tn (taktowanie narastającym zboczem)

Rejestr kontrolny C Zegara/Licznika3 – TCCR3C

Bit	7	6	5	4	3	2	1	0	
	FOC3A	FOC3B	FOC3C	-	-	-	-	-	TCCR3C
Odczyt/Zapis (R/W)	W	W	W	R	R	R	R	R	
Wartość początkowa	0	0	0	0	0	0	0	0	

• Bit 7 – FOCnA: Wymuszenie Porównania na Wyjściu dla Kanału A
• Bit 6 – FOCnB: Wymuszenie Porównania na Wyjściu dla Kanału B
• Bit 5 – FOCnC: Wymuszenie Porównania na Wyjściu dla Kanału C
Bity FOCnA/FOCnB/FOCnC są aktywne tylko gdy bity WGMn3:0 są ustawione na tryb inny niż PWM.
• Bit 4:0 – Bity Zarezerwowane

Rejestr Porównawczy 1A – OCR1AH oraz OCR1AL

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL

Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Rejestr Porównawczy 1B – OCR1BH oraz OCR1BL

Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Rejestr Porównawczy 1C – OCR1CH oraz OCR1CL

Bit	7	6	5	4	3	2	1	0	
	OCR1C[15:8]								OCR1CH
	OCR1C[7:0]								OCR1CL
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Rejestr Porównawczy 3A – OCR1AH oraz OCR1AL

Bit	7	6	5	4	3	2	1	0	
	OCR3A[15:8]								OCR3AH
	OCR3A[7:0]								OCR3AL
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Rejestr Porównawczy 3B – OCR3BH oraz OCR3BL

Bit	7	6	5	4	3	2	1	0	
	OCR3B[15:8]								OCR3BH
	OCR3B[7:0]								OCR3BL
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Rejestr Porównawczy 3C – OCR3CH oraz OCR3CL

Bit	7	6	5	4	3	2	1	0	
	OCR3C[15:8]								OCR3CH
	OCR3C[7:0]								OCR3CL
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Wyjściowy rejestr porównawczy zawiera 16-bitową wartość która jest ciągle porównywana z wartością licznika (TCNTn). Porównanie może zostać użyte do generowania przerwania lub fali na pinie OCnx. Dostęp procesora do rejestrów odbywa się poprzez 8-bitowy rejestr TEMP. Ten rejestr jest „dzielony" przez wszystkie 16-bitowe rejestry.

Wejściowy Rejestr 1 – ICR1H oraz ICR1L

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Wejściowy Rejestr 3 – ICR3H oraz ICR3L

Bit	7	6	5	4	3	2	1	0	
	ICR3[15:8]								ICR3H
	ICR3[7:0]								ICR3L
Odczyt/Zapis (R/W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Wartość początkowa	0	0	0	0	0	0	0	0	

Dostęp procesora do rejestrów odbywa się poprzez 8-bitowy rejestr TEMP. Ten rejestr jest „dzielony" przez wszystkie 16-bitowe rejestry.

UKŁAD CZASOWO-LICZNIKOWY 3, UKŁAD CZASOWO-LICZNIKOWY 2, UKŁAD CZASOWO-LICZNIKOWY 1, PRESCALERY

Układ czasowo-licznikowy 3, układ czasowo-licznikowy 2, układ czasowo-licznikowy 1 współdzielą ten sam moduł prescalera, ale mogą mieć różne jego ustawienia. Poniższy opis odnosi się do wszystkich wspomnianych układów czasowo-licznikowych.

WEWNĘTRZNE ŹRÓDŁO ZEGAROWE

Układ czasowo-licznikowy może być taktowany bezpośrednio przez zegar systemowy (przez ustawienie CSn 2:0 = 1). Operacje wykonują się wtedy najszybciej z maksymalną częstotliwością zegara układu czasowo-licznikowego równą częstotliwości zegara systemowego ($f_{CLK\_I/O}$). Inne rozwiązanie polega na

użyciu jednego z czterech zaworów (taps) prescalera jako źródła zegarowego. Zegar prescalera ma wówczas częstotliwość równą $f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O})/256$ lub $f_{CLK\_I/O}/1024$.

RESET PRESCALERA

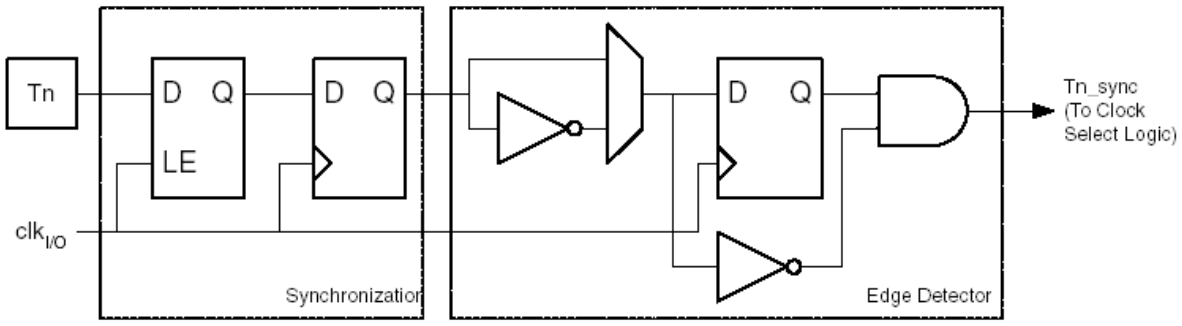
Prescaler działa niezależnie od układu wyboru zegara układu czasowo-licznikowego i jest współdzielony przez układ czasowo-licznikowy 1, układ czasowo-licznikowy 2 oraz układ czasowo-licznikowy 3. Ponieważ wybór zegara układu czasowo-licznikowego nie oddziałuje na prescaler, jego stan będzie miał znaczenie w sytuacjach w których preskalowany zegar jest używany. Przykład preskalowania artefaktów (artifacts) pojawia się gdy układ czasowy jest aktywny i taktowany przez prescaler (6 > CSn 2:0 >1). Liczba taktów zegara systemowego od chwili gdy układ czasowy jest aktywny do momentu pojawienia się pierwszego zliczenia zawiera się od 1 do N+1, gdzie N jest dzielnikiem prescalera (8, 64, 256, 1024). Istnieje możliwość użycia resetu prescalera do synchronizacji układu czasowo-licznikowego z wykonaniem programu. Jednak w tym przypadku należy zachować ostrożność jeżeli inny układ czasowy który współdzieli prescaler również używa preskalowania. Reset prescalera ma wpływ na okres prescalera wszystkich układów czasowo-licznikowych jakie są do niego podłączone.

ZEWNĘTRZNE ŹRÓDŁO ZEGAROWE

Zewnętrzne źródło zegarowe podpięte do pinu Tn może zostać użyte jako zegar układu czasowo-licznikowego (clk_{T1}, clk_{T2}, clk_{T3}). Pin jest próbkowany raz w każdym takcie zegara systemowego przez pin układu synchronizującego. Zsynchronizowany (próbkowany) sygnał jest przepuszczany przez detektor zbocza. Rys. 58 pokazuje schemat blokowy układu synchronizacji i detektora zbocza. Rejestry są taktowane narastającym zboczem wewnętrznego zegara systemowego ($clk_{I/O}$). Zatrzask jest przezroczysty dla wysokich okresów wewnętrznego zegara systemowego.

Detektor zbocza generuje jeden impuls $clk_{T1}/clk_{T2}/clk_{T3}$ dla każdego narastającego bądź opadającego zbocza jakie wykryje.

Rys. 58 Próbkowanie pinu Tn



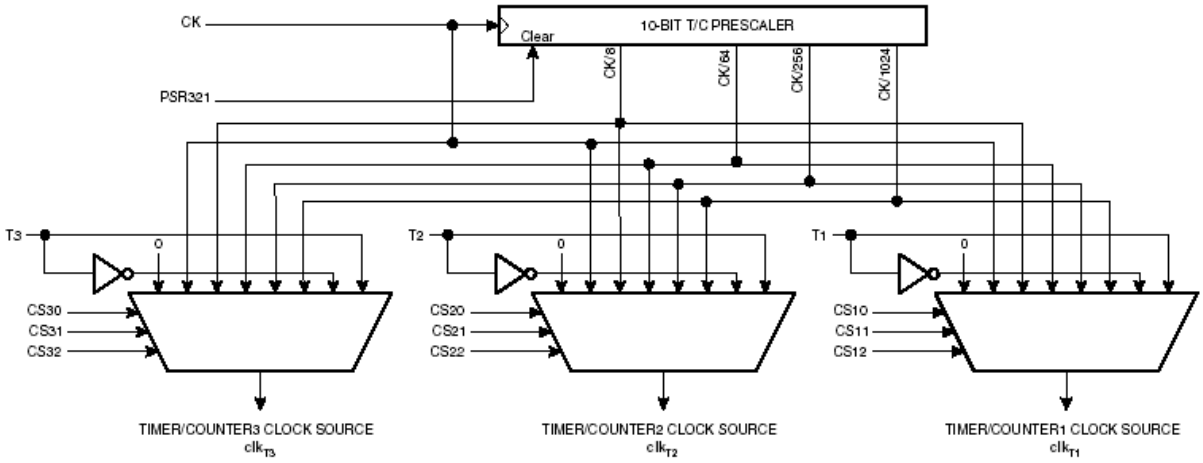
Układ synchronizacji i detektor zbocza wprowadzają opóźnienie od 2,5 do 3,5 taktu zegara systemowego.

Aktywacja i dezaktywacja wejścia zegara musi zostać dokonana kiedy Tn jest stabilny przez przynajmniej jeden takt zegara systemowego, w przeciwnym razie istnieje ryzyko że zostanie wygenerowany fałszywy impuls zegara układu czasowo-licznikowego.

Każda połowa okresu dodanego zewnętrznego zegara musi być dłuższa niż jeden takt zegara systemowego aby zapewnić właściwe próbkowanie. Zewnętrzny zegar musi mieć zagwarantowaną częstotliwość mniejszą od połowy częstotliwości zegara systemowego ($f_{ExtClk} < f_{clk\_I/O}/2$). Od czasu gdy detektor zbocza używa próbkowania, maksymalna częstotliwość zewnętrznego zegara jaka może zostać przez niego wykryta równa się połowie częstotliwości próbkowania (twierdzenie Nyquista). Zmienność częstotliwości zegara systemowego i cyklu roboczego jest spowodowana przez tolerancję źródła oscylacji (kryształ, rezonator i kondensatory) sprawiają, że zalecana maksymalna częstotliwość zewnętrznego zegara jest mniejsza niż $f_{CLK\_I/O}/2{,}5$.

Zewnętrzne źródło zegarowe nie może być preskalowane.

Rys. 59 Prescaler układów czasowych



REJESTR SPECJALNY WEJŚCIA/WYJŚCIA – SFIOR

Bit	7	6	5	4	3	2	1	0	
	TSM	–	–	ADHSM	ACME	PUD	PSR0	PSR321	SFIOR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM: Tryb synchronizacji układu czasowo-licznikowego**

  Wpisanie 1 na bit TSM spowoduje że PSR0 i PSR321 staną się rejestrami, które utrzymają swą początkową wartość dopóki nie zostaną zapisane ponownie lub bit TSM nie zostanie zmieniony na zero. Ten tryb jest użyteczny jeżeli chcemy zsynchronizować układy czasowo-licznikowe. Ustawienie wszystkich bitów TSM oraz odpowiednich/odpowiedniego bitów/bitu PSR spowoduje że odpowiednie układy czasowo-licznikowe zostaną zatrzymane i mogą zostać skonfigurowane tą

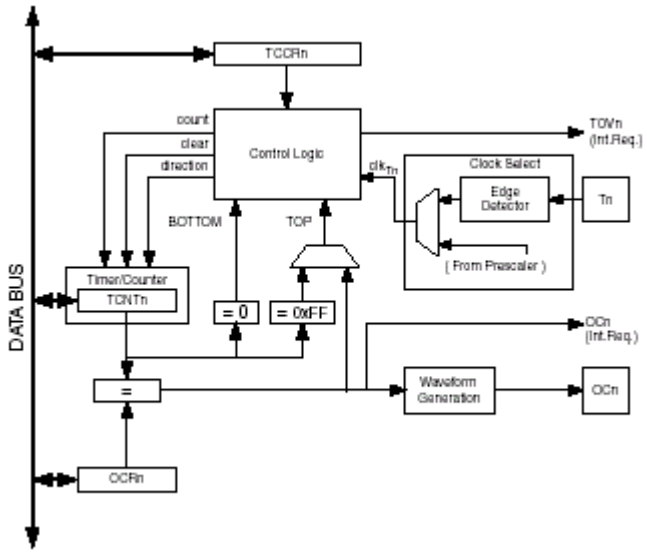
samą wartością bez ryzyka, że jeden z nich przesunie się naprzód podczas konfiguracji. Kiedy bit TSM jest ustawiony na zero układy czasowo-licznikowe zaczynają liczyć równocześnie.

- **Bit 0 – PSR321: Reset prescalera układu czasowo-licznikowego 3, układu czasowo-licznikowego 2, układu czasowo-licznikowego 1**
  Wpisanie na bit PSR321 jedynki resetuje prescaler układu czasowo-licznikowego 3, układu czasowo-licznikowego 2 i układu czasowo-licznikowego 1. Bit zostanie skasowany sprzętowo po rozpoczęciu operacji. Wpisanie zera na ten bit nie przynosi żadnych efektów. Należy zauważyć że wszystkie układy czasowo-licznikowe dzielą ten sam prescaler i reset tego prescalera ma wpływ na wszystkie układy czasowe. Bit zawsze jest odczytywany jako zero.

8-BITOWY LICZNIK PWM

Główne cechy 8-bitowego Timera/Licznika2:
- Licznik z pojedynczym kanałem
- Timer z zerowaniem przy zgodności z zadanym warunkiem (samoprzeładowanie)
- Bezhazardowy modulator szerokości impulsu z poprawną fazą
- Generator
- Licznik zdarzeń zewnętrznych
- Źródła przerwań przepełnienia i porównania (TOV2, OCF2)

Uproszczony schemat blokowy licznika:



BOTTOM – zawartość licznika wynosi 0x00
MAX – zawartość licznika wynosi 0xFF

TOP – zawartość licznika jest równa najwyższej wartości w sekwencji liczenia

REJESTRY
Rejestr licznika TCNT2 oraz rejestr porównawczy OCR2 są rejestrami 8-bitowymi.
Wszystkie sygnały przerwań są umieszczane w rejestrze TIFR. Każde przerwanie jest indywidualnie maskowane przez rejestr TIMSK.
Licznik może być sterowany wewnętrznie poprzez prescalera lub zewnętrznie poprzez pin T2. Licznik jest nieaktywny, gdy żadne źródło nie jest wybrane.

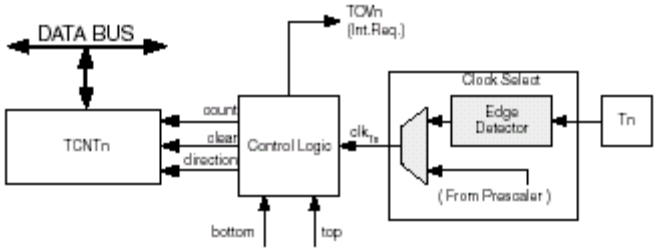
Podwójnie buforowany rejestr OCR2 jest porównywany z wartością licznika przez cały czas. Wyniki porównania może być przez generator.

Źródła taktowania

Licznik może być taktowany przez wewnętrzne lub zewnętrzne źródło. Wyboru dokonuje się przez ustawienie bitów CS22:0 znajdujących się w rejestrze TCCR2.

Jednostka licząca

Główną częścią 8-bitowego licznika jest programowalna dwukierunkowa jednostka licząca:



Count zwiększenie lub zmniejszenie TCNTn o 1.
Direction Wybór pomiędzy inkrementacją i dekrementacją.
Clear Wyzerowanie TCNTn (ustawienie wszystkich bitów na zero).
clkTn Impuls zegara/licznika.
TOP Sygnalizuje że TCNTn osiągnął wartość maksymalną.
BOTTOM Sygnalizuje że TCNTn osiągnął wartość minimalną.

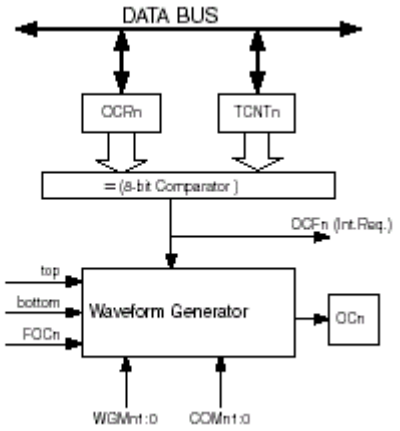
W zależności od wybranego trybu licznik jest czyszczony, inkrementowany lub dekrementowany za każdym impulsem zegara. Impulsy mogą być generowane z zewnętrznego lub zewnętrznego źródła w zależności od ustawienia bitów wyboru zegara(CS22:0). Jeżeli żadne źródło nie jest wybrane (CS22:0=0) licznik jest zatrzymany

Sekwencja liczenia jest określana za pomocą bitów WGM01 i WGM00 zlokalizowanych w rejestrze kontrolnym TCCR2.

Flaga przepełniena licznika TOV2 jest ustawiana zgodnie z trybem wybranym przez bity WGM21:0. TOV2 może być użyta do generowania przerwań.

Komparator

8-bitowy komparator ciągle porównuje zawartość TCNT2 z zawartością rejestru OCR2. Kiedy zawartości są takie same, kompartor sygnalizuje dopasowanie. Spowoduje to ustawienie flagi OCF2 w następnym cyklu. Jeżeli OCIE2 = 1, to flaga OCF2 generuje przerwanie. Flaga jest automatycznie czyszczona po wykonaniu przerwania.



Rejestr OCR2 jest podwujnie buforowany, gdy używany jest jeden z trybów PWM(modulacja szerokości impulsu). W trybie normalnym i CTC podwójne buforowanie jest niedostępne.
Przy podwójnym buforowaniu procesor posiada dostęp do rejestru buforowego OCR2. Jeżeli podwójne buforowanie jest wyłączone procesor ma bezpośredni dostęp do rejestru OCR2.

Wymuszanie komparacji wyjścia

W trybie generacji przebiegów nie-PWM wyjście komparatora może zostać wymuszone za pomocą zapisu jedynki do bitu FOC2 tak aby spełniało przewidzianą funkcję .Czynność ta nie spowoduje ustawienia flagi OCF2 ani powtórnego załadowania/wyzerowania timera ,lecz stan pinu OC2 zostanie uaktualniony tak jakby stało się to w przypadku rzeczywistej zgodności porównywanych wartości (ustawienia bitu COM21:0 definiują czy pin OC2 jest ustawiony, wyzerowany lub zmienił wartość)

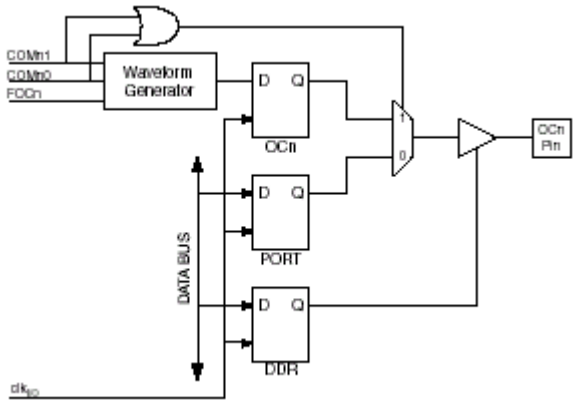
Wszystkie operacje zapisywania przez procesor rejestru TCNT2 blokują każde porównanie, które wystąpi w następnym cyklu zegarowym, nawet gdy licznik jest zatrzymany. Dzięki temu OCR2 może być inicjalizowany tą samą wartością co TCNT2 bez wywoływania przerwania.

Użycie jednostki komparacji wyjścia

Jeśli zapis TCNT2 w dowolnym trybie pracy zablokuje wszystkie możliwości porównania na jeden okres zegara to istnieją zagrożenia związane ze zmianą TCNT2 gdy użyty jest jakikolwiek kanał komparacji wyjścia, niezależnie od tego czy Timer/Licznik zlicza czy nie. Jeżeli wartość zapisana do TCNT2 równa jest wartości OCR2 wtedy dobry rezultat porównania może zostać przeoczony czego skutkiem będzie błąd w generowanym przebiegu. Podobnie nie należy zapisywać wartości do TCNT2 równej BOTTOM kiedy licznik zlicza w dół.

Jednostka zgodności komparacji wyjścia

Bity trybu porównywania wyjścia (COM21:0) mają dwie funkcje. Generator przebiegów używa bitów COM21:0 do definiowania stanu porównania wyjścia (OC2) w sprawdzeniu zgodności następnego porównania. Bity COM21:0 kontrolują również pin OC2.



Generator falowy używa bitów COM21:0 różnie w trybie normalnym, CTC i PWM. We wszystkich trybach ustawienie COM21:0=0 informuje generator, że żadne działanie na rejestrze OCnx nie będzie podjęte przy następnym dopasowaniu.
Zmiana bitów COMnx1:0 będzie miała efekt przy pierwszym dopasowaniu po zapisaniu bitów.
W trybie CTC i normalnym można wymusić natychmiastowy efekt przez użycie strobujących bitów FOC2.

Tryby Operacji

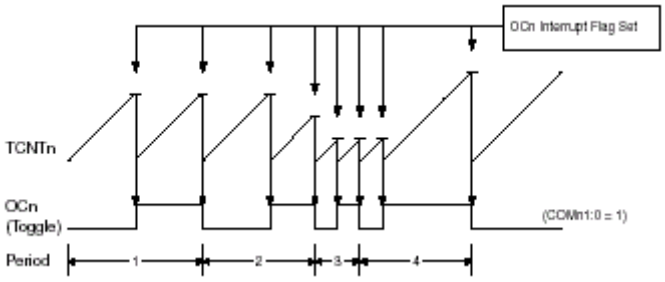
Tryb operacji, czyli zachowanie Licznika i pinów porównania wyjścia, jest zdefiniowane przez kombinację bitów trybu generacji fal (WGM21:0) i trybu porównania wyjścia (COM21:0). Bity trybu porównania wyjścia nie mają wpływu na sekwencję liczenia, dopóki określają to bity trybu generacji fali. Bity COM21:0 określają, czy generowany sygnał wyjściowy powinien być odwrócony czy nie. Dla trybów różnych od PWM bity COM21:0 czy wyjście powinno być ustawione, czyszczone czy przełączone przy porównaniu.

Tryb Normalny

Najprostszym trybem operacji jest tryb normalny (WGM21:0=0). W tym trybie licznik zawsze liczy w górę, i nie jest wykonywane czyszczenie licznika. Licznik gdy osiąga swoją maksymalną wartość (MAX=0xFF) zaczyna liczyć od wartości minimalnej (0x00). W tym trybie flaga przekroczenia zakresu TOV2 zostanie ustawiona w tym samym cyklu zegarowym, w którym wartość TCNT2 staje się zerem. W tym przypadku flaga TOV2 zachowuje się jak dziewiąty bit, z wyjątkiem tego że jest tylko ustawiana, nie czyszczona.
Nie ma w tym trybie żadnych specjalnych przypadków do rozpatrywania. Nowa wartość licznika może być zapisana w każdej chwili. Komparator wyjściowy może być użyty do generowania przerwań, nie zaleca się stosowania do generacji fal ponieważ wtedy za bardzo obciążony jest procesor.

Tryb CTC (czyszczenia zegara)

W trybie CTC (WGM21:0=2) rejestr OCR2 jest używany do manipulowania rozdzielczością licznika. W trybie CTC licznik jest czyszczony do zera kiedy wartość licznika (TCNT2) jest zgodna z rejestrem OCR2. Rejestr ten może określać maksymalną wartość licznika, a zatem i jego rozdzielczość. Ten tryb pozwala na większą kontrolę nad dopasowaniem częstotliwości na wyjściu. Upraszcza on także operacje liczenia zewnętrznych zdarzeń.
Na rysunku pokazano wykres czasowy dla trybu CTC. Wartość licznika (TCNT2) jest zwiększana dopóki nie wystąpi zgodność z OCR2. Wtedy następuje czyszczenie licznika.



Przerwanie może zostać wygenerowane, za każdym razem gdy licznik osiągnie wartość maksymalną (TOP) używając flagi OCF2. Jeżeli przerwanie jest dostępne, podprogram obsługi przerwań może zostać użyty do odświeżania wartości TOP. Jednak zmienianie wartości maksymalnej na bliską wartości minimalnej podczas pracy licznika należy robić bardzo ostrożnie jeżeli tryb CTC nie jest podwójnie buforowany. Jeżeli nowa wartość maksymalna jest mniejsza od bieżącej wartości licznika, to licznik tego nie wychwyci i będzie liczył do swojej maksymalnej wartości (0xFF) i zacznie liczyć od zera.

Dla generowania fali na wyjściu w trybie CTC, wyjście OCnA może być ustawione na zmianę swojego stanu logicznego przy każdym dopasowaniu. Robi się to ustawiając bity trybu porównania wyjścia na tryb zmiany (COM21:0=1). Generowana fala będzie miała maksymalną wartość częstotliwości gdy OCR2 jest ustawiony na zero (0x00):

$$f_{OC2} = f_{clk\_I/O}/2$$

Częstotliwość generowanej fali jest określona następującym wzorem:

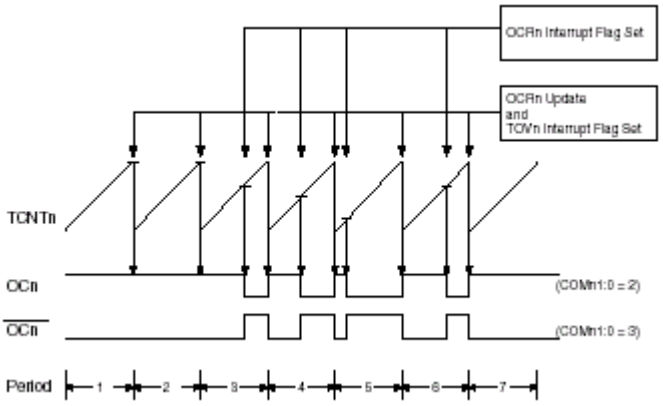
$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

Współczynnik N przyjmuje wartości 1,8,64,256,1024.
Tak jak w trybie normalnym flaga TOV2jest ustawiana w tym samym cyklu zegarowym, w którym licznik przechodzi z wartości MAX do 0x00.

Szybki tryb Modulacji Szerokości Impulsu(Szybki tryb PWM)
Szybki tryb PWM (WGM21:0=1) zapewnia opcję generacji fal wysokiej częstotliwości. Szybki tryb różni się od innych trybów PWM operacją na pojedynczym zboczu.
Licznik liczy w pętli od wartości BOTTOM do wartości TOP. W nieodwracającym trybie porównania wyjścia, wyjście OC2 jest czyszczone przy porównaniu rejestru OCR2 i TCNT2, i jest ustawiane przy wartości minimalnej (BOTTOM). W trybie odwracającym jest ustawiane przy porównaniu i czyszczone przy BOTTOM. Przez operacje wykonywane na pojedynczym zboczu częstotliwość pracy w szybkim

trybie PWM może być dwa razy większa niż przy operacjach wykonywanych na podwójnym zboczu. Wysoka częstotliwość czyni szybki tryb PWM odpowiedni do regulacji zasilania, prostowania, przetworników analogowo-cyfrowych.
W szybkim trybie PWM licznik jest inkrementowany dopóki nie osiągnie MAX. Następnie licznik jest czyszczony w następnym cyklu zegarowym.

Przebieg czasowy dla omawianego trybu:



Flaga przepełnienia TOV2 jest ustawiana za każdym razem gdy licznik osiąga Max.

W szybkim trybie PWM komparator umożliwia generowanie przebiegów PWM na pinie OC2. Ustawienie bitów COM21:0 na wartość 2 spowoduje generacje przebiegów odwróconych, a ustawienie tych bitów na wartość 3 nieodwróconych.

Fale PWM są generowane poprzez ustawienie (lub czyszczenie) rejestru OC2 przy zgodności porównania pomiędzy OCR2 i TCNT2, i czyszczenie (lub ustawianie) rejestru OC2 w cyklu zegarowym w którym licznik jest czyszczony (zmiana stanu z MAX na BOTTOM).

Częstotliwość można obliczyć według wzoru:

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

Współczynnik N przyjmuje wartości 1,8,64,256,1024.

Graniczne wartości rejestru OCR2 określają specjalne zdarzenia podczas generacji w szybkim trybie PWM. Jeśli OCR2 jest ustawiony na wartość BOTTOM, to na wyjściu pojawi się ostry impuls dla każdego MAX+1-szego cyklu zegara. Ustawienie rejestru OCRnx na wartość MAX, spowoduje pojawienie się na wyjściu stanu wysokiego lub niskiego (w zależności od polaryzacji wyjścia ustawionej na bitach COMnx1:0).
Częstotliwość generacji równą 50% cyklu pracy, można osiągnąć poprzez ustawienie OC2 na zmianę swojego stanu logicznego przy dopasowaniu (COM21:0=1).

Generowane fale mają maksymalną częstotliwość:

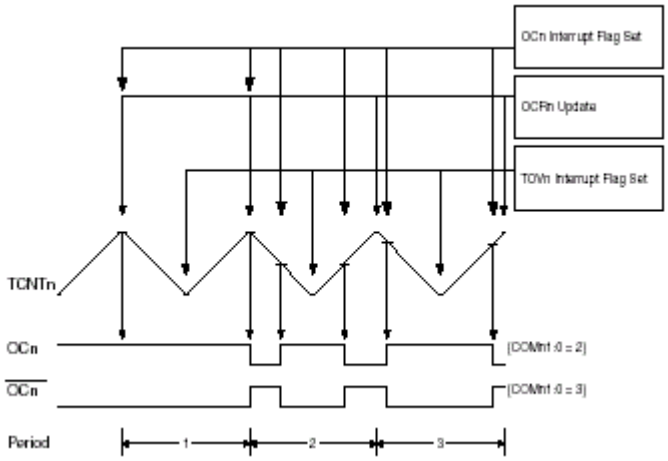
$$f_{OC2}=f_{clk\_I/O}/2$$

gdy rejestr OCR2 jest ustawiony na zero (0x00).

Tryb Modulacji Szerokości Impulsu z Korekcją Fazy
Tryb modulacji szerokości impulsu z korekcją fazy (WGM21:0=3) zapewnia opcje generowania fal o wysokiej rozdzielczości. Ten tryb bazuje na operacjach wykonywanych na podwójnym zboczu. Licznik liczy od wartości BOTTOM do wartości MAX, i następnie od MAX do BOTTOM. W nieodwracającym trybie porównania wyjścia, OC2 jest czyszczone przy dopasowaniu pomiędzy TCNT2 i OCR2 podczas liczenia w górę. OC2 jest ustawiane przy dopasowaniu podczas liczenia w dół.

W omawianym trybie licznik jest inkrementowany dopóki nie osiągnie wartości MAX. Wtedy licznik zmienia kierunek liczenia. Wartość TCNTn będzie równa MAX na jeden cykl zegarowy.

Poniższy przebieg przedstawia tryb PWM z korekcją :



Flaga przepełnienia licznika TOV2 jest ustawiana za każdym razem gdy licznik osiąga wartość BOTTOM. Flaga przerwania może zostać wykorzystana do wygenerowania przerwania za każdym razem gdy licznik osiągnie wartość BOTTOM.

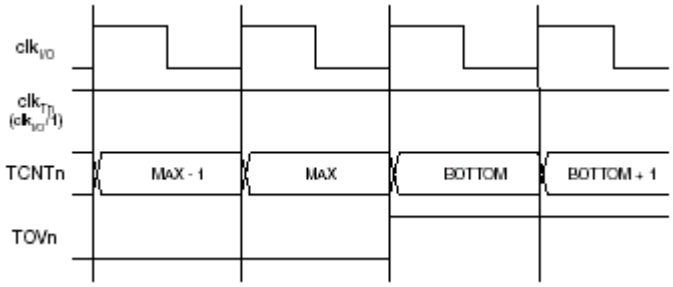
W trybie PWM z korekcją fazy możliwe jest generowanie fali na pinie OC2. Częstotliwość na wyjściu w tym trybie można obliczyć według wzoru:

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

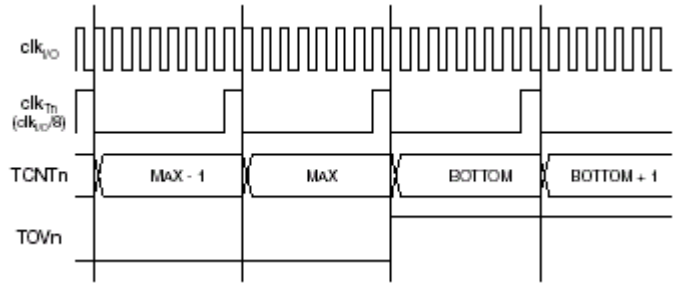
Współczynnik N przyjmuje wartości 1,8,64,256,1024.
Graniczne wartości rejestru OCR2 reprezentują specjalne przypadki podczas generacji w trybie PWM z korekcją fazy. Jeśli OCRnx jest ustawiona na wartość BOTTOM to na wyjściu pojawi się stan niski, a jeśli rejestr jest ustawiony na MAX to na wyjściu pojawi się stan wysoki. W trybie inwersyjnym wyjście będzie miało przeciwne logicznie wartości.

Przebiegi czasowe Licznika

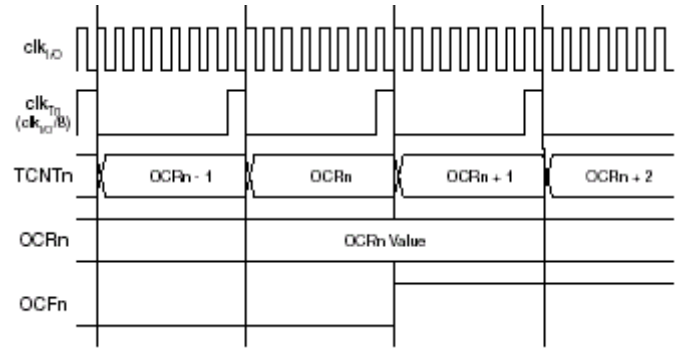
Poniższy wykres pokazuje przebiegi czasowe, podczas liczenia przy wartości bliskiej MAX, dla wszystkich trybów oprócz trybu PWM z korekcją fazy:



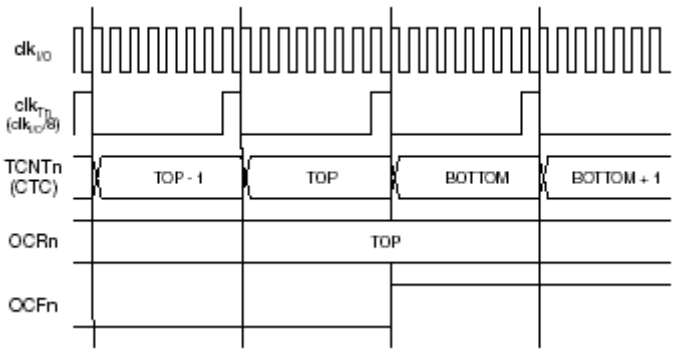
Poniższy wykres przedstawia ten sam przebieg, lecz z włączoną opcją przelicznika wstępnego(prescaler) :



Kolejny przebieg przedstawia ustawianie OCF2 dla wszystkich trybów oprócz CTC:



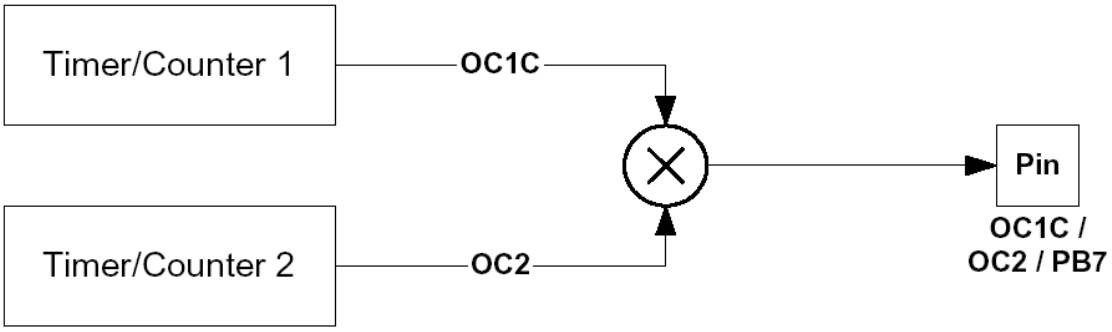
Przebieg przedstawia ustawianie OCF2 i czyszczenie TCNT2 dla trybu CTC:



OUTPUT COMPARE MODULATOR (OCM1C2)

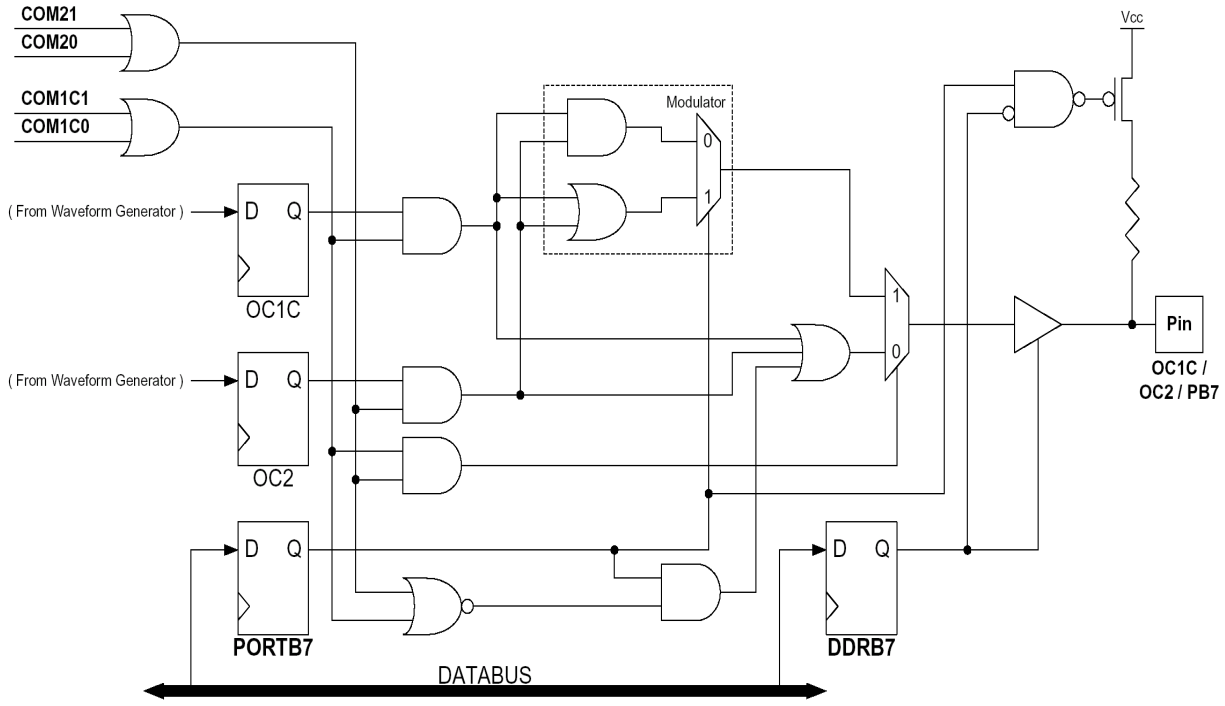
Output Compare Modulator (OCM) pozwala na generowanie fal przez modulację częstotliwości nośnej. Komparator używa wyjść: licznika 16 bitowego (wyjście Output Compare Unit C) i licznika 8 bitowego 2 (wyjście Output Compare Unit).

Rys. 71. Schemat blokowy OCM



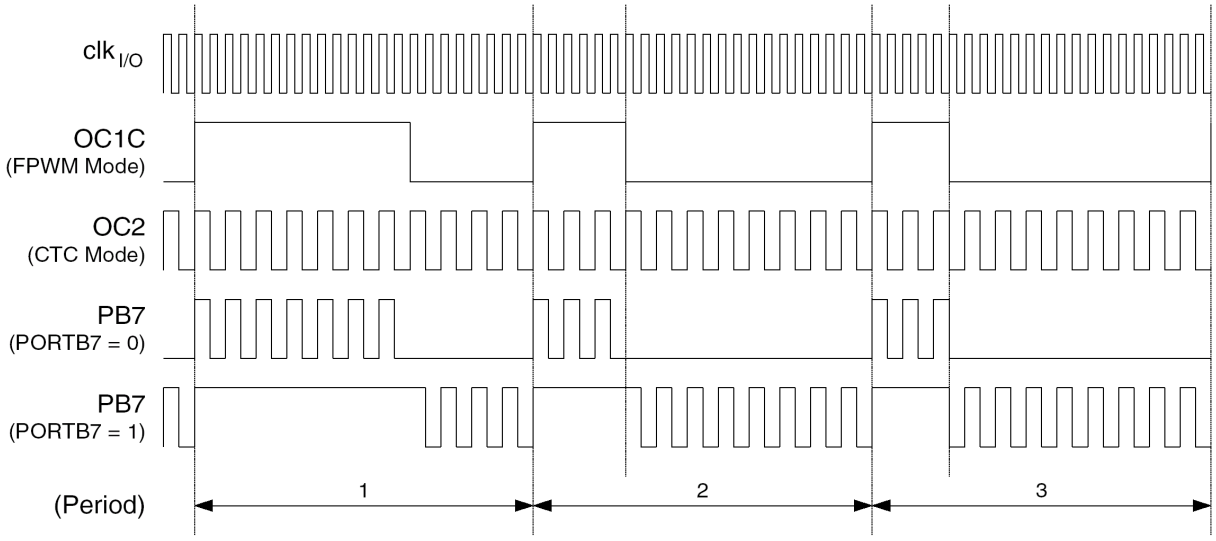
Kiedy modulator jest włączony oba wyjścia liczników są modulowane razem jak zaznaczono na rys. 71. Output Compare unit 1C (OCIC) and Output Compare unit 2 (OC2) współdzielą wyprowadzenie PB7 (port B) jako wyjście (jest do alternatywna funkcja tych wyprowadzeń portu B). Jeśli OC1C i OC2 są włączone jednocześnie zostaje automatycznie włączony modulator OCM.

Rys. 72. Funkcjonalny schemat zastępczy OCM



Kiedy modulator jest włączony typ modulacji (suma lub iloczyn logiczny) jest ustawiany przez rejestr PORTB7.

Rys 73. Przykładowy przebieg czasowy modulatora OCM



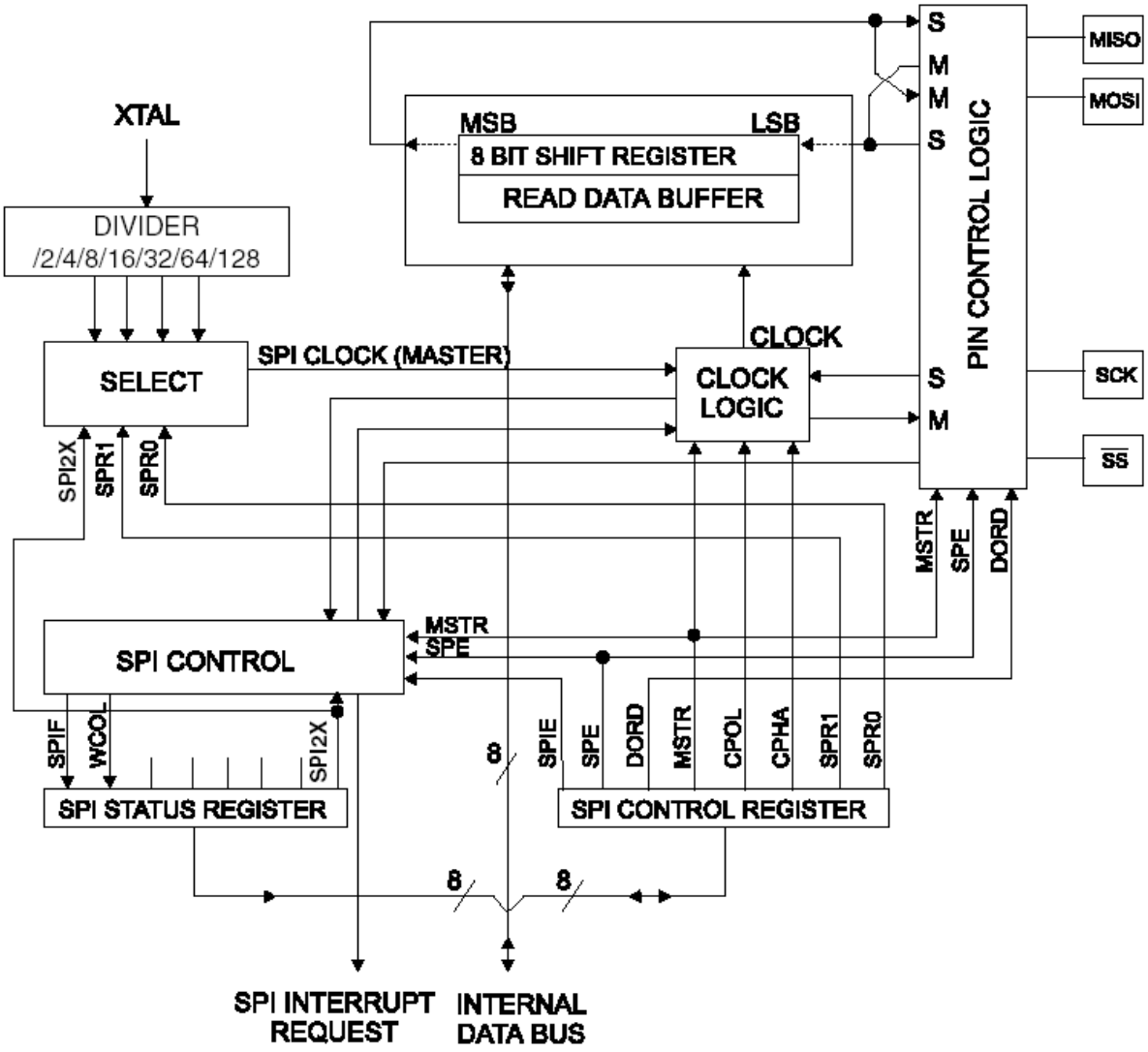
Rozdzielczosc sygnału PWM (OC1C) jest redukowana w modulacji. Czynnik redukcji jest równy ilości cykli zegara systemowego przypadających na jeden okres czestotliwości nośnrj (OC2).

SZEREGOWY INTERFEJS PERYFERYJNY (SERIAL PERIPHERIAL INTERFACE – SPI)

Szeregowy interfejs peryferyjny umożliwia bardzo szybką, synchroniczną wymianę danych pomiędzy procesorem ATmega128 a urządzeniami zewnętrznymi oraz pomiędzy poszczególnymi urządzeniami AVR. SPI posiada następujące cechy:

- operację Master oraz Slave
- możliwość jednoczesnej komunikacji w obu kierunkach, trójprzewodowy synchroniczny przesył danych
- możliwość przesyłu danych od najstarszego lub najmłodszego bitu
- siedem programowalnych szybkości przesyłu danych w bitach na sekundę
- flaga przerwania – koniec transmisji
- tryb master z podwojoną prędkością
- znacznik zabezpieczenia przed kolizją podczas zapisu
- wzbudzanie ze stanu uśpienia

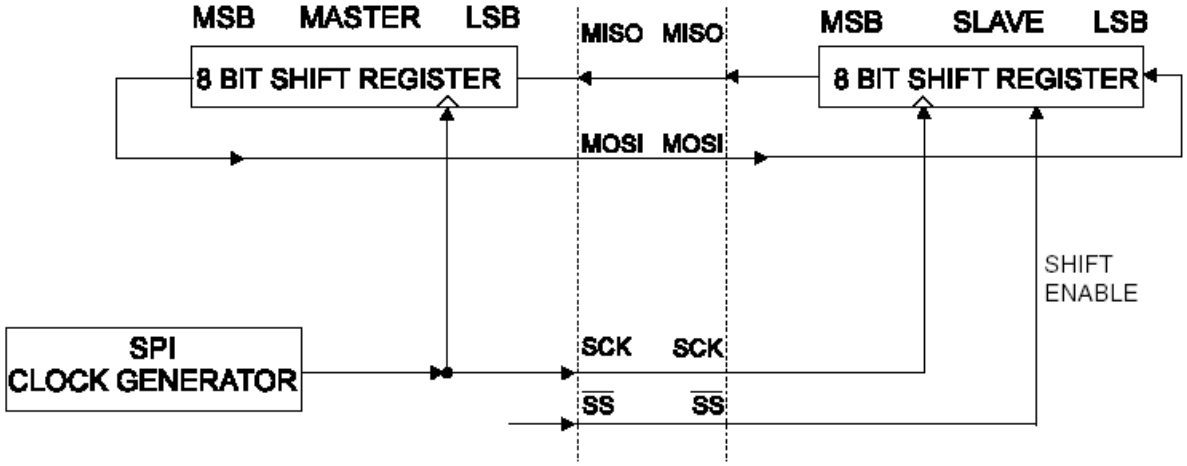
Rys. 74 Schemat blokowy SPI



Wewnętrzne połączenie jednostki centralnej Master i Slave z SPI zostało pokazane na rys. 75. System składa się z dwóch rejestrów przesuwnych oraz zegara należącego do modułu Master. SPI Master inicjuje cykl komunikacyjny kiedy wymusi stan niski na pinie Slave Select (/SS) żądanego modułu Slave. Moduły Master i Slave przygotowują dane do wysłania w swoich rejestrach przesuwnych. Z kolei moduł Master generuje impulsy zegarowe na linii SCK aby w ich takt wymienić dane. Dane zawsze są przesuwane z modułu Master do modułu Slave na linii Master Out - Slave In (MOSI), natomiast z modułu Master do modułu Slave na linii Master In - Slave Out (MISO). Po każdym pakiecie danych, moduł Master synchronizuje się z modułem Slave przez ustawienie stanu wysokiego na linii Slave Select. Jeżeli moduł jest skonfigurowany jako Master, interfejs SPI nie ma automatycznej kontroli nad linią /SS. Ten problem musi zostać rozwiązany przez użytkownika na drodze programowej zanim komunikacja się rozpocznie. Wpisanie bajtu do rejestru danych SPI (SPDR) włączy zegar, a układ przesunie 8 bitów do modułu Slave.

Po przesunięciu jednego bajtu, zegar SPI zatrzyma się ustawiając znacznik końca transmisji (SPIF). Jeżeli bit zezwolenia na przyjęcie przerwania w rejestrze SPCR jest ustawiony przerwanie zostaje zgłoszone. Moduł Master może kontynuować przesuwanie następnego bajtu poprzez wpisanie go do SPDR. Może też zakończyć transmisję pakietu danych ustawiając stan wysoki na linii Slave Select. Ostatni otrzymany bajt będzie zachowany w rejestrze buforowym, aby można go użyć później.
Jeżeli moduł jest skonfigurowany jako Slave, interfejs SPI będzie pozostawał w uśpieniu ze stanem wysokiej impedancji na linii MISO tak długo dopóki pin /SS nie znajdzie się w stanie wysokim. W tym czasie program może uaktualnić zawartość SPDR. Dane nie zostaną jednak przesunięte przez nadchodzące impulsy zegara na pinie SCK, dopóki na /SS jest stan niski. Po przesłaniu w całości jednego bajtu znacznik końca transmisji SPIF zostaje ustawiony. Jeżeli bit zezwolenia na przyjęcie przerwania w rejestrze SPCR jest ustawiony występuje żądanie przyjęcia przerwania. Moduł Slave może kontynuować umieszczanie nowych danych przeznaczonych do przesłania w SPDR przed odczytaniem przychodzących danych. Ostatni otrzymany bajt będzie zachowany w rejestrze buforowym, aby można go użyć później.

Rys. 75 Wewnętrzne połączenie Master-Slave



System jest pojedynczo buforowany w kierunku nadawania i podwójnie w kierunku wysyłania. Oznacza to że nadawany bajt nie może być wczytany do rejestru danych SPI zanim cały cykl przesuwania nie zostanie zakończony. Kiedy dane są odbierane, otrzymywany znak musi zostać przeczytany z rejestru danych SPI zanim następny znak zostanie w całości wprowadzony. W przeciwnym razie doszło by do utraty danych.
W trybie SPI Slave, układ sterujący będzie próbkował nadchodzący sygnał na pinie SCK. Aby zapewnić właściwe próbkowanie sygnału zegarowego, częstotliwość zegara SPI nie powinna nigdy przekroczyć $f_{OSC}/4$.
Kiedy SPI jest włączony, kierunki przesyłania danych linii MOSI, MISO, SCK oraz pinów /SS są przeładowane zgodnie z tabelą 69.

Tabela 69 Przeładowania pinów

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Poniższy przykładowy kod pokazuje jak zainicjować SPI w trybie Master oraz jak przygotować prostą transmisję. DDR_SPI w przykładach należy zstąpić przez aktualny rejestr kierunku danych sterujący pinami SPI. DD_MOSI, DD_MISO i DD_SCK muszą być zastąpione przez aktualne bity określające kierunek danych dla tych pinów. Np. jeżeli MOSI jest na miejscu pinu PB5, zastępujemy DD_MOSI DDB5 a DDR_SPI DDRB.

Assembly Code Example[1]

```
SPI_MasterInit:
  ; Set MOSI and SCK output, all others input
  ldi   r17,(1<<DD_MOSI)|(1<<DD_SCK)
  out   DDR_SPI,r17
  ; Enable SPI, Master, set clock rate fck/16
  ldi   r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
  out   SPCR,r17
  ret

SPI_MasterTransmit:
  ; Start transmission of data (r16)
  out   SPDR,r16
Wait_Transmit:
  ; Wait for transmission complete
  sbis  SPSR,SPIF
  rjmp  Wait_Transmit
  ret
```

C Code Example[1]

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input */
  DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF)))
    ;
}
```

Poniższy kod pokazuje jak zainicjować SPI w trybie Slave i jak przygotować prosty odbiór danych.

Assembly Code Example[1]

```
SPI_SlaveInit:
  ; Set MISO output, all others input
  ldi   r17,(1<<DD_MISO)
  out   DDR_SPI,r17
  ; Enable SPI
  ldi   r17,(1<<SPE)
  out   SPCR,r17
  ret

SPI_SlaveReceive:
  ; Wait for reception complete
  sbis  SPSR,SPIF
  rjmp  SPI_SlaveReceive
  ; Read received data and return
  in    r16,SPDR
  ret
```

C Code Example[1]

```
void SPI_SlaveInit(void)
{
  /* Set MISO output, all others input */
  DDR_SPI = (1<<DD_MISO);
  /* Enable SPI */
  SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
  /* Wait for reception complete */
  while(!(SPSR & (1<<SPIF)))
    ;
  /* Return data register */
  return SPDR;
}
```

FUNKCJONALNOŚĆ PINU /SS

Tryb Slave

Kiedy SPI jest skonfigurowany jako Slave, Slave Select jest zawsze wejściem. Kiedy /SS jest w stanie niskim, SPI zostaje uaktywniony, a MISO staje się wyjściem jeżeli tak został skonfigurowany przez użytkownika. Wszystkie inne piny są wejściami. Kiedy /SS jest w stanie wysokim, wszystkie piny są

wejściami, a SPI jest bierny co oznacza że nie będzie odbierał nadchodzących danych. Należy zauważyć że układ SPI będzie resetowany za każdym razem gdy pin /SS znajdzie się w stanie wysokim.
Pin /SS jest użyteczny przy synchronizacji pakiet/bajt aby utrzymać synchronizację pomiędzy licznikiem bitowym oraz zegarem modułu Master. Kiedy pin /SS przejdzie w stan wysok, SPI natychmiast skasuje przesyłane dane oraz układ odbiornika, jak również utraci częściowo otrzymane dane znajdujące się w rejestrze przesuwnym.

Kiedy SPI jest skonfigurowany jako Master Tryb Master (MSTR jest ustawione w SPCR), użytkownik może zadecydować o tym czy /SS będzie wejściem czy wyjściem.

Jeżeli pin /SS jest wyjściem nie wpływa na system SPI.

Jeżeli pin /SS jest wejściem należy na nim utrzymać stan wysoki aby zapewnić operacje SPI w trybie Master.

Aby uniknąć konfliktów na magistrali system SPI podejmuje następujące działania:

1. Bit MSTR w SPCR jest kasowany, a system SPI przechodzi w tryb Slave, MOSI i SCK stają się wejściami
2. Flaga SPIF w rejestrze SPSR jest ustawiona i jeżeli przerwania SPI są aktywne oraz I-ty bit w SREG jest ustawiony, procedura obsługi przerwania zacznie się wykonywać.

Dlatego kiedy sterowana przerwaniami transmisja SPI jest używana w trybie Master i istnieje możliwość, że /SS zostanie sprowadzone do stanu niskiego, przerwanie powinno zawsze sprawdzić czy bit MSTR jest ciągle ustawiony. Jeżeli bit MSTR zostanie skasowany przez wybór Slave'a musi być ustawiony przez użytkownika aby reaktywować SPI w trybie Master.

Rejestr sterujący SPI – SPCR

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
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 – SPIE: SPI Interrupt Enable**

  Ten bit powoduje że przerwanie SPI będzie wykonane jeżeli ustawiony jest bit SPIF w rejestrze SPSR oraz globalna flaga zezwolenia na przyjęcie przerwania w rejestrze SREG.
- **Bit 6 – SPE: SPI Enable**

  Jeżeli bit SPE jest ustawiony na jeden, SPI jest aktywne. Ustawienie go jest niezbędne gdy chcemy wykonywać jakiekolwiek operacje SPI.
- **Bit 5 – DORD: Data Order**

  Jeżeli DORD jest równy 1 LSB danej jest transmitowany jako pierwszy. Ustawienie DORD na 0 powoduje przesłanie MSB w pierwszej kolejności.
- **Bit 4 – MSTR: Master/Slave Select**

  1 – tryb Master, 0 – tryb Slave

Jeżeli /SS jest skonfigurowane jako wejście i jest w stanie niskim kiedy MSTR jest ustawione, MSTR zostanie wyczyszczone a SPIF w SPSR zostanie ustawione. Użytkownik będzie musiał ustawić MSTR aby ponownie uaktywnić tryb Master.

- **Bit 3 – CPOL: Clock Polarity**

  1 – SCK jest w stanie wysokim w chwilach bezczynności

  0 - SCK jest w stanie niskim w chwilach bezczynności

  Przykład na rys. 76 i 77. Funkcjonalność CPOL przedstawiona jest w poniższej tabeli.

Tabela 70 Funkcjonalność CPOL

CPOL	Leading edge	Trailing edge
0	Rising	Falling
1	Falling	Rising

- **Bit 2 – CPHA: Clock Phase**

  Ustawienie tego bitu określa aktywne zbocze SCK. Przykład na rys. 76 i 77. Funkcjonalność CPHA przedstawiona jest w poniższej tabeli.

Tabela 71 Funkcjonalność CPHA

CPHA	Leading edge	Trailing edge
0	Sample	Setup
1	Setup	Sample

- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

Bity SPR1 i SPR0 sterują prędkością urządzeń skonfigurowanych jako master. Jeżeli SPI jest w trybie Slave ich ustawienie nie daje żadnych skutków. Zależność pomiędzy SCK a częstotliwością zegara f_{OSC} przedstawia poniższa tabela.

TABELA 72 RELACJE POMIĘDZY SCK A CZĘSTOTLIWOŚCIĄ OSCYLATORA

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Rejestr stanu SPI – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

Jeżeli szeregowy transfer jest zakończony znacznik SPIF jest ustawiony. Przerwanie jest generowane jeżeli ustawiony został SPCR oraz globalne przerwania są aktywne. Jeżeli /SS jest wejściem i znajduje się w stanie niskim, podczas gdy SPI jest w trybie Master również spowoduje to ustawienie znacznika SPIF. SPIF jest kasowane sprzętowo kiedy wykonywany jest odpowiedni wektor przerwania. SPIF może zostać wykasowane także wtedy gdy po pierwszym odczycie rejestru stanu z ustawionym SPIF zażądamy dostępu do SPDR.

- **Bit 6 – WCOL: Write COLision Flag**

Bit WCOL jest ustawiony jeżeli SPDR jest zapisywany podczas transferu danych. Bit WCOL i bit SPIF są kasowane przez pierwszy odczyt rejestru stanu z ustawionym WCOL i późniejszy dostęp do rejestru danych SPI.

- **Bits 5..1 – Res: Reserved Bits**

Te bity są zarezerwowane i zawsze będą odczytywane jako 0.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

Ustawienie tego bitu na 1 częstotliwość SCK zostanie podwojona gdy SPI znajduje się w trybie Master (patrz tabela 4). Oznacza to, że minimalny okres SCK będzie równy dwóm okresom zegara jednostki centralnej. W przypadku gdy SPI został skonfigurowany jako Slave, SPI gwarantuje pracę z częstotliwością mniejszą lub równą $f_{OSC}/4$.

Interfejs SPI na Atmega128 jest również używany do programowania pamięci oraz pobierania/ładowania danych z/do pamięci typu EEPROM.

Rejestr danych – SPDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

SPDR jest rejestrem przeznaczonym do zapisu i odczytu, używa się go przy przesyłaniu danych pomiędzy plikiem rejestru i rejestrem przesuwnym SPI. Wpis to rejestru inicjuje transmisję danych. Odczyt rejestru powoduje rozpoczęcie odczytywania z bufora przesuwnego rejestru nadajnika (???).

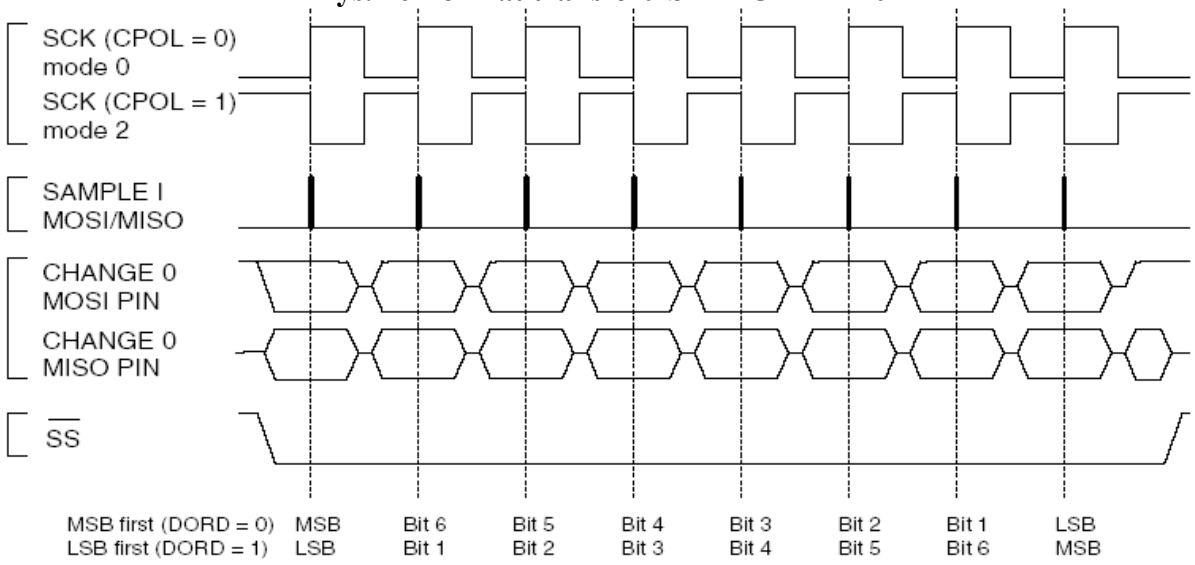
TRYBY DANYCH

Istnieją 4 kombinacje fazy i polaryzacji SCK odnoszące się do danych w postaci szeregowej, które określają bity sterujące CPHA oraz CPOL. Format przesyłu danych SPI pokazują rys. 76 i 77. Bity danych są przesuwane i zatrzaskiwane na przeciwnych zboczach sygnału SCK, zapewniając tym samym wystarczającą ilość czasu na ustabilizowanie się sygnału danych.

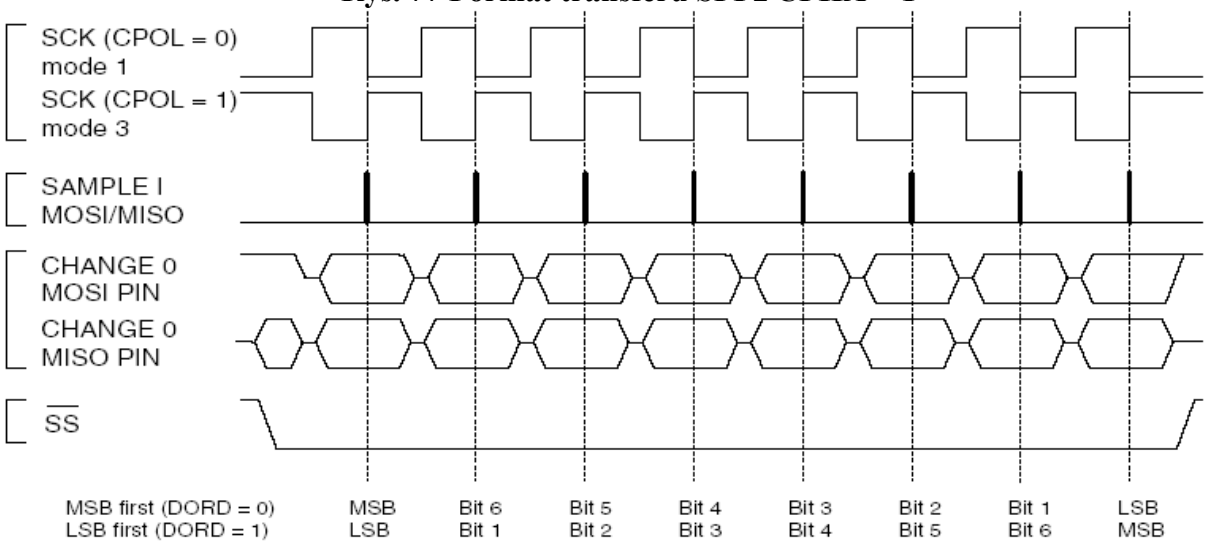
Tabela 73 Funkcjonalność CPOL

	Leading edge	Trailing edge	SPI mode
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)	0
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)	1
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)	2
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)	3

Rys. 76 Format transferu SPI z CPHA = 0



Rys. 77 Format transferu SPI z CPHA = 1



USART

Uniwersalny synchroniczny i asynchroniczny odbiornik i nadajnik (USART) jest wysoce elastycznym szeregowym urządzeniem do transmisji danych. Jego główne cech to:

- Całkowite, dwustronne operacje (niezależne rejestry szeregowego odbiornika i nadajnika)
- Operacje asynchroniczne i synchroniczne.
- Operacje synchroniczne taktowane zegarem Master-Slave
- Generator szybkiej transmisji o wysokiej rozdzielczości
- Wsparcie dla szeregowych ramek o 5,6,7,8, lub 9 bitach danych i 1 lub 2 bitach stopu
- Parzyste lub nieparzyste generatory parzystości i sprawdzanie parzystości wspierane sprzętowo
- Wykrycie przekroczenia danych
- Detekcja błędów ramki
- Filtrowanie zakłóceń zawierające wykrywanie fałszywych bitów startu i binarny filtr dolnoprzepustowy
- Trzy oddzielne przerwania TX zakończenie, pusty rejestr danych TX, zakończenie RX
- Tryb komunikacji wieloprocesorowej
- Tryb komunikacji asynchronicznej o podwójnej prędkości

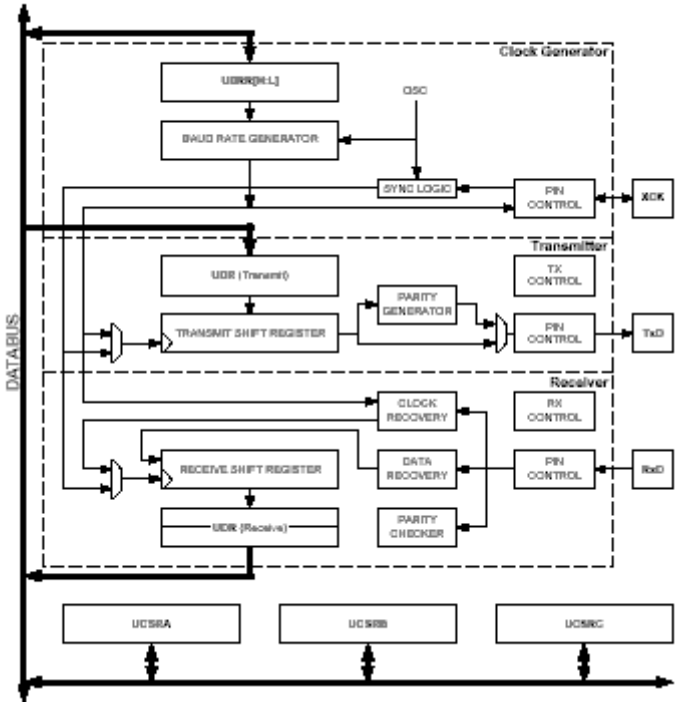
- **Podwójny USART**

ATmega128 ma dwa USART'y: USART0 i USART1. Funkcjonalność obu USART'ów jest opisana poniżej. USART0 i USART1 mają różne rejestry I/O co jest pokazane w „streszczeniu rejestrów" na stronie 341. Należy zauważyć, że w trybie kompatybilności z ATmega103, USART1 jest niedostępny, jak i rejestry UBRROH i UCRSOC. To oznacza, że w trybie kompatybilności z ATmega103, ATmega128 wspiera tylko asynchroniczną transmisję przez USART0.

- **Krótki opis**

Uproszczony diagram blokowy USART'u przedstawia rysunek 78. Dostępne rejestry I/O CPU tak jak ich wyjścia są pogrubione.

Figure 78. USART Block Diagram



Uwaga: Należy odnieść się do rysunku 1 na stronie 2, tabeli 36 na stronie 72 i tabeli 93 na stronie 75 gdzie zostało opisane umiejscowienie wyjść.

Prostokąty zaznaczone linią przerywaną w diagramie blokowym dzielą USART na trzy główne części (patrząc od góry): generator zegara, nadajnik, odbiornik. Rejestry kontroli są dzielone między tymi trzema jednostkami. Generacja logicznego zegara składa się z logicznej synchronizacji dla wejścia zewnętrznego zegara wykorzystywanego przez synchroniczne operacje zależne i generator szybkości transmisji. Nóżka XCK (zegar transferu) jest tylko wykorzystywana w trybie synchronicznej transmisji. Nadajnik składa się z pojedynczego buforu do zapisu, szeregowego rejestru przesunięcia, generatora parzystości i kontroli logiczne obsługującej różne szeregowe formaty ramek. Bufor do zapisu umożliwia ciągły transfer danych bez opóźnień między ramkami. Odbiornik jest najbardziej złożoną częścią w module USART'u ze względu na zegar i jednostki odzyskiwania danych. Jednostki odzyskiwania danych są wykorzystywane w pobieraniu asynchronicznym danych. Ponad to odbiornik posiada sprawdzanie parzystości, kontrolę logiczną, rejestr przesunięcia i dwa poziomy buforów odbiorczych (UDR). Odbiornik wspiera te same rodzaje ramek co nadajnik, może wykryć błąd ramki i przekroczenie danych i błąd parzystości.

- **Kompatybilność AVR USART z AVR UART**

USART jest w pełni kompatybilny z AVR UART co do:

- Lokacji bitów we wszystkich rejestrach USART'u
- Generacji szybkości transmisji
- Operacji nadajnika
- Funkcjonalności buforu nadajnika

- Operacji odbiornika

Jednakże buforowanie odbiornika ma dwa ulepszenia, które wpływają na kompatybilność w niektórych specjalnych przypadkach:

- Został dodany drugi rejestr buforujący. Obydwa te bufory działają na zasadzie kolejki FIFO. Dlatego UDR może być odczytywane tylko raz na każde przyjście danych. Ważniejszy jest jednak fakt, że flagi błędów (FE i DOR) i dziewiąty bit danych (RXB8) są buforowane z danymi w buforze odbiornika. Dlatego bit statusu musi zawsze być odczytywany przed odczytem z rejestru UDR. W przeciwnym wypadku status błędu zostanie zgubiony wraz ze statusem bufora.
- Rejestr przesuwny odbiornika może teraz działać jako trzeci poziom bufora. Zostało to zapewnione poprzez pozostawienie danych w tym rejestrze (rysunek 78) jeżeli bufory są zapełnione, do czasu wykrycia nowego bitu startu. Dlatego USART jest bardzie wytrzymały na przeładowanie danymi.

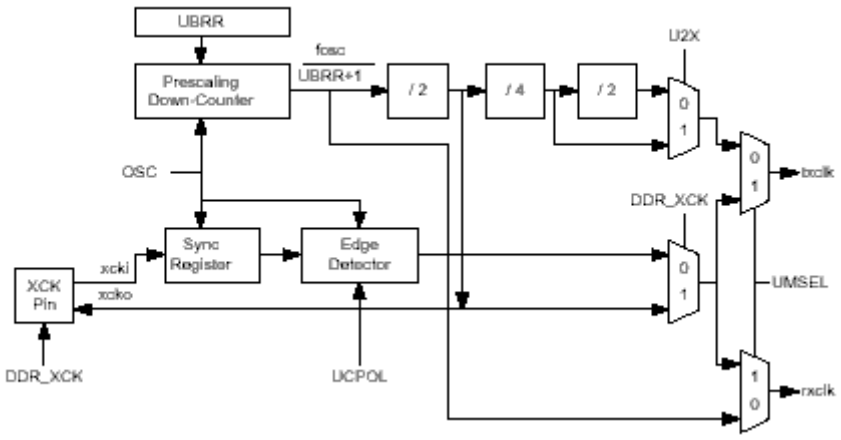
Następne bity kontrolne mają zmienione nazwy, ale mają ta samą funkcjonalność i umiejscowienie w rejestrach:

- CHR9 został zmieniony na UCSZ2
- OR na DOR

- **Generacja zegarów**

Logiczna generacja zegarów generuje podstawowy zegar dla nadawania i odbioru. USART wspiera cztery tryby operacji czasowych: normalny asynchroniczny, asynchroniczny o podwójnej szybkości, nadrzędny synchroniczny i podrzędny synchroniczny. Bit UMSEL w rejestrze statusu i kontroli C (UCSRC) wybiera między trybem asynchronicznym i synchronicznym. Podwójna szybkość (w trybie asynchronicznym) jest kontrolowana przez U2X znajdujący się w rejestrze UCSRA. Podczas korzystania z trybu synchronicznego (UMSEL=1), rejestr kierunku przepływu danych sprawdza dal wyjścia XCK (DDR_XCK) czy źróło zegara jest wewnętrzne (nadrzędne) czy zewnętrzne (podrzędne). Wyjście XCK jest tylko aktywne w trybie synchronicznym.

Figure 79. Clock Generation Logic, Block Diagram



Opis sygnałów:

txclk zegar nadajnika (wewnętrzny sygnał)
rxclk podstawa zegara dla odbiornika

xclk wejście z XCK (wewnętrzny sygnał) Wykorzystywany w synchronicznych podrzędnych operacjach.

xclo Wyjście zegara dla XCK (wewnętrzny sygnał). Wykorzystywany w synchronicznych nadrzędnych operacjach.

fosc częstotliwość wejścia XTAL (zegar systemowy)

- **Wewnętrzna generacja zegarów – generator szybkości transmisji**

Wewnętrzny generator zegara jest wykorzystywany w asynchronicznych i asynchronicznych trybach nadrzędnych operacji. Opis w tej sekcji odnosi się do rysunku 79.

Rejestr szybkości transferu (UBRR) i przyłączony do niego licznik(zliczający w dół) funkcjonuj ą jako programowalny prescaler lub generator szybkości transmisji. Licznik, napędzany systemowym zegarem (fosc), jest ładowany wartości z UBRR za każdym razem, kiedy doliczył do zera lub kiedy dokonano zapisu do rejestru UBRRL. Zegar jest wywoływany(generated) za każdym razem kiedy licznik dojdzie do zera. Ten zegar jest generatorem szybkości transferu dla wyjścia zegara (=fosc/(UBRR+1)). Nadajnik dzieli wartości pojawiające się na wyjściu zegara szybkości transmisji przez 2, 4, 8, 16 zależnie od trybu, w którym się znajduje. Wyjście to jest podłączane bezpośrednio do zegara odbiornika i modułów odświeżania danych. Jednakże, moduły odświeżania korzystają z statycznej maszyny, która wykorzystuje 2, 8 lub 16 stanów w zależności od trybu ustawionego przez bity UMSEL, U2X i DDR_XCK.

Tabela 74 zawiera równania służące obliczeniu szybkości transferu (bity na sekundę) oraz obliczeniu wartości UBRR dla każdego trybu operacji wykorzystującej wewnętrzne źródło generujące zegar.

Tryb operacji	Szybkość transferu[1]	Wartość UBRR
Normalny, asynchroniczny tryb (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchroniczny tryb o podwójnej prędkości (U2X =1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchroniczny tryb nadrzędny	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Uwaga: 1.szybkość transferu jest definiowana w bitach na sekundę
f_{OSC} – częstotliwość systemowego zegara

Przykłady wartości UBBR dla niektórych częstotliwości zegarowych podane sa w tabeli 82 na stronie 189.

- **Operacje o podwójnej szybkości – U2X**

Szybkość transferu może zostać zdwojona poprzez ustawienie bitu U2X w UCSRA. Ustawienie tego bitu daje efekt tylko dla asynchronicznych operacji. W przypadku operacji synchronicznych należy ustawić ten bit na zero.

Ustawienie tego bitu spowoduje zmniejszenie dzielnika częstotliwości z 16 na 8, efektywnie podwajając szybkość transferu dla asynchronicznej komunikacji. Należy zauważyć, że odbiornik w tym przypadku będzie używał połowy próbek w celu próbkowania danych i odświeżania zegara i dlatego bardziej odpowiednie ustawienia szybkości transferu i zegara systemowego jest wymagane w tym trybie pracy. Dla nadajnika nie ma dolnych ograniczeń.

- **Zewnętrzny zegar**

Zewnętrzny zegar jest wykorzystywany w przypadku trybu synchronicznego podrzędnego operacji. Opis odnosi się do rysunku 79.

Wejście zewnętrznego zegara jest próbkowane z wyjścia XCK poprzez rejestr synchronizacji w celu zmniejszenia szansy na stan stabilny (meta-stability). Wyjście z rejestru synchronizacji musi przejść przez detektor zboczy nim może zostać wykorzystany przez nadajnik i odbiornik. Proces ten wprowadza dwa okresy opóźnienia w CPU i dlatego maksymalna zewnętrzna częstotliwość zegara XCK jest limitowana przez poniższe równania:

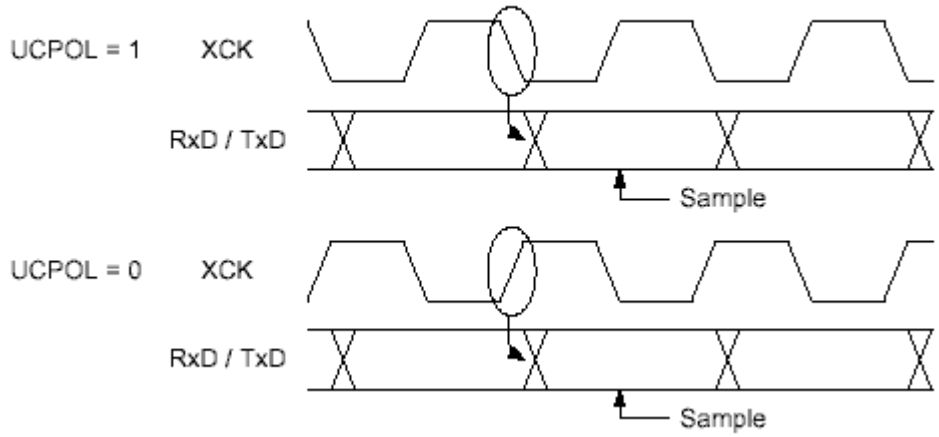
$$f_{XCK} < \frac{f_{OSC}}{4}$$

Należy zauważyć, że f_{OSC} zależy od stabilności źródła systemowego zegara. Dlatego radzi się aby dodać jakiś margines wahań, aby uniknąć straty danych w wyniku wahań częstotliwości.

- **Synchroniczne operacje na zegarze**

Pracując w trybie synchronicznym (UMSEL = 1), nóżka XCK będzie wykorzystywana jako wejście zegara (podrzędny układ) lub wyjście zegara (nadrzędny układ). Zależność pomiędzy zboczami zegara i próbkowaniem danych lub zmianą danych jest takie samo. Podstawową zasadą jest by wejście danych (on RxD) było próbkowane na przeciwnym zboczu zegara XCK do zbocza testującego wyjście z danymi (TxD) zmienianymi.

Figure 80. Synchronous Mode XCK Timing.



Bit UCPOL z UCRSC wybiera które ze zboczy zegara XCK jest wykorzystywane do próbkowania danych i które jest wykorzystywane do zmiany danych. Jak pokazuje rysunek 80, kiedy UCPOL jest zerem dane będą zmieniane na rosnącym zboczu XCK a próbkowane na

opadającym zboczu. Jeżeli UCPOL jest ustawiony dane będą zmieniane na opadającym zboczu XCK i próbkowane na rosnącym.

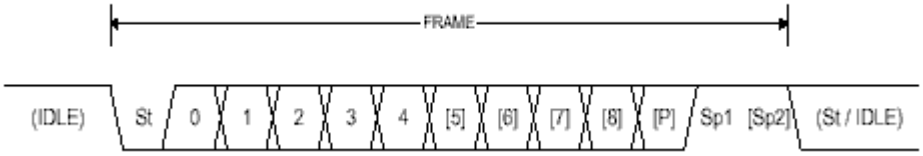
- **Formaty ramek**

Szeregowa ramka ma jeden znak będą cy bitem danych oraz bit asynchronizacji (bit startu lub stopu) i opcjonalnie bit parzystości służący sprawdzeniu błędów. USART akceptuje wszystkie 30 następujący kombinacji jako poprawne formaty ramek:

- 1 bit startowy
- 5, 6, 7, 8 lub 9 bitów danych
- brak, bit parzystości lub nieparzystości
- 1 lub 2 bity stopu

Ramka rozpoczyna się bitem startu, po którym następuje najmniej znaczący bit danych. Następne przychodzą bity danych, do 9, zakończone bitem najbardziej znaczącym. Jeżeli jest odblokowany bit parzystości jest on dołączany po bitach danych a przed bitem stopu. Po kompletnie przesłanej ramce można bezpośrednio wysyłać następną lub linia komunikacyjna może zostać ustawiona w trzeci stan (IDLE).

Figure 81. Frame Formats



St bit startowy – zawsze 0
(n) bity danych(od 0 do 8)
P bit parzystości
Sp bit stopu – zawsze 1
IDLE brak transferu na liniach komunikacyjnych (RxD lub TxD) .Linia IDLE musi być w trzecim stanie.

Format ramek używany przez USART jest ustawiany poprzez bity UCSZ2..0 i UMP1..0 oraz USBS w rejestrach UCSRB i UCSRC. Nadajnik i odbiornik wykorzystują te same ustawienia. Należy zauważyć, że zmiana ustawień powoduje zniekształcenie trwającej komunikacji zarówno dla odbiornika jak i nadajnika.

Bity znaku USART (UCSZ2..0) wybierają liczbę bitów danych w ramce. Bity UPM1..0 ustawiają tryb parzystości. Wybór pomiędzy jednym lub dwoma bitami stopu jest dokonywany przez bit USBS (bit wyboru stopu). Odbiornik ignoruje drugi bit stopu. Błąd ramki (FE) będzie więc wykryty tylko w przypadku kiedy pierwszy bit stopu jest zerem.

- **Obliczanie bitu parzystości**

Bit parzystości jest obliczany poprzez wykonanie operacji xor na wszystkich bitach danych. Jeżeli ten bit jest wykorzystywany jako bit nieparzystości wynik jest negowany binarnie. Relacja między bitem parzystości i bitami danych jest następująca:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even} bit parzystości wykorzystywany jako bit parzystości
P_{odd} bit parzystości wykorzystywany jako bit nieparzystości
d_n n-ty bit znaku danych

Jeżeli jest ten bit wykorzystywany, jest on umieszczony między ostatnim bitem danych , a bitem stopu.

- **Inicjalizacja USART**

USART musi zostać zainicjalizowany przed rozpoczęciem komunikacji. Proces inicjalizacji normalnie składa się z ustawienia szybkości transferu, formatu ramki i odblokowaniu nadajnika lub odbiornika w zależności czego chcemy używać. W celu wykrycia przerwania wywołanego przez operacje USART, globalna flaga przerwań powinna być wyzerowana podczas inicjalizacji. Przed dokonywaniem zmian szybkości transferu lub formatu ramki należy się upewnić czy nie jest wykonywany właśnie jakaś transmisja. Flaga TxC może być wykorzystana do sprawdzenia czy nadajnik ukończył transmisję, a flaga RxC do sprawdzenia czy nie pozostały jeszcze jakieś dane w buforze nadajnika. Należy pamiętać o wyzerowaniu flagi TxC przed każdą transmisją (nim UDR jest zapisany) jeżeli jest on wykorzystywany w tym celu.

Następujące kody pokazują funkcje inicjalizujące w Asemblerze i C. Przykłady te zakładają asynchroniczne operacje wykorzystujące odpytywanie (wszystkie przerwania powinny być zablokowane) i stały format ramek. Szybkość transferu jest zwracana jako parametr funkcji. W przypadku asemblera zakłada się, że parametr szybkości transmisji jest dany rejestrami r17 i r16.

Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  out   UBRRH, r17
  out   UBRRL, r16
  ; Enable receiver and transmitter
  ldi   r16, (1<<RXEN)|(1<<TXEN)
  out   UCSRB,r16
  ; Set frame format: 8data, 2stop bit
  ldi   r16, (1<<USBS)|(3<<UCSZ0)
  out   UCSRC,r16
  ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<USBS)|(3<<UCSZ0);
}
```

Uwaga: 1. Przykłady kodu zakładają, że specyficzne nagłówki zostały dołączone. Dla rejestrów I/O umiejscowionych w rozszerzonej mapie rozkaz I/O, IN, OUT, SBIS, SBIC, CBI i SBI muszą być zamienione przez rozkazy umożliwiające dostęp do rozszerzonej mapy pamięci. Zazwyczaj są to rozkazy LDS i STS połączone z SBRC, SBRS, SBR i CBR.

Dodatkowe zaawansowane metody inicjalizacji mogą zawierać formaty ramek jako parametry, blokowanie przerwań itp. Jednakże, wiele programów wykorzystuje stałe ustawienia rejestrów bodów i kontroli i dla tych typów oprogramowania kody inicjalizacji mogą być umieszczone bezpośrednio w głównej procedurze lub być połączone z kodami inicjalizującymi inne moduły I/O.

- **Transmisja danych – nadajnik USART**

Nadajnik USART jest odblokowywany poprzez ustawienie bitu umożliwiającego nadawanie (TXEN) w rejestrze UCSRB. Kiedy nadajnik jest odblokowany, normalne operacje na nóżkach TxD są przeciążone poprzez USART i funkcjonują jako wyjście szeregowe USART'a. Szybkość transmisji , tryb operacji i rodzaj ramki muszą zostać ustawione przed rozpoczęciem transmisji. W przypadku wykorzystywania operacji synchronicznych zegar na wyjściach XCK będzie przeciążony i wykorzystywany jako zegar transmisji.

  - **Wysyłanie ramek z 5 do 8 bitami danych**

Transmisja danych jest inicjowana przez załadowanie bufora transmisji danymi, które maja być wysłane. CPU może załadować bufor zapisując do lokacji UDR I/O. Zbuforowane dane w buforze nadajnika będą przeniesione do rejestru przesuwnego kiedy rejestr ten będzie gotowy do

wysłania nowej ramki. Rejestr przesuwny jest załadowany nowymi danymi jeżeli jest w stanie IDYL (nie ma żadnej trwającej transmisji) lub natychmiastowo po wysłaniu ostatniego bitu poprzedniej transmisji. Kiedy rejestr przesuwny jest załadowany nowymi danymi , wyśle dane z szybkością określoną przez rejestr szybkości, bit U2X lub przez XCK co zależy od trybu operacji.

Poniższe przykłady kodu pokazują proste funkcje transmisji bazujące na sprawdzaniu flagi rejestru danych (UDRE). Używając ramek z mniejszą ilością bitów danych niż 5, bity najbardziej znaczące są ignorowane. USART musi być inicjalizowany przed wykorzystywaniem tych funkcji. Dla kodu w asemblerze zakłada się, że dane, które mają zostać wysłane są w rejestrze r16.

Assembly Code Example[1]

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis  UCSRA,UDRE
  rjmp  USART_Transmit
  ; Put data (r16) into buffer, sends the data
  out   UDR,r16
  ret
```

C Code Example[1]

```
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
        ;
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Uwaga: 1. Przykłady kodu zakładają, że specyficzne nagłówki zostały dołączone. Dla rejestrów I/O umiejscowionych w rozszerzonej mapie rozkaz I/O, IN, OUT, SBIS, SBIC, CBI i SBI muszą być zamienione przez rozkazy umożliwiające dostęp do rozszerzonej mapy pamięci. Zazwyczaj są to rozkazy LDS i STS połączone z SBRC, SBRS, SBR i CBR.

Funkcja ta oczekuje na opróżnienie bufora nadajnika poprzez proste sprawdzanie flagi UDRE, przed załadowaniem bufora nowymi danymi. Jeżeli przerwanie oznaczające pusty rejestr danych jest wykorzystywane, procedura obsługi przerwania zapisuje dane do bufora.

- **Wysyłanie ramek z 9 bitami danych**

Jeżeli są wykorzystywane 9-bitowe znaki (UCSZ = 7) dziewiąty bit musi być zapisany do bitu TXB8 w rejestrze UCSRB nim młodszy bajt tego znaku zostanie zapisany do UDR. Następujące przykłady kodu przedstawiają funkcje służące do transmisji 9 bitowychznaków. Dla kodu w asemblerze zakłada się, że dane składowane są w rejestrach r16 i r17.

Assembly Code Example[1]

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis  UCSRA,UDRE
  rjmp  USART_Transmit
  ; Copy 9th bit from r17 to TXB8
  cbi   UCSRB,TXB8
  sbrc  r17,0
  sbi   UCSRB,TXB8
  ; Put LSB data (r16) into buffer, sends the data
  out   UDR,r16
  ret
```

C Code Example

```
void USART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
        ;
  /* Copy 9th bit to TXB8 */
  UCSRB &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSRB |= (1<<TXB8);
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Uwaga: 1.Te funkcje przesyłu powinny służyć jako główne funkcje do przesyłania danych. Mogą zostać zoptymalizowane jeżeli zawartość UCSRB będzie statyczna, np.: tyko bit TXB8 rejestru UCSRB jest wykorzystywany po inicjalizacji.
Dla rejestrów I/O umiejscowionych w rozszerzonej mapie rozkaz I/O, IN, OUT, SBIS, SBIC, CBI i SBI muszą być zamienione przez rozkazy umożliwiające dostęp do rozszerzonej mapy pamięci. Zazwyczaj są to rozkazy LDS i STS połączone z SBRC, SBRS, SBR i CBR.

Dziewiąty bit może być wykorzystywany jako wskazanie na adres ramki kiedy korzystamy z trybu wieloprocesorowej komunikacji lub dla obsługi innych protokołów jak dla przykładowej synchronizacji.

- **Flaga transmisji i przerwania**

Nadajnik USART na dwie flagi, które wskazują stan: Pusty rejestr danych (UDRE) i ukończona transmisja (TXC). Obydwie flagi mogą być wykorzystywane do generowania przerwań.

Flaga pusty rejestr danych wskazuje, czy bufor nadajnika jest gotowy do otrzymania nowych danych. Bit ten jest ustawiany kiedy bufor nadajnika jest pusty i zerowany kiedy bufor nadajnika zawiera dane, które jeszcze nie zostały przekazane do rejestru przesuwnego. W celu zapewnienia kompatybilności z przyszłymi urządzeniami należy zerować ten bit przy każdym zapisie do rejestru UCSRA.

Kiedy bit odblokowujący przerwania ze względu na pusty rejestr danych (UDRIE) w rejestrze UCSRB jest zapisany jedynką, przerwanie to będzie wywoływane tak długo, aż bit UDRE nie zostanie ustawiony (oczywiście, przy globalnej fladze przerwań umożliwiającej obsługę przerwania). UDRE jest zerowane przez zapisanie UDR. Kiedy wykorzystywana jest transmisja danych przez przerwania procedura obsługi przerwania oznaczającego pusty bufor nadajnika musi albo zapisywać nowe dane do UDR w celu wyzerowania UDRE lub blokować przerwanie pustego bufora nadajnika, w przeciwnym wypadku nowe przerwanie zostanie zgłoszone zaraz po wykonaniu procedury obsługi poprzedniego.

Flaga ukończona transmisja (TXC) jest ustawiana kiedy całe dane zostały przesunięte poza rejestr przesuwny i nic nie oczekuje w buforze nadajnika. Flaga ta jest automatycznie zerowana kiedy przerwanie zgłaszające ukończenie transmisji jest generowane i później obsługiwane lub może zostać wyzerowane poprzez wpisanie jedynki do odpowiednich bitów. Flaga ta jest użyteczna w interfejsie komunikacji half-duplex (np. standard RS485) gdzie oprogramowanie przesyłające dane musi wejść w tryb odbioru danych i zwolnić magistralę komunikacyjną zaraz po ukończeniu transmisji.

Kiedy bit umożliwiający przerwanie informujące o zakończeniu transmisji (TXCIE) w UCSRB jest ustawiony, przerwanie informujące o tym zdarzeniu zostanie zgłoszone kiedy flaga TXC zostanie ustawiona (jeżeli globalna flaga przerwań nie blokuje ich). Kiedy wykorzystuje się przerwania, procedura obsługi nie musi zerować flagi TXC,jest to robione automatycznie podczas wykonywania przerwania.

- **Generator parzystości**

Generator parzystości oblicza bit parzystości dla szeregowych ramek danych. Kiedy bit ten jest odblokowany (UPM1 = 1) nadajnik kontroli logicznej umiejscawia bit parzystości między ostatnim bitem danych a pierwszym bitem stopu w wysyłanej ramce.

- **Blokowanie nadajnika**

Blokowanie nadajnika (ustawienie TXEN na zero) nie będzie miało skutku dopóki trwająca i nie zakończona transmisja jest ukończona, np.: kiedy rejestr przesuwny nadajnika i rejestr buforu nadajnika nie zawierają danych do transmisji. Kiedy nadajnik jest zablokowany, nadajnik nie będzie dłużej przeciążał wyjścia TxD.

- **Odbieranie danych – odbiornik USART**

Odbiornik USART jest odblokowywany poprzez ustawienie bitu umożliwiającego odbiór (RXEN) w rejestrze UCSRB. Kiedy odbiornik jest odblokowany, normalne operacje na nóżkach RxD są przeciążone przez USART i funkcjonują jako wejście szeregowe USART'a. Szybkość transmisji, tryb operacji i rodzaj ramki muszą zostać ustawione przed rozpoczęciem transmisji. W przypadku wykorzystywania operacji synchronicznych zegar na wyjściach XCK będzie przeciążony i wykorzystywany jako zegar transmisji.

- **Odbieranie ramek z 5 do 8 bitami danych**

Odbiornik zaczyna odbierać kiedy wykryje poprawny bit startu. Każdy bit następujący po nim będzie próbkowany przez szybkość transferu i zegar XCK i przesuwany w przesuwnym rejestrze odbiornika dopóki nie nadejdzie pierwszy bit stopu. Drugi bit stpu będzie zignorowany przez odbiornik. Kiedy zostanie odebrany pierwszy bit stopu np.: kompletna szeregowa ramka znajduje się w przesuwnym rejestrze odbiornika, zawartość tego rejestru zostanie przeniesiona do bufora odbiornika. Bufor odbiornika może następnie być przeczytany poprzez odczytanie lokacji UDR I/O.

Następujący przykładowy kod pokazuje prostą funkcję odbierająca dane bazującą na odpytywaniu flagi ukończenia odbioru(RXC). Wykorzystując mniej niż 8 bitów danych najstarsze bity odczytu danych są maskowane zerem przez UDR. USART musi być zainicjalizowany nim rozpocznie się wykonywanie tej funkcji.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis  UCSRA, RXC
  rjmp  USART_Receive
  ; Get and return received data from buffer
  in    r16, UDR
  ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
        ;
  /* Get and return received data from buffer */
  return UDR;
}
```

Uwaga: 1. Przykłady kodu zakładają, że specyficzne nagłówki zostały dołączone. Dla rejestrów I/O umiejscowionych w rozszerzonej mapie rozkaz I/O, IN, OUT, SBIS, SBIC, CBI i SBI muszą być zamienione przez rozkazy umożliwiające dostęp do rozszerzonej mapy pamięci. Zazwyczaj są to rozkazy LDS i STS połączone z SBRC, SBRS, SBR i CBR.

Funkcja ta oczekuje na dane, aż będą w buforze odbiornika poprzez sprawdzanie flagi RXC przed odczytem bufora i zwróceniem wartości.

- **Odbieranie ramek z 9 bitami danych**

Jeżeli znaki dziewięcio-bitowe są używane (USCZ = 7) dziewiąty bit musi być odczytany z bitu RXB8 w UCSRB przed odczytem najmłodszych bitów z UDR. Zasada ta odnosi się również do statusu flag FE, DOR i UPE. Najpierw należy odczytać status z UCSRA, potem dane z UDR, Odczytując lokacje UDR I/O zmieni stan buforu odbiornika FIFO i konsekwentnie bity TXB8, FE, DOR i UPE, które są tam przechowywane.

Następujące przykłady kodu pokazują proste funkcje odbierające dane obsługujące zarówno 9 bit jak i bity statusu.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis  UCSRA, RXC
  rjmp  USART_Receive
  ; Get status and 9th bit, then data from buffer
  in    r18, UCSRA
  in    r17, UCSRB
  in    r16, UDR
  ; If error, return -1
  andi  r18,(1<<FE)|(1<<DOR)|(1<<UPE)
  breq  USART_ReceiveNoError
  ldi   r17, HIGH(-1)
  ldi   r16, LOW(-1)
USART_ReceiveNoError:
  ; Filter the 9th bit, then return
  lsr   r17
  andi  r17, 0x01
  ret
```

C Code Example[1]

```
unsigned int USART_Receive( void )
{
  unsigned char status, resh, resl;
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
        ;
  /* Get status and 9th bit, then data */
  /* from buffer */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* If error, return -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
    return -1;
  /* Filter the 9th bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

Uwaga: 1. Przykłady kodu zakładają, że specyficzne nagłówki zostały dołączone. Dla rejestrów I/O umiejscowionych w rozszerzonej mapie rozkaz I/O, IN, OUT, SBIS, SBIC, CBI i SBI muszą być zamienione przez rozkazy umożliwiające dostęp do rozszerzonej mapy pamięci. Zazwyczaj są to rozkazy LDS i STS połączone z SBRC, SBRS, SBR i CBR.

Przykład funkcji odbierającej dane odczytuje wszystkie rejestry I/O do zestawu rejestrów nim jakiekolwiek obliczenia zostaną wykonane. Daje to optymalne wykorzystanie bufora odbiornika odkąd odczyt lokacji bufora będzie wolny, aby przyjąć nowe dane jak najszybciej.

- **Odbieranie konkurujących flag i przerwań**

Odbiornik USART ma jedną flagę, która wskazuje stan odbiornika.

Flaga ukończenia odbioru (RXC) wskazuje nie przeczytane dane w buforze odbiornika. Jest ona wtedy jedynką i zerem kiedy bufor jest pusty. Kiedy odbiornik jest zablokowany (RXEN = 0) bufor odbiornika będzie opróżniony i bit RXC stanie się zerem.

Kiedy flaga odblokowująca przerwanie ukończenia odbioru (RXCIE(w UCSRB jest ustawiona, przerwanie ukończenia odbioru USART będzie wykonywane tak długo jak flaga RXC jest ustawiona (przy odblokowanym globalnym bicie przerwań). Kiedy receptor przerwań jest wykorzystywany, obsługa przerwania musi odczytywać dane odebrane z UDR w celu wyzerowania flagi RXC w przeciwnym wypadku nowe przerwanie zostanie wygenerowane po skończeni obsługi poprzedniego.

- **Odbieranie flag błędów**

Odbiornik USART na trzy flagi błędów: błąd ramki (FE), przekroczenie danych (DOR) i błąd parzystości (UPE). Wszystkie można odczytać z rejestru UCSRA. Flagi błędów są umieszczane w buforze odbiornika wraz z ramką, dla której wskazują status błędu. Ze względu na buforowanie flag błędu, USCRA musi zostać odczytane przed buforem odbiornika, ponieważ odczyt lokacji UDR I/O zmienia lokacje buforu odczytywanego. Innym równaniem dla flag błędu jest to, że nie mogą one zostać zmienione poprzez oprogramowanie zapisujące pod te adresy. Jednakże, wszystkie flagi muszą być ustawione na zero kiedy UCSRA jest zapisywany dla zapewnienia kompatybilności z przyszłymi implementacjami USART'a. Żadna z flag błędów nie może generować przerwań.

Błąd ramki (FE) wskazuje stan pierwszego bitu stopu w następnej do odczytu ramce przechowywanej w buforze odbiornika. Jest ona zerem jeżeli bit stopu został poprawnie odczytany (1) lub jest jedynką kiedy bit ren nie był odczytany poprawnie (0). Flaga ta może zostać wykorzystana do detekcji braku synchronizacji, warunku przerwania i obsługi protokołu. Na flagę FE nie mają wpływu ustawienia bitu USBS e UCSRC odkąd odbiornik ignoruje wszystkie, poza pierwszym, bity stopu. W celu zapewnienia kompatybilności z przyszłymi urządzeniami, należy ten bit zawsze zerować przy zapisie do UCSRA.

Błąd przekroczenia danych wskazuje na utratę danych na skutek zapchania bufora odbiornika. Zachodzi to kiedy bufor odbiornika jest pełny, jest zapisywany nowy znak w rejestrze przesuwnym odbiornika i został wykryty nowy bit startu. Jeżeli bit ten jest ustawiony oznacza to, że zostało zgubionych jedna lub więcej ramek między ramką ostatnio odczytaną z UDR i następną do odczytu. W celu zapewnienia kompatybilności z przyszłymi urządzeniami należy zawsze zerować ten bit podczas zapisu do UCSRA. Flaga DOR jest zerowana kiedy otrzymana ramka został bez problemów przeniesiona z rejestru przesuwnego do bufora odbiornika.

Flaga błędu parzystości wskazuje, że następna odebrana ramka ma błąd parzystości przy odbiorze. Jeżeli sprawdzanie parzystości nie jest odblokowane bit UPE będzie zawsze wyzerowany. W celu zapewnienia kompatybilności z przyszłymi urządzeniami zawsze należy zerować ten bit przy zapisie do UCSRA. Więcej szczegółów w „Obliczanie bitu parzystości" na stronie 171 i „Kontroler parzystości" na stronie 178.

- **Sprawdzanie parzystości**

Kontroler parzystości jest aktywny kiedy bit wysokiego trybu Parzystości USART (UPM1) jest ustawiony). Typ kontrolera parzystości jest ustawiany przez bit UPM0. Kiedy odblokowany, kontroler parzystości oblicza parzystość przychodzących bitów danych i porównuje rezultat z bitem parzystości w szeregowej ramce. Rezultat porównania jest przechowywany w buforze odbiornika wraz z otrzymanymi bitami danych i bitami stopu. Flaga błędu parzystości (UPE) może wtedy zostać odczytana przez oprogramowanie w celu sprawdzenia czy wystąpił błąd parzystości.

UPE jest ustawiane jeżeli następny znak, który może zostać odczytany z bufora odbiornika ma błąd parzystości przy odbiorze i sprawdzanie parzystości zostało odblokowane (UMP1 = 1). Bit ten jest poprawny do czasu odczytu z bufora odbiornika.

- **Blokowanie odbiornika**

W przeciwieństwie do nadajnika, zablokowanie odbiornika następuje natychmiast. Dane przychodzące będą dlatego stracone. Kiedy jest on zablokowany (RXEN = 0) odbiornik nie będzie dłużej przeciążał wyjść portu RxD. Bufor FIFO odbiornika będzie opróżniony kiedy odbiornik zostanie zablokowany. Pozostające dane w buforze zostaną utracone.

- **opróżnianie bufora odbiornika**

Bufor odbiornika FIFO zostanie opróżniony kiedy odbiornik zostanie odłączony, np.: bufor zostanie opróżniony z zawartości. Nie przeczytane dane zostaną stracone. Jeżeli bufor musi zostać opróżniony podczas normalnej operacji, np..: po zaistnieniu warunków błędu, należy odczytywać lokacje UDR I/O dopóki flaga RXC jest wyzerowana. Następujące przykłady kodu pokazują jak należy opróżniać bufor:

Assembly Code Example[1]

```
USART_Flush:
  sbis  UCSRA, RXC
  ret
  in    r16, UDR
  rjmp  USART_Flush
```

C Code Example[1]

```
void USART_Flush( void )
{
  unsigned char dummy;
  while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

Uwaga: 1. Przykłady kodu zakładają, że specyficzne nagłówki zostały dołączone. Dla rejestrów I/O umiejscowionych w rozszerzonej mapie rozkaz I/O, IN, OUT, SBIS, SBIC, CBI i SBI muszą być zamienione przez rozkazy umożliwiające dostęp do rozszerzonej mapy pamięci. Zazwyczaj są to rozkazy LDS i STS połączone z SBRC, SBRS, SBR i CBR.

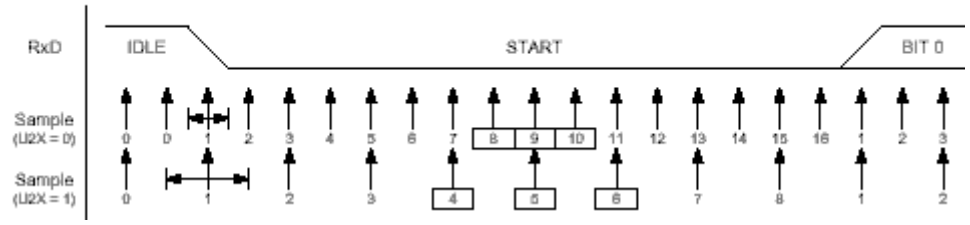
- **Asynchroniczne odbieranie danych**

USART zawiera odzyskiwanie zegara i moduł odzyskiwania danych dla obsługi asynchronicznego odbioru danych. Odzyskiwanie logiki zegara jest wykorzystywany dla synchronizacji zewnętrznie generowanej szybkości transmisji dla przychodzących asynchronicznie szeregowych ramek na wejścia RxD. Odzyskiwanie logiczne danych próbkuje i przepuszcza przez filtr dolnoprzepustowy każdy przychodzący bit, co ulepsza odporność na zakłócenia odbiornika. Zakres asynchronicznych operacji odbiornika zależy od dokładności wewnętrznego zegara szybkości transferu, szybkości przychodzenia nowych ramek, rozmiaru ramki w bitach.

- **Odzyskiwanie asynchronicznego zegara**

Logiczny zegar odzyskiwania synchronizuje wewnętrzny zegar do przychodzących szeregowo ramek. Rysunek 82 przedstawia proces próbkowania bitu startowego przychodzącej ramki. Szybkość próbkowania jest 16 krotna dla szybkości transferu dla normalnego trybu. I 8 krotna w trybie o podwójnej prędkości. Poziome strzałki ilustrują wahania synchronizacji wynikające z procesu próbkowania. Należy zauważyć, że podczas korzystania z trybu podwójnej prędkości wahania te są większe. Próbki wskazujące zero sa próbkowane kiedy linia RxD jest IDYL (nie ma aktywnej komunikacji).

Figure 82. Start Bit Sampling



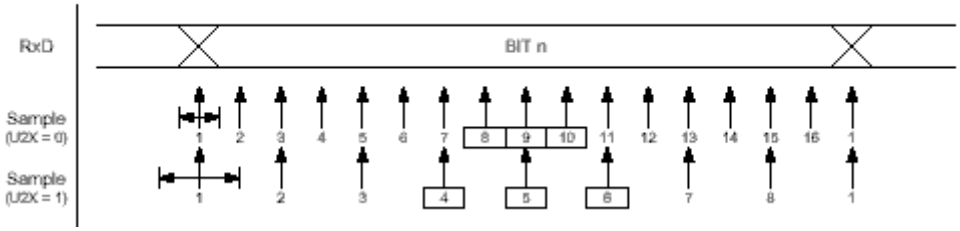
Kiedy zegar logicznego odzyskiwania wykryje przejście z wysokiego (IDYL) do niskiego (START) stanu na linii RxD jest inicjalizowana sekwencja wykrywająca bit startowy. Niech próbka 1 oznacza pierwszy zerowy przykład jak pokazuje to rysunek. Logika zegara odzyskiwania wykorzystuje wtedy próbki 8, 9 i 10 dla normalnego trybu i próbki 4, 5, 6 dla trybu o podwójnej prędkości, w celu sprawdzenia czy poprawny bit startu został otrzymany. Jeżeli 2 lub więcej z tych próbek ma stan wysoki (większość wygrywa), bit startowy jest odrzucany jako wynik zakłóceń i odbiornik zaczyna szukać następnego przejścia ze stanu wysokiego na niski. Jeżeli poprawny bit startu został wykryty, zegar logiczny odzyskiwania jest synchronizowany i odzyskiwanie danych może się rozpocząć. Proces synchronizacji jest powtarzany dla każdego bitu startu.

- **Asynchroniczny odzyskiwanie danych**

Kiedy zegar odbiornika jest zsynchronizowany do bitu startowego, odzyskiwanie danych może się rozpocząć. Moduł odzyskiwania danych wykorzystuje stan urządzenia, który może mieć 16

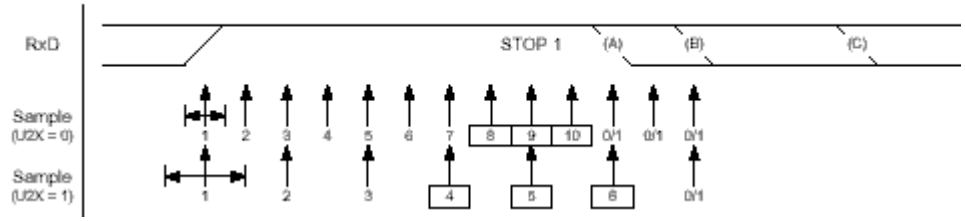
stanów na każdy bit w normalnym trybie pracy i 8 stanów dla trybu o podwójnej prędkości. Rysunek 83 pokazuje próbkowanie bitów danych i bitu parzystości. Każda z próbek ma nadawany numer równy temu ze statusu modułu odzyskiwania.

Figure 83. Sampling of Data and Parity Bit



Decyzja o logicznym poziomie odebranych bitów jest podejmowana poprzez ‘głosowanie’ wartości logicznych trzech próbek w środku odebranych bitów. Środkowe próbki są wyróżnione poprzez wpisanie ich wartości do ramek. Większość procesu ‘głosowania’ jest realizowana następująco: Jeżeli dwa lub wszystkie próbki mają wysoki poziom logiczny, otrzymany bit jest jedynką. Jeżeli dwa lub trzy próbki mają niski poziom , otrzymany bit jest zerem. Taki sposób odzyskiwania danych można traktować jako dolno przepustowy filtr na liniach przychodzących sygnałów RxD. Proces odzyskiwania jest później powtarzany dopóki kompletna ramka nie zostanie odebrana. Zawierając pierwszy bit stopu. Należy zauważyć, że odbiornik wykorzystuje tylko pierwszy bit stopu ramki. Rysunek 84 przedstawia próbkowanie bitu stopu i wcześniejszych możliwości wykrycia bitu startu dla wcześniej rozpoczynających się nowych ramek.

Figure 84. Stop Bit Sampling and Next Start Bit Sampling



Takie samo rozpoznawanie bitów jest robione w odniesieniu do bitu stopu jak i do wszystkich pozostałych bitów. Jeżeli jest zarejestrowane, że bit stopu ma mieć logiczną wartość zero, błąd ramki będzie ustawiony.

Nowe przejście ze stanu wysokiego do niskiego oznacza, że bit startu nowej ramki może przyjść po ostatnim bicie wykorzystywanym do większościowego ‘głosowania’. Dla normalnego trybu pracy, pierwsza próbka niskiego stanu może być nad punktem oznaczonym – A. Dla trybu o podwójnej prędkości pierwszy niski poziom musi zostać usunięty, aż do punktu – B. C zaznacza bit stopu pełnej długości. Wczesne wykrycie bitu startu wpływa na zakres operacyjny odbiornika.

- **Zakres operacji asynchronicznych**

Zakres operacji odbiornika zależy od niedopasowania między szybkością transmisji bitów i wewnętrznie generowanej szybkości transmisji. Jeżeli nadajnik wysyła ramki za szybko lub za

wolno lub wewnętrzny generator szybkości transmisji odbiornika nie ma podobnej bazowej frekwencji, odbiornik nie będzie w stanie zsynchronizować ramki z bitem startu.

Następujące równania służą obliczeniu stosunku szybkości przychodzących danych i wewnętrznej szybkości transmisji.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D **suma rozmiaru znaków i rozmiaru parzystości (D = 5 – 10 bit)**

S próbka na bit. S = 16 dla normalnej prędkości i S =8 dla podwojonej prędkości

S_F Pierwszy numer próbki używanej do większościowego ‘glosowania’. S_F = 8 dla normalnej prędkości i S_F =4 dla podwojonej prędkości

S_M Środkowy numer próbki używanej do większościowego ‘glosowania’. S_M = 9 dla normalnej prędkości i S_M =5 dla podwojonej prędkości

R_{slow} jest stosunkiem najwolniej przychodzących danych, które mogą zostać odebrane do szybkości transmisji odbiornika. R_{fast} jest stosunkiem najszybciej przychodzących danych, które mogą zostać odebrane do szybkości transmisji odbiornika

Tabele 75 i 76 opisują błędy maksymalnej szybkości transmisji odbiornika, które mogą być jeszcze tolerowane. Należy zauważyć, że tryb normalnej szybkości ma większą tolerancje na wahania szybkości transmisji.

Tabela 75 – Normalny tryb pracy

D	R_{slow} %	R_{fast} %	Maksymalny całkowity błąd %	Zalecana największy błąd odbioru %
5	93,20	106,67	+6,67/-6,8	±3,0
6	94,12	105,79	+5,79/-5,88	± 2,5
7	94,81	105,11	+5,11/-5,19	±2,0
8	95,36	104,58	+4,58/-4,54	±2,0
9	95,81	104,14	+4,14/-4,19	±1,5
10	96,17	103,78	+3,78/-3,83	±1,5

Tabele 76 – tryb podwojonej szybkości

D	R_{slow} %	R_{fast} %	Maksymalny całkowity błąd %	Zalecana największy błąd odbioru %
5	94,12	105,66	+5,66/-5,88	±2,5
6	94392	104,92	+4,92/-5,08	± 2,0
7	95,52	104,35	+4,35/-4,48	±1,5
8	96,00	103,90	+3,90/-4,00	±1,5
9	96,39	103,53	+3,53/-3,61	±1,5
10	96,70	103,23	+3,23/-3,30	±1,0

Zalecenia maksymalnego błędu szybkości transferu odbiornika zostały zrobione przy Założeniu, że nadajnik i odbiornik równo dzielą maksymalny błąd.

Są dwa możliwe źródła błędu szybkości transmisji dla odbiornika. Zegar systemowy odbiornika (XTAL) będzie zawsze miał jakąś niestabilność jeżeli zasilanie przekroczy górny poziom jak i temperatura. Korzystając z kwarcowego zegara do generowania systemowego zegara, sprawia to wtedy mniejszy problem, ale dla rezonatora system zegara może różnić się więcej niż 2% w zależności od tolerancji rezonatora. Drugie źródło można bardziej kontrolować. Generator szybkości transmisji nie może zawsze dokładnie dzielić częstotliwości systemowej w celu otrzymania szybkości transferu. W tym przypadku wartość UBRR dająca najniższy do zaakceptowania błąd może zostać wykorzystana.

- **Tryb komunikacji wieloprocesorowej**

Ustawiając bit tryb komunikacji wieloprocesorowej (MPCM) w UCSRA odblokowuje funkcje filtrujące przychodzących ramek odbieranych przez odbiornik USART. Ramki nie zawierające informacji o adresie będą ignorowane i nie przekazywane od bufora odbiornika. To efektywnie redukuje liczbę przychodzących bramek, które muszą zostać obsłużone przez CPU, w systemie z wieloma MCU, które komunikują się przez tą samą magistralę. Na nadajnik nie wywierają wpływu ustawienia MCU, ale musi być wykorzystywany w inny sposób kiedy jeżeli jest częścią systemu wykorzystującego tryb komunikacji wieloprocesorowej.

Jeżeli odbiornik jest przygotowany do odbioru ramek zawierających od 5 do 8 bitów danych , wtedy pierwszy bit stopu wskazuje czy są to dane czy adresy .Jeżeli odbiornik jest ustawiony na odbieranie ramek z 9 bitami danych, to dziewiąty bit (RXB8) jest wykorzystywany do identyfikacji ramek z adresami i danymi. Kiedy bit typu ramki jest jedynką, ramka zawiera adres. Jeżeli jest to zero w ramce znajdują się dane.

Tryb komunikacji wieloprocesorowej uaktywnia serwer podrzędny do MCU w celu odbioru danych od nadrzędnego MCU. Najpierw jest dekodowana ramka z adresem, aby wiedzieć, do którego MCU mają zostać przekazane dane. Jeżeli konkretny MCU został zaadresowany będzie on dostawał ramki z danymi, podczas gdy pozostałe MCU będą ignorowały nadchodzące ramki dopóki nie przyjdzie nowa ramka z adresem.

  - **Korzystanie z MPCM**

Aby MCU pracowało jako nadrzędne MCU, może ono używać formatu ramki z 9-bitami danych (UCSZ = 7). Dziewiąty bit (TXB8) musi być ustawiony w ramkach zawierających adres (TXB8 = 1) lub wyzerowany w ramkach z danymi (TXB8 = 0). MCU podrzędne muszą w tym przypadku być przystosowane do odbioru ramek z 9 bitami danych.

Następująca procedura powinna być przestrzegana w celu wymiany danych w wieloprocesorowym trybie komunikacji:

1. Wszystkie podrzędne MCU muszą pracować w trybie wieloprocesorowej komunikacji. (MPCM w UCSRA musi być ustawiony).

2. Nadrzędny MCU przesyła ramkę z adresem i wszystkie podrzędne MCU ją otrzymują i czytają. W podrzędnych MCU flaga RXC w UCSRA będzie ustawiona jak normalnie.
3. Każdy z podrzędnych MCU odczytuje rejestr UDR i decyduje czy został wybrany. Jeżeli tak, to zeruje bit MPCM, w przeciwnym wypadku oczekuje na następną ramkę z adresem.
4. Zaadresowane MCU otrzyma wszystkie ramki z dany mi aż do następnej ramki z adresem. Pozostałe podrzędne MCU, które nadal mają ustawiony bit MPCM, będą ignorowały ramki z danymi.
5. Kiedy ostatnia ramka z danymi została odebrana przez wybrane MCU, to wybrane MCU ustawia bit MPCM i oczekuje na nową ramkę z adresem od urządzenia nadrzędnego. Proces później przebiega od punktu 2.

Wykorzystywanie każdego z formatu ramek 5 –8 bitowych jest możliwe , ale niepraktyczne odkąd odbiornik musi zmieniać ustawienia między odbiorem formatu o n i n+1 znakach. To sprawia że operacje full-duplex są trudno wykonalne odkąd nadajnik i odbiornik używają tych samych ustawień znaków. Jeżeli są wykorzystywane ramki o znakach od 5 – 8 to nadajnik musi być ustawiony na wysyłanie dwóch bitów stopu (USBS = 1) odkąd pierwszy bit stopu wskazuje rodzaj ramki.

Nie należy używać instrukcji read-modify-write (SBI i CBI) do ustawiania lub zerowania bitu MPCM. Bit MPCM dzieli te same lokacje I/O co flaga TXC i może przez przypadek zostać wyzerowana przy wykorzystaniu rozkazów SBI i CBI.

- **Opis rejestrów USART**

  - **Rejestr danych I/O USATR – UDR**

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
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	

Rejestr buforowy nadajnika danych USART i rejestr buforowy odbiornika danych USART dzielą te same adresy I/O odnoszące się do rejestru danych USART lub UDR. Rejestr buforowy nadajnika danych (TXB) będzie rejestrem, do którego będą zapisywane dane przeznaczone do lokacji rejestrów UDR. Czytając z lokacji rejestrów UDR zostanie zwrócona zawartość rejestru bufora danych (RXB).

Dla 5, 6 lub 7-mio bitowych znaków danych starsze bity będą ignorowane przez nadajnik i ustawiane na zero w odbiorniku.

Bufor nadajnika może być tylko zapisany kiedy flaga UDRE w rejestrze UCSRA jest ustawiona. Dane zapisane do tego rejestru, kiedy ta flaga nie jest ustawiona będą ignorowane przez nadajnik USART. Kiedy dane są zapisane do bufora nadajnika i nadajnik jest odblokowany załaduje on te dane do rejestru przesuwnego kiedy ten będzie pusty. Następnie dane będą transmitowane poprzez wyjścia TxD.

Bufor odbiornika składa się z dwóch poziomów FIFO. FIFO zmieni swój stan zawsze kiedy została z niego pobrana dana. W związku z tym zachowaniem bufor nadajnika nie wykorzystuje instrukcji SBI i CBI na tych lokacjach. Należy być ostrożnym korzystając z rozkazów testujących bit SBIC i SBIS, ponieważ te rozkazy również zmieniają stan FIFO.

- **Rejestr statusu i kontroli A USART – UCSRA**

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

a: bit 7 – RXC: zakończona transmisja USART

Ten bit jest ustawiony kiedy w buforze odbiornika są nie przeczytane dane i jest wyzerowany kiedy bufor jest pusty. Jeżeli odbiornik jest zablokowany, bufor odbiornika zostanie opróżniony i w konsekwencji bit będzie wyzerowany. Flaga RXC może być wykorzystywana do generacji przerwania ukończenia odbioru (przejrzyj opis bitu RXCIE)

b: bit 6 – TXC: zakończone nadawanie USART

Ten bit jest ustawiony kiedy cała ramka został wysłana z rejestru przesuwnego w nadajniku i nie ma nowych danych oczekujących w buforze (UDR). Flaga ta jest automatycznie zerowana przy generacji przerwania zakończenia transmisji lub może zostać ręczenie ustawiona na 1. Flaga ta może generować przerwanie ukończenia transmisji.

c: bit 5 – UDRE: pusty rejestr danych

Flaga UDRE wskazuje czy bufor nadajnika jest gotowy, aby otrzymać nowe dane. Jeżeli UDRE jest jedynką, bufor jest pusty i gotowy na przyjęcie nowych danych. Flaga ta może generować przerwanie pustego rejestru danych(przejrzyj opis bitu UDRIE).

d: bit 4 - FE: błąd ramki

Ten bit jest ustawiony jeśli następny znak w buforze odbiornika miał błąd ramki przy odbiorze. Bit ten jest poprawny dopóki bufor odbiornika nie zostanie odczytany. Bit FE jest zerem jeżeli bit stopu odebranej danej jest jedynką. Należy zawsze ustawiać ten bit na zero przy zapisie do UCSRA.

e: bit 3 – DOR: przekroczenie danych

Bit ten jest ustawiony kiedy warunek przekroczenia danych został wykryty. Zachodzi to kiedy bufor odbiornika jest pełny (dwa znaki), do rejestru przesuwnego zostaje wpisywany nowy znak i zostaje wykryty następny bit startu. Bit ten jest poprawny dopóki bufor odbiornika (UDR) jest odczytywany. Należy zawsze ustawić ten bit na zero przy zapisywaniu UCSRA.

f: bit 2 – UPE – błąd parzystości

Bit ten jest ustawiony, kiedy następny znak w buforze odbiornika miał błąd parzystości wykryty podczas odbioru. Bit ten jest poprawny dopóki bufor odbiornika jest odczytywany. Należy zawsze ustawić ten bit na zero przy zapisywaniu UCSRA.

g: bit 1 – U2X: podwójna prędkość transmisji danych

Bit ten wpływa tylko na asynchroniczne operacje. Powinien być zapisany zerem dla synchronicznych operacji.

Ustawienie tego bitu na jeden zmniejszy dzielnik podzielnika częstotliwości transmisji z 16 na 8 efektywnie przyspieszając szybkość transmisji dla asynchronicznej komunikacji.

h: bit 0 – MPCM: tryb komunikacji wieloprocesorowej

Bit ten uaktywnia tryb komunikacji wieloprocesorowej. Kiedy ma wartość 1 wszystkie odbierane ramki w odbiorniku, które nie zawierają adresów będą ignorowane. Bit ten nie ma wpływu na nadajnik. Więcej informacji w opisie trybu komunikacji wieloprocesorowej na stronie 182.

- **Rejestr kontroli i statusu B USART – UCSRB**

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

a: bit 7 – RXCIE: odblokowanie przerwania zakończenia RX

Zapisując jeden do tego bitu odblokowujemy flagę przerwań RXC. Przerwanie to będzie generowane tylko jeśli bit RXCIE jest ustawiony na jeden, globalna flaga przerwań SREG jest ustawiona na jeden i bit RXC w UCSRA jest ustawiony.

b: bit 6 - TXCIE: odblokowanie przerwania zakończenia TX

Zapisanie tego bitu jedynką odblokowuje przerwania generowane przez flagę TXC. Przerwanie to będzie generowane tylko jeśli bit TXCIE jest ustawiony na jeden, globalna flaga przerwań SREG jest ustawiona na jeden i bit TXC w UCSRA jest ustawiony.

c: bit 5 – UDRIEL odblokowanie przerwania zgłaszającego pusty rejestr danych

Zapisując ten bit jedynką odblokowujemy flagę przerwania UDRE. Przerwanie to będzie generowane tylko jeśli bit UDRIE jest ustawiony na jeden, globalna flaga przerwań SREG jest ustawiona na jeden i bit UDRE w UCSRA jest ustawiony.

d: bit 4 – RXENL odblokowanie odbiornika

Zapisując ten bit jedynką odblokowujemy odbiornik USART. Odbiornik przeciąży normalne wyjścia portów dla RxD. Blokując odbiornik bufor odbiornika zostanie opróżniony unieważniając flagi FE, DOR, UPE.

e: bit 3 – TXEN: odblokowanie nadajnika

Zapisując ten bit jedynką odblokowujemy nadajnik USART. Nadajnik przeciąży normalne wyjścia portów dla TxD. Blokowanie nadajnika nie będzie efektywne, aż do czasu zakończenia trwającej transmisji. Po zablokowaniu porty TxD nie będą dłużej przeciążone.

f: bit 2 – UCSZ2: Rozmiar znaku

Bit ten w połączeni z bitami UCSZ!—0 w rejestrze UCSRC ustala liczbę bitów danych w ramce odbiornika i odbiornika.

g: bit 1 - RXB8: 8 bitów danych odbiornika

Jest to dziewiąty bit danych odebranego znaku kiedy operujemy na szeregowych ramkach z 9-bitami danych. Musi zostać przeczytany przed odczytem najmłodszego bitu z UDR.

h: bit 0 – TXB8: 8 bit danych nadajnika

Jest to dziewiąty bit danych nadawanego znaku kiedy operujemy na szeregowych ramkach z 9-bitami danych. Musi zostać zapisany przed zapisem najmłodszego bitu z UDR.

- **Rejestr kontroli i statusu C USART – UCSRC**

Bit	7	6	5	4	3	2	1	0	
	–	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
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	1	1	0	

Uwaga: rejestr ten nie jest dostępny w trybie współpracy z Atmego103

a: bit 7 – bit zarezerwowany

Bit ten jest zarezerwowany do użytku w przyszłości. W celu zapewnienia kompatybilności z przyszłymi urządzeniami ten bit musi mieć wartość zero.

b: bit 6 – UMSEL: wybór trybu USART

Bite ten wybiera pomiędzy trybem synchronicznym i asynchronicznym operacji..0 – operacje asynchroniczne, 1 – synchroniczne.

c: bity 5, 4 – UPM1..0: bity parzystości

Bity te odblokowują i ustawiają tryb generacji i sprawdzania parzystości. Jeśli nadajnik jest odblokowany automatycznie generuje i wysyła bity parzystości z każda ramką. Odbiornik będzie

wtedy generował bit parzystości dla odebranych bitów danych i porównywał go z ustawieniami UPM0. Jeżeli zostanie wykryty błąd flaga UPE w UCSRA zostanie ustawiony.

UPM1	UPM0	Tryb parzystości
0	0	Zablokowany
0	1	Zarezerwowany
1	0	Parzystość
1	1	Nieparzystość – bitowa negacja parzystości

d: bit 3 – USBS: bit wyboru stopu

Bit ten wybiera liczbę bitów stopu, która ma zostać dodana przez nadajnik. Odbiornik ignoruje te ustawienia. 0 – jeden bit stopu , 1 –2bity stopu.

e: bity 2..1 – UCSZ!..0: rozmiar znaku

Bity te w połączeniu z UCSZ2 w UCSRB ustawiają liczbę bitów danych (rozmiar znaku) w ramce odbiornika i nadajnika.

USCZ2	USCZ1	USCZ0	Rozmiar znaku
0	0	0	5 bitów
0	0	1	6 bitw
0	1	0	7 bitów
0	1	1	8 bitów
1	0	0	Zarezerwowane
1	0	1	Zarezerwowane
1	1	0	Zarezerwowane
1	1	1	9 bitów

f: bit 0 – UCPOL: polaryzacja zegara

Ten tryb jest wykorzystywany tylko w trybie synchronicznym. W trybie asynchronicznym powinien zostać zapisany zerem. Ustawia on zależność pomiędzy zmianą wyjścia danych i próbką wejściowa danych oraz synchronicznym zegarem.

UCPOL	Zmiana danych w nadajniku Wyjście TxD	Zmiana próbek w odbiorniku Wejście RxD
0	Opadające zbocze XCK	Narastające zbocze XCK
1	Narastające zbocze XCK	Opadające zbocze XCK

- **Rejestry szybkości transmisji USART – UBRRL i UBRRH**

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

UBRRH nie jest dostępne w trybie kompatybilności z ATmega103

a: bity 15..12 – bity zarezerwowane

Bity te są zarezerwowane do przyszłego wykorzystania. W celu zapewnienia kompatybilności z przyszłymi urządzeniami bity te powinny być ustawione zero przy zapisie UBRRH.

b: bity 11..0 – UBRR11..0: rejestr szybkości transmisji USART

Jest to 12 bitowy rejestr, który zawiera dane o szybkości transferu USART. UBRRH zawiera cztery najstarsze bity a UBRRL 8 młodszych bitów. Trwająca transmisja zostanie przerwana w przypadku zmian w tym rejestrze. Zapisanie UBRRL spowoduje natychmiastową aktualizację szybkości transferu prescalera.

- **Przykłady ustawień szybkości transmisji**

Dla standardowego kwarcu i rezonatora częstotliwości tabela 82 przedstawia szybkość transferu w trybie asynchronicznym. Wartość UBRR, która dostarcza aktualnego podzielnika szybkości transferu nie różni się bardziej niż o 0.5% od zamierzonej szybkości – wytłuszczone w tabeli. Większe oszacowanie błędu jest możliwe ale wtedy odbiornik będzie miał mniejszą odporność na zakłócenia kiedy to oszacowanie jest wysokie, szczególnie dla dużych szeregowych ramek („Zakres operacji asynchronicznych” str 181). Wartości błędów można obliczyć korzystając z następujących równań:

$$blad[\%] = \left[\frac{szybkosc\_transferu_{najblizsz\_wartosc}}{szybkosc\_transferu} - 1 \right] * 100\%$$

Table 82. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 1.0000 MHz				f_{osc} = 1.8432 MHz				f_{osc} = 2.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max (1)	62.5 kbps		125 kbps		115.2 kbps		230.4 Mbps		125 kpbs		250 kbps	

1. UBRR = 0, błąd = 0.0%

Table 83. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 3.6864 MHz				f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max (1)	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kpbs		921.6 kbps	

1. UBRR = 0, Error = 0.0%

Table 84. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRR = 0, Error = 0.0%

Table 85. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 16.0000 MHz				f_{osc} = 18.4320 MHz				f_{osc} = 20.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max (1)	1 Mbps		2 Mbps		1.152 Mbps		2.304 Mbps		1.25 Mbps		2.5 Mbps	

1. UBRR = 0, Error = 0.0%

SZEREGOWY INTERFEJS DWUPRZEDWODOWY

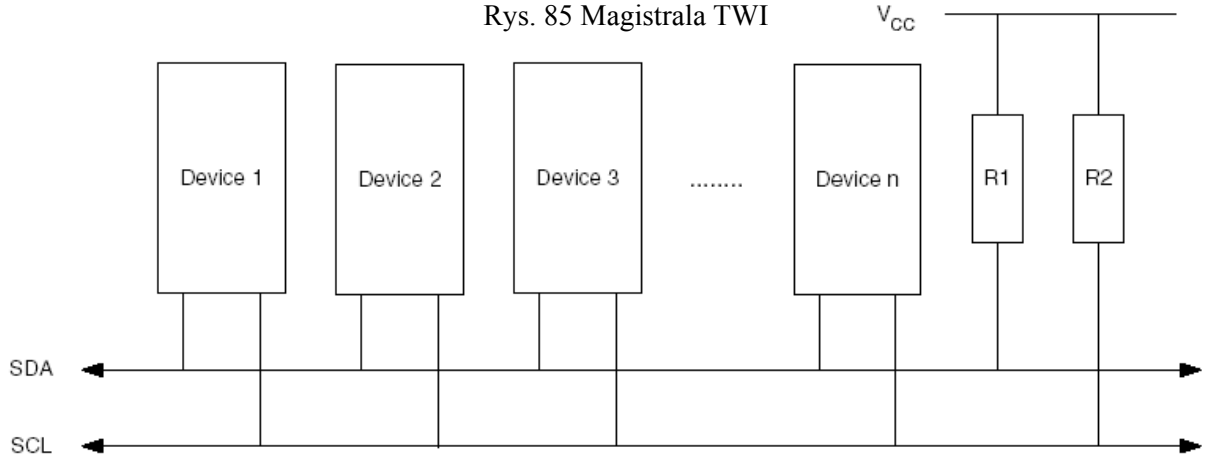
WŁAŚCIWOŚCI

- Wspieranie operacji Master i Slave
- Działanie urządzenia zarówno w trybie nadajnika jak i odbiornika
- 7-bitowa przestrzeń adresowa umożliwiająca zaadresowanie 128 różnych urządzeń w trybie Slave
- Prędkość transferu danych do 400kHz
- W pełni programowalne adresy urządzeń Slave z wsparciem wywołań ogólnych

SZEREGOWY INTERFEJS DWUPRZEWODOWY, DEFINICJA MAGISTRALI

Szeregowy interfejs dwuprzewodowy (TWI) idealnie pasuje do aplikacji mikrosterowników. Protokół TWI pozwala projektantowi systemu na połączenie do 128 różnych urządzeń używając tylko dwóch dwukierunkowych magistral, jednej dla zegara (SCL) i drugiej dla danych (SDA). Jedynym zewnętrznym sprzętem potrzebnym do implementacji magistrali jest pojedynczy rezystor podciągający dla każdej z magistral TWI. Wszystkie urządzenia podpięte do magistrali mają odrębne adresy. Mechanizmy rozwiązujące problem konfliktów na magistrali są zawarte w protokole TWI.

Rys. 85 Magistrala TWI



TERMINOLOGIA TWI
Następujące definicje są często spotykane w tej części dokumentacji:

MASTER – urządzenie które inicjuje oraz przerywa transmisję, master generuje również impulsy zegara SCL.

SLAVE – urządzenie zaadresowane przez mastera

TRANSMITTER – urządzenie umieszczające dane na magistrali

RECEIVER – urządzenie czytające dane z magistrali

POŁĄCZENIE ELEKTRYCZNE

Jak przedstawiono na rys. 1, obydwie magistrale są podłączone do dodatniego bieguna zasilania poprzez rezystory podciągające. Sterowniki magistral wszystkich urządzeń są typu otwarty dren lub otwarty kolektor. Umożliwia to realizację funkcji AND na drucie co jest niezbędne dla operacji wykonywanych przez interfejs. Stan niski na magistrali TWI pojawia się jeżeli jedno lub więcej urządzeń TWI ma na wyjściu zero. Stan wysoki pojawia się gdy wyjścia wszystkich urządzeń TWI są w stanie wysokiej impedancji, pozwalając w ten sposób rezystorom podciągającym podwyższyć napięcie. Należy zauważyć że wszystkie urządzenia AVR podłączone do magistrali TWI muszą być zasilane w kolejności pozwalającej na jakiekolwiek operacje na magistrali.

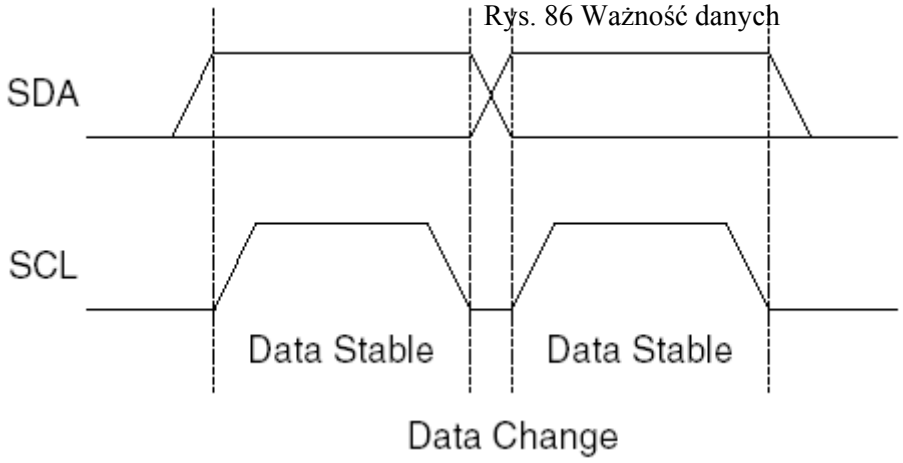
Liczba urządzeń, które mogą być podłączone do magistrali jest ograniczona przez limit pojemnościowy wynoszący 400 pF oraz przez 7-bitową przestrzeń adresową przeznaczoną dla urządzeń slave.

TRANSFER DANYCH I FORMAT RAMKI

PRZESYŁANIE BITÓW

Każdemu bitowi danych przesyłanemu przez magistralę TWI towarzyszy impuls na linii zegara. Sygnał danych musi być stabilny w momencie kiedy na linii zegara jest stan wysoki. Jedynym wyjątkiem od tej reguły jest generowanie warunków startu i stopu.

Rys. 86 Ważność danych

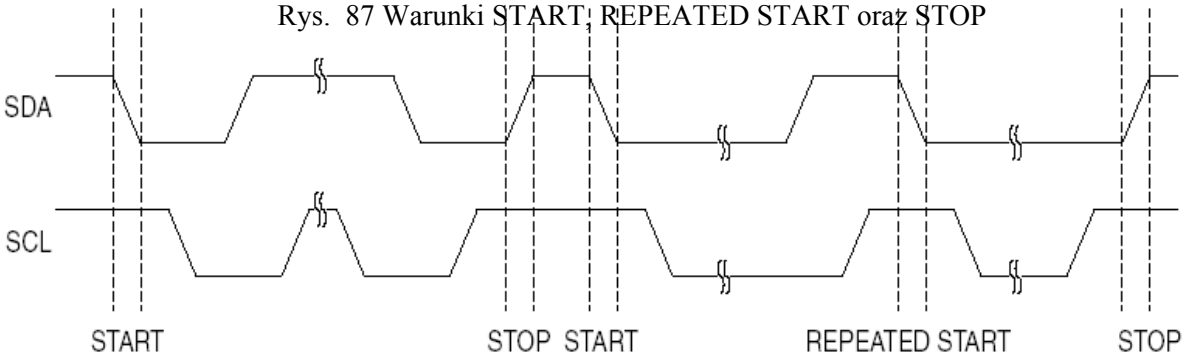


WARUNKI START I STOP

Master inicjuje oraz przerywa transmisję. Transmisja jest inicjowana kiedy master wyda warunek START na magistralę, przerywana gdy pojawi się warunek STOP. Pomiędzy warunkami START i STOP magistrala jest uważana za zajętą i żaden inny master nie powinien próbować przejęcia nad nią kontroli. Z przypadkiem specjalnym mamy do czynienia gdy nowy warunek START pojawi się pomiędzy warunkami

START i STOP. Tego typu warunek nosi nazwę REPEATED START i jest wykorzystywany gdy master życzy sobie zainicjować nowy transfer bez porzucania sterowania nad magistralą. Po REPEATED START magistralę uważa się za zajętą dopóki nie pojawi się następny STOP. Takie zachowanie jest identyczne z tym które miało miejsce w przypadku START dlatego przyjęło się że START opisuje zarówno START jak i REPEATED START. Jak przedstawiono na poniższym rysunku warunki START i STOP są sygnalizowane przez zmianę poziomu na linii SDA podczas gdy na linii SCL jest stan wysoki.

Rys. 87 Warunki START, REPEATED START oraz STOP



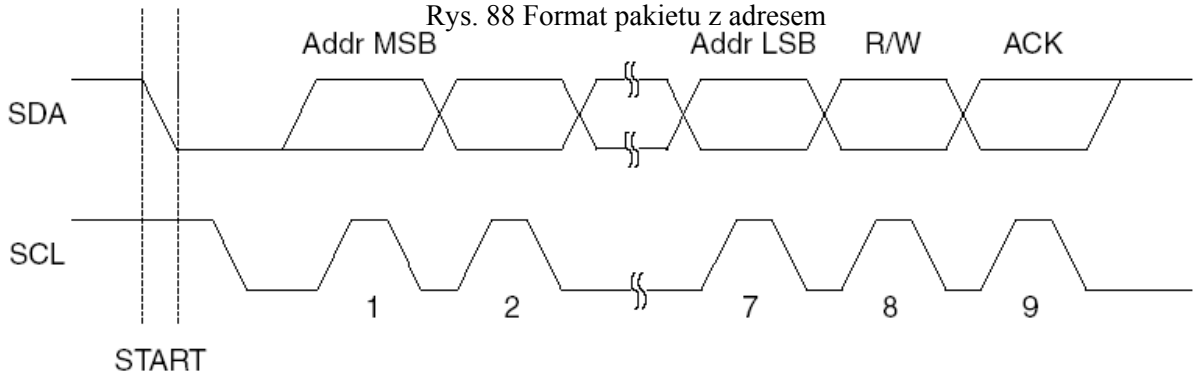
FORMAT PAKIETU ZAWIERAJĄCEGO DANE

Wszystkie pakiety adresów przesyłane na magistrali TWI mają długość 9 bitów i zawierają 7 bitów adresowych, bit sterujący odczyt/zapis oraz bit potwierdzenia. Jeżeli bit odczytu/zapisu jest ustawiony, rozpoczyna się operacja odczytu, w przeciwnym wypadku powinna się rozpocząć operacja zapisu. Kiedy urządzenia slave rozpoznało swój adres na magistrali powinno to potwierdzić przez ustawienie stanu niskiego na linii SDA w dziewiątym cyklu SCL(ACK). Jeżeli urządzenie którego adres pojawił się na magistrali jest zajęte lub z innych przyczyn nie może świadczyć usług na żądanie urządzenia master, linia SDA powinna być pozostawiona w stanie wysokim w takcie ACK. Master może wtedy wysłać warunek STOP lub REPEATED START aby zainicjować nową transmisję. Cześć pakietu zawierająca adres urządzenia oraz bit READ lub WRITE jest nazywana odpowiednio SLA+R lub SLA+W. MSB bajtu adresu jest przesyłane najpierw. Adres urządzenia slave może być przydzielony dowolnie przez projektanta, ale adres 0000 000 jest zarezerwowany dla wywołań ogólnych.

Jeżeli wydane zostało wywołanie ogólne wszystkie urządzenia slave powinny odpowiedzieć przez ustawienie stanu niskiego na linii SDA w cyklu ACK. Wywołanie ogólne jest używane gdy master życzy sobie przesłać tę samą wiadomość to kilku urządzeń slave w systemie. Kiedy adres wywołania ogólnego za którym znajdzie się bit zapisu zostanie przesłany na magistralę, wszystkie urządzenia slave ustawione na potwierdzenie wywołania ogólnego wymuszą stan niski na linii SDA w cyklu ACK. Wysłane następnie dane otrzymają wszystkie urządzenia slave, które potwierdziły wywołanie ogólne. Należy zauważyć że transmisja adresu wywołania ogólnego za którą podąża bit odczytu jest bezsensowna, tak jak w przypadku transmisji różnych danych przez różne urządzenia slave jednocześnie.

Wszystkie adresy w formacie 1111 xxx powinny być zarezerwowane dal przyszłych zastosowań.

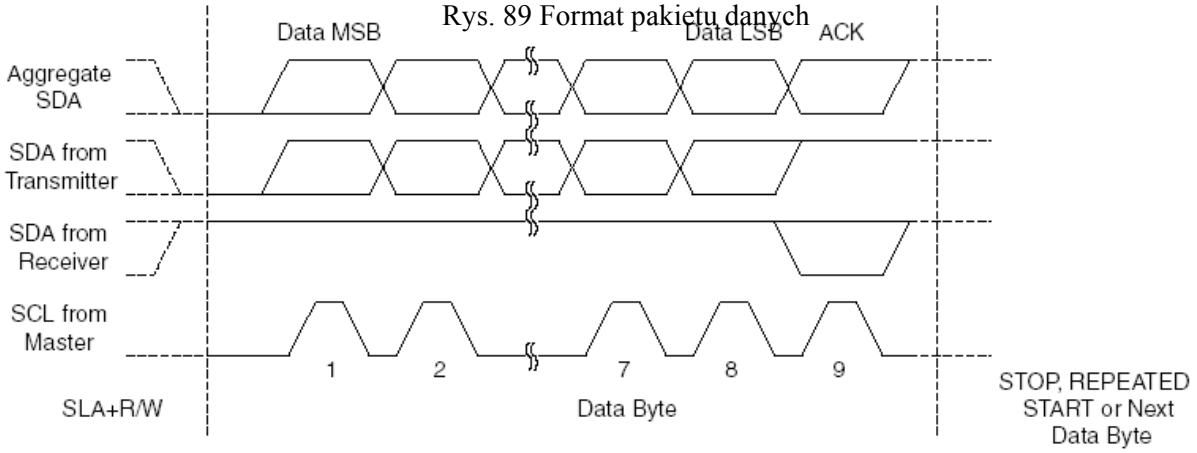
Rys. 88 Format pakietu z adresem



FORMAT PAKIETU DANYCH

Wszystkie pakiety danych przesłane na magistralę TWI mają długość 9 bitów, wśród których znajduje się bajt danych oraz bit potwierdzenia. Podczas transferu danych master generuje impulsy zegara oraz warunki START i STOP, podczas gdy odbiornik jest odpowiedzialny za potwierdzenie przyjęcia. Potwierdzenie (ACK) jest sygnalizowane przez odbiornik poprzez wymuszenie stanu niskiego na linii SDA podczas dziewiątego cyklu SCL. Jeżeli odbiornik pozostawi linię SDA w stanie wysokim sygnalizowany jest NACK. Kiedy odbiornik odebrał ostatni bajt czy też z innych przyczyn nie może odebrać już więcej bajtów powinien poinformować nadajnik wysyłając NACK po końcowym bajcie. Początkowo odbierany jest MSB bajtu danych.

Rys. 89 Format pakietu danych

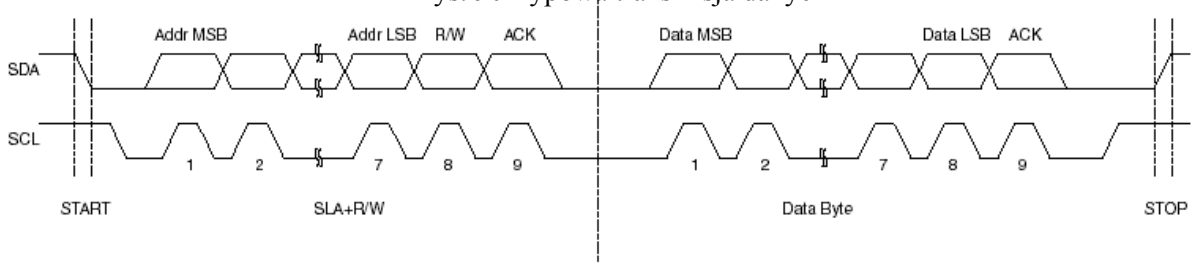


TRANSMISJA PAKIETÓW ŁĄCZONYCH

Transmisja zwykle zawiera warunek START, SLA+R/W, jeden lub więcej pakietów danych oraz warunek STOP. Puste wiadomości składające się z warunku START po którym następuje warunek STOP są niedozwolone. Należy zauważyć że funkcja AND na drucie na linii SCL może być wykorzystana do implementacji uzgodnienia pomiędzy urządzeniem master i slave. Urządzenie slave może rozszerzyć okres w którym na linii SCL jest stan niski poprzez jego wymuszenie. Jest to użyteczne w przypadku gdy szybkość zegara ustawiona przez mastera jest zbyt duża dla urządzenia slave, lub gdy slave potrzebuje

dodatkowego czasu na obliczenia pomiędzy transmisjami danych. Slave rozszerzający okres stanu niskiego na linii SCL nie będzie wpływał na okres stanu wysokiego który zależy od mastera. W konsekwencji slave może redukować prędkość transferu danych TWI przez wydłużanie cyklu roboczego. Rys. 6 pokazuje typowa transmisję danych. Należy zauważyć, że kilka bajtów danych może być przesyłanych pomiędzy SLA+R/W a warunkiem STOP, w zależności od protokołu programowego zaimplementowanego w aplikacji.

Rys. 90 Typowa transmisja danych



SYSTEMY MAGISTRAL MULTI-MASTER, ARBITRAŻ, SYNCHRONIZACJA

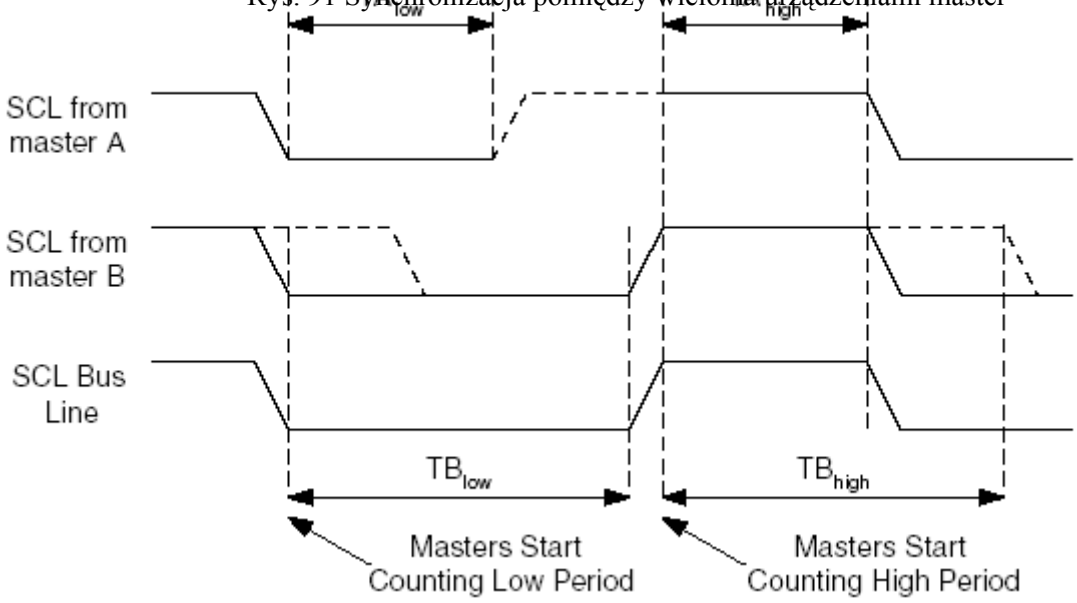
Protokół TWI pozwala na tworzenie systemów magistral z kilkoma urządzeniami typu master. W tym celu podjęto specjalne środki żeby zapewnić normalną kontynuację transmisji nawet wtedy gdy dwa lub więcej urządzeń master zainicjują transmisję w tym samym czasie. W systemach multi-master pojawiły się dwa problemy:

- Należy zaimplementować algorytm pozwalający tylko jednemu urządzeniu master zakończyć transmisję. Inne urządzenia tego typu powinny zaprzestać transmisji kiedy odkryją że przegapiły proces wyboru zwany arbitrażem. Kiedy współzawodniczące urządzenie master odkryje że przegapiło proces arbitrażu powinno natychmiast przełączyć się w tryb slave aby sprawdzić czy zwycięski master nie wysłał jego adresu na magistralę. Fakt że wiele urządzeń master zaczyna transmisję w tym samym czasie nie powinien być wykrywalny przez urządzenia slave, tzn. dane przesłane na magistralę nie mogą być fałszywe.
- Różne urządzenia master mogą używać różnych częstotliwości SCL. Trzeba wymyślić schemat synchronizacji szeregowych zegarów wszystkich urządzeń typu master, aby pozwolić na kontynuację transmisji. Ułatwi to proces arbitrażu.

AND na drucie na magistralach jest wykorzystywany w celu rozwiązania obu tych problemów. Szeregowe zegary wszystkich urządzeń master zostaną wymnożone na drucie tworząc zegar złożony z okresem występowania stanu wysokiego równym okresowi występowania stanu wysokiego urządzenia master z najkrótszym okresem stanu wysokiego. Okres stanu niskiego złożonego zegara jest równy okresowi niskiemu urządzenia master z najdłuższym okresem niskim. Należy zauważyć że wszystkie

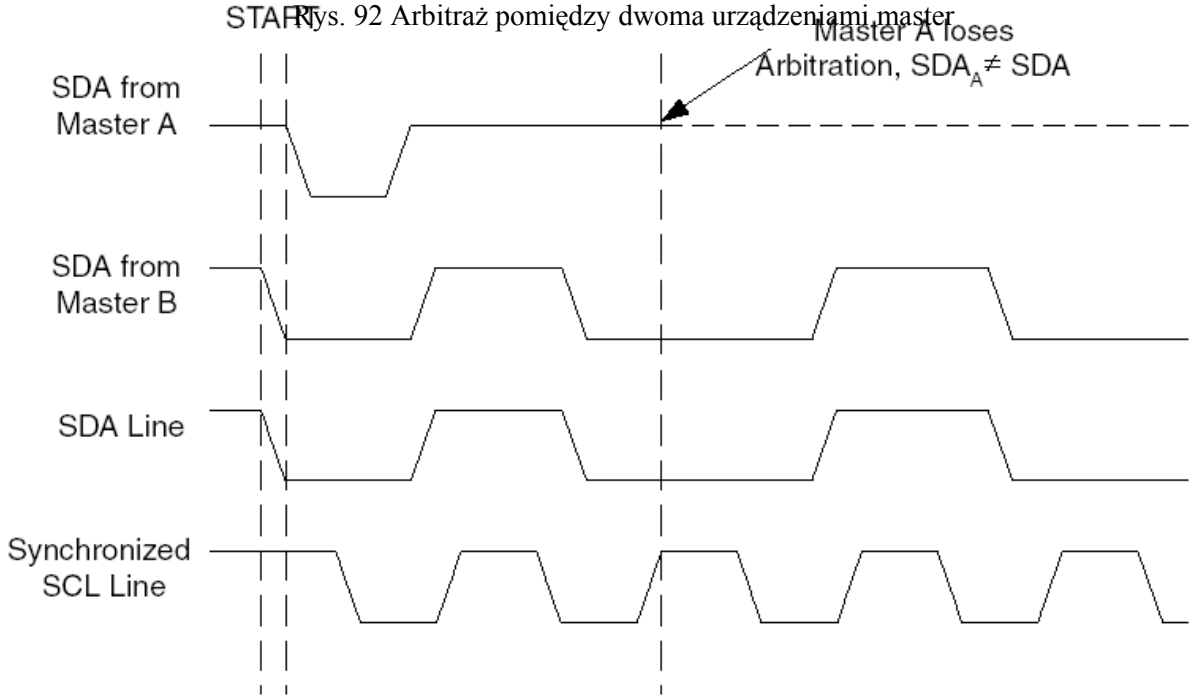
urządzenia master nasłuchujące na linii SCL, zaczynają efektywnie zliczać swoje wysokie i niskie okresy kiedy złożona linia SCL przechodzi odpowiednio w stan wysoki lub niski.

Rys. 91 Synchronizacja pomiędzy wieloma urządzeniami master



Arbitraż jest wykonywany stale przez wszystkie urządzenia master monitorujące linię SDA po wyprowadzeniu danych. Jeżeli wartość odczytana z linii SDA nie pasuje do wartości wyprowadzonej przez mastera, stracił on arbitraż. Urządzenie master może stracić arbitraż jedynie w przypadku gdy wyprowadzi wysoką wartość SDA podczas gdy inne urządzenie master wyprowadzi wartość niską. Pokonany master powinien natychmiast przejść w tryb slave, sprawdzając czy nie został zaadresowany przez zwycięskie urządzenie master. Linia SDA powinna pozostać w stanie wysokim, ale przegranym urządzeniom master pozwala się generować sygnał zegarowy do końca bieżących danych lub adresu. Arbitraż będzie kontynuowany tak długo dopóki nie pozostanie tylko jedno urządzenie master, co może zabrać wiele bitów. Jeżeli kilka urządzeń master chce zaadresować to samo urządzenie slave arbitraż jest kontynuowany w czasie przesyłania pakietu danych.

Rys. 92 Arbitraż pomiędzy dwoma urządzeniami master



Należy zauważyć że arbitraż nie jest możliwy pomiędzy:

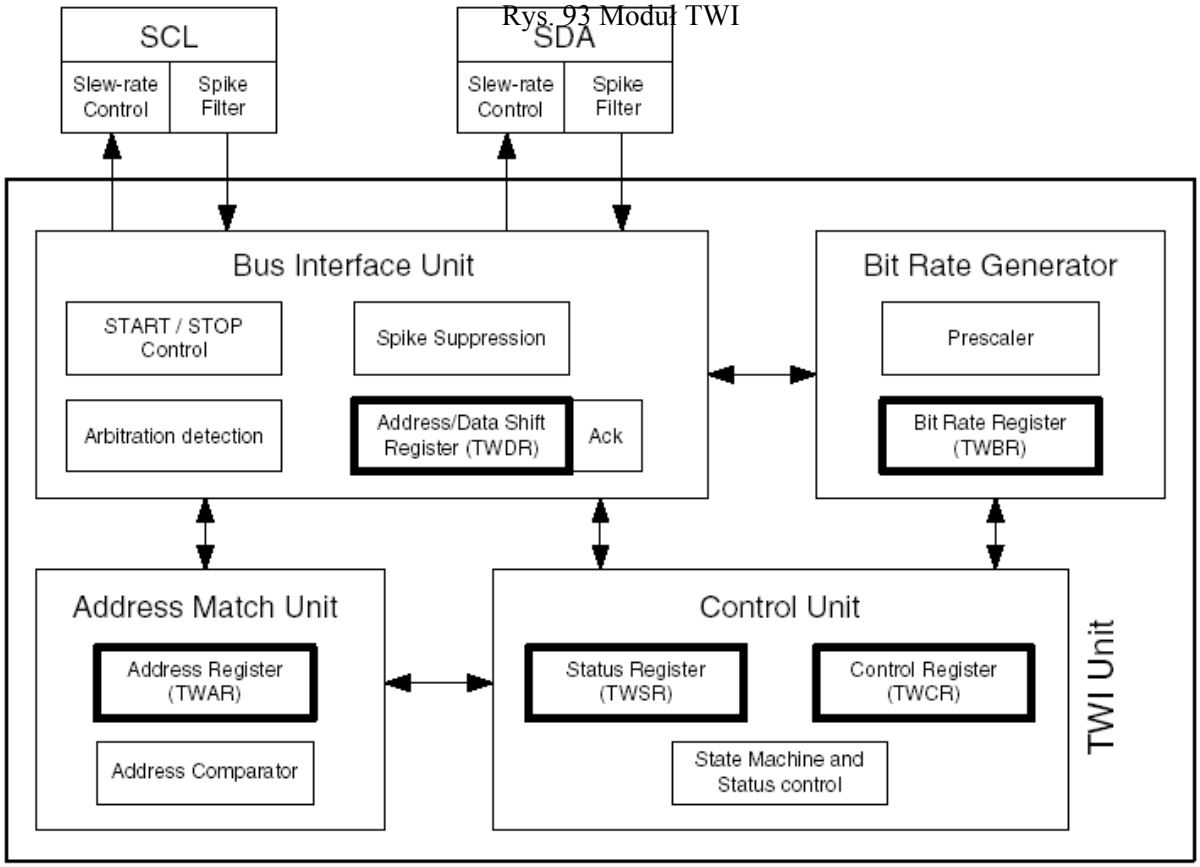
- Warunkiem REPEATED START a bitem danych
- Warunkiem STOP a bitem danych
- Warunkami REPEATED START i STOP

Odpowiedzialność, aby te zabronione sytuacje nigdy nie miały miejsca spoczywa na oprogramowaniu użytkownika. Oznacza to że w systemach multi-master wszystkie transfery danych muszą wykorzystywać te samą kompozycję SLA+R/W i pakietu danych. Innymi słowy: Wszystkie transmisje danych muszą zawierać tę samą ilość pakietów danych, w przeciwnym wypadku nie da się przewidzieć rezultatów arbitrażu.

PRZEGLĄD MODUŁU TWI

Moduł TWI zawiera kilka podjednostek jak pokazano na rys. 93. Wszystkie rejestry narysowane pogrubioną linią są dostępne przez magistralę danych AVR.

Rys. 93 Moduł TWI



PINY SCL I SDA

Te piny pełnią rolę interfejsu pomiędzy AVR TWI a resztą systemu jednostki centralnej mikrosterownika (MCU). Sterowniki wyjściowe zawierają ogranicznik slew-rate, aby pozostać w zgodzie ze specyfikacją TWI. Części wejściowe zawierają jednostkę tłumiącą impulsy o dużej amplitudzie i krótkim czasie trwania (krótszym niż 50 ns). Należy zauważyć że wewnętrzne zatrzymywanie się w blokach AVR może zostać aktywowane przez ustawienie bitów PORT odpowiadających pinom SCL i SDA. Wewnętrzne zatrzymania mogą w niektórych systemach eliminować potrzebę użycia zatrzymań zewnętrznych.

MODUŁ GENERATORA BIT RATE

Ta jednostka steruje okresem SCL kiedy działamy w trybie master. Okres SCL jest uzależniony od ustawień rejestru TWBR oraz bitów prescalera w rejestrze stanu TWI (TWSR). Operacje wykonywane w trybie slave nie zależą od ustawień TWBR oraz TWSR, ale częstotliwość zegara jednostki centralnej w urządzeniu slave musi być przynajmniej 16 razy większa od częstotliwości SCL. Należy zauważyć że urządzenia slave mogą przedłużać okres niski na linii SCL redukując przez to średni okres zegara magistrali TWI. Częstotliwość SCL jest generowana zgodnie z następującym równaniem:

$$f\ SCL = \frac{f\ zegara\ CPU}{16 + 2(TWBR) * 4^{TWPS}}$$

- TWBR = wartość rejestru TWBR
- TWPS = wartość bitów prescalera w rejestrze stanu TWSR

Uwaga: TWBR powinien być równy 10 lub więcej jeżeli działamy w trybie master. Jeżeli TWBR jest niższy niż 10, master może generować nieprawidłowe sygnały na wyjściu SDA i SCL dla przypomnienia bajtu. Problem może pojawić się gdy działamy w trybie master, wysyłamy START + SLA+R/W do urządzenia slave (slave nie musi być podłączony do magistrali aby ten warunek zaszedł).

MODUŁ INTERFEJSU MAGISTRALI

Ta jednostka zawiera rejestr przesuwny danych i adresu (TWDR), sterownik STAR/STOP oraz sprzętowe wykrywanie arbitrażu. TWDR zawiera adres lub dane które mają zostać wysłane lub odebrany adres lub dane. Oprócz 8-bitowego TWDR moduł interfejsu magistrali zawiera rejestr z bitem (N)ACK. Rejestr (N)ACK nie jest bezpośrednio dostępny dla oprogramowania. Jednakże podczas odbierania może zostać ustawiony lub skasowany przez manipulację rejestrem sterującym (TWCR). W trybie nadajnika wartość otrzymanego bitu (N)ACK może być określona przez wartość w TWSR.

Sterownik START/STOP jest odpowiedzialny za generowanie i wykrywanie warunków START, REPEATED START i STOP. Sterownik START/STOP jest w stanie wykryć warunki START i STOP nawet wtedy gdy AVR MCU jest w jednym z trybów uśpienia, poprzez aktywację trybu wzbudzenia MCU jeżeli jest adresowana przez urządzenie master.

Jeżeli TWI zainicjował transmisję jako master, sprzętowe wykrywanie arbitrażu stale monitoruje transmisję próbując decydować jeżeli trwa arbitraż. Jeżeli TWI przegrał w procesie arbitrażu jednostka sterująca jest o tym informowana. Można wtedy zareagować prawidłowo i stosowne kody stanu zostaną wygenerowane.

MODUŁ DOPASOWANIA ADRESU

Jednostka dopasowania adresu sprawdza czy otrzymany adres pasuje do 7-bitowego adresu w rejestrze adresowym (TWAR). Jeżeli bit rozpoznawania wywołań ogólnych (TWGCE) w TWAR jest ustawiony na jeden, wszystkie przychodzące bity adresów będą również porównywane z adresami wywołań ogólnych. Po dopasowaniu adresu moduł sterujący jest informowany, zezwalając na podjecie stosownego działania. TWI może potwierdzić swój adres lub nie, zależnie od ustawień w TWCR. Jednostka dopasowania adresu jest w stanie porównywać adresy nawet wtedy gdy AVR MCU jest w stanie uśpienia, poprzez aktywację trybu wzbudzenia MCU jeżeli jest adresowana przez urządzenie master. Jeżeli pojawi się kolejne przerwanie dopasowania adresu podczas wyłączania TWI i wzbudzania jednostki centralnej, TWI przerwie działanie i powróci do stanu bezczynności. Jeżeli powoduje to jakieś problemy zapewniamy że dopasowanie adresu TWI jest jedynym aktywnym przerwaniem podczas wyłączania.

JEDNOSTKA STERUJĄCA

Jednostka sterująca monitoruje magistralę TWI i generuje odpowiedzi odpowiadające ustawieniom w rejestrze sterującym (TWCR). Jeżeli pojawi się zdarzenie na magistrali TWI wymagające uwagi aplikacji, znacznik przerwania (TWINT) jest potwierdzany. W następnym cyklu zegarowym, TWSR jest aktualizowany z kodem stanu identyfikującym zdarzenie. TWSR zawiera jedynie istotne informacje o stanie kiedy TWINT jest potwierdzany. W każdym innym razie, TWSR zawiera specjalny kod stanu wskazujący że żadne istotne informacje nie są dostępne. Tak długo jak znacznik TWINT jest ustawiony, linia SCL jest utrzymywana w stanie niskim. Pozwala to oprogramowaniu zakończyć swoje zadania zanim zezwoli się na kontynuację transmisji TWI.

Znacznik TWINT jest ustawiany w następujących sytuacjach:

- Po przesłaniu przez TWI warunku START/REPEATED START
- Po przesłaniu przez TWI SLA+R/W
- Po przesłaniu przez TWI bajtu adresu
- Po utracie przez TWI arbitrażu
- Po zaadresowaniu TWI jego adresem slave lub wywołaniem ogólnym
- Po odebraniu przez TWI bajtu danych
- Po odebraniu STOP lub REPEATED START gdy urządzenie ciągle jest adresowane jako slave
- Kiedy pojawił się błąd magistrali spowodowany niedozwolonym warunkiem START/STOP

OPIS REJESTRÓW TWI

REJESTR TWI BIT RATE – TWBR

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
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	

- **Bity 7..0 – Rejestr TWI Bit Rate**

  TWBR służy do wyboru dzielnika generatora bit rate. Generator bit rate jest dzielnikiem częstotliwości , który generuje częstotliwość zegara SCL w trybie master.

REJESTR STERUJĄCY TWI – TWCR

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TWCR jest wykorzystywany do sterowania operacjami TWI. Używa się go do aktywacji TWI, inicjacji dostępu do urządzenia master przez wydanie warunku START na magistralę, generacji potwierdzenia odbiornika, generacji warunku STOP, sterowania zawieszaniem magistrali kiedy dane do zapisu są

zapisywane do TWDR. Rejestr TWCR wskazuje również kolizję zapisu, kiedy próbuje się zapisać dane do TWDR podczas gdy jest on niedostępny.

- **Bit 7 – TWINT: znacznik przerwań TWI**

Ten bit jest ustawiany sprzętowo kiedy TWI zakończy bieżące zadanie i oczekuje odpowiedzi ze strony oprogramowania. Jeżeli I-ty bit w SREG i TWIE w TWCR są ustawione, MCU skoczy pod wektor przerwań TWI. Kiedy TWINT jest ustawiony, niski okres SCL jest rozciągany. Flaga TWINT musi być kasowana sprzętowo poprzez wpisanie logicznej jedynki. Należy zauważyć że ta flaga nie jest automatycznie kasowana sprzętowo gdy wykonuje się procedura obsługi przerwania. Trzeba też zwrócić uwagę że kasowanie tego znacznika rozpoczyna działanie TWI, tak więc każdy dostęp do TWAR, TWSR i TWDR musi się zakończyć zanim ta flaga zostanie skasowana.

- **Bit 6 – TWEA: bit aktywacji potwierdzenia TWI**

Bit TWEA steruje generacją impulsu potwierdzenia. Jeżeli na ten bit wpiszemy jeden, impuls ACK jest generowany na magistrali TWI jeżeli są spełnione następujące warunki:

1. Otrzymany został własny adres urządzenia slave
2. Otrzymane zostało ogólne wywołanie, podczas gdy TWGCE w TWAR został ustawiony.
3. Otrzymany został bajt danych w trybie odbiornik master lub odbiornik slave

Przez wpisanie TWEA na zero urządzenie może być czasowo wirtualnie odłączone od szeregowej dwuprzewodowej magistrali. Rozpoznawanie adresu może być wznowione przez ponowne wpisanie bitu TWEA na jeden.

- **Bit 5 – TWSTA: bit TWI warunku START**

Aplikacja zapisuje jeden na bit TWSTA kiedy chce stać się urządzeniem master na dwuprzewodowej szeregowej magistrali. Układ TWI sprawdza czy magistrala jest dostępna i generuje warunek STRAT jeżeli magistrala jest wolna. Jednakże jeżeli magistrala jest zajęta, TWI czeka aż warunek STOP zostanie wykryty, i wówczas generuje nowy warunek START, przyznając magistrali status Master. TWSTA jest kasowany przez układ TWI kiedy warunek START został przesłany.

- **Bit 4 – TWSTO: bit TWI warunku STOP**

Wpisanie jedynki na bit TWSTO w trybie master wygeneruje warunek STOP na magistrali. Kiedy warunek STOP jest wykonywany, bit TWSTO jest kasowany automatycznie. W trybie slave ustawienie bitu TWSTO może zostać użyte do powrotu z warunku błędu. Wtedy nie zostanie wygenerowany warunek STOP, ale TWI powróci do nieadresowanego trybu slave i wprowadzi linie SCL i SDA w stan wysokiej impedancji.

- **Bit 3 – TWWC: znacznik kolizji zapisu TWI**

Bit TWWC jest ustawiany gdy następuje próba zapisu do TWDR kiedy TWINT jest w stanie niskim. Ta flaga jest kasowana przez zapis rejestru TWDR, kiedy TWINT jest w stanie wysokim.

- **Bit 2 – TWEN: bit aktywacji TWI**

Bit TWEN aktywuje działanie interfejsu TWI. Kiedy do TWEN wpisana jest jedynka, TWI przejmuje sterowanie nad pinami wejścia/wyjścia podłączonymi do pinów SCL i SDA, aktywuje ograniczenia slew-rate i filtry krótkich impulsów o dużej amplitudzie. Jeżeli w tym bicie wpiszemy zero, TWI jest wyłączone i wszystkie transmisje TWI są przerywane, łącznie z aktualnie wykonywanymi.

- **Bit 1 – Res: bit zarezerwowany**

Ten bit jest zarezerwowany i zawsze odczytuje się go jako 0.

- **Bit 0 – TWIE: aktywacja przerwań TWI**

Kiedy ten bit jest ustawiony na jeden i I-ty bit w SREG jest ustawiony, żądanie przyjęcia przerwania będzie aktywne tak długo jak TWINT jest w stanie wysokim.

REJESTR STANU TWI – TWSR

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- **Bity 7..3 – TWS: stan TWI**

Tych 5 bitów odzwierciedla stan układu TWI i magistrali. Różne kody statusów zostaną opisane później. Należy zauważyć, że wartość przeczytana z TWSR zawiera zarówno 5-bitową wartość stanu oraz 2-bitowa wartość prescalera. Projektant aplikacji powinien zamaskować bity prescalera na zero podczas sprawdzania bitów stanu. Sprawia to że sprawdzanie stanu jest niezależne od ustawień prescalera.

- **Bit 2 – Res: bit zarezerwowany**

Ten bit jest zarezerwowany i zawsze odczytuje się go jako 0.

- **Bity 1..0 – TWPS: bity TWI prescalera**

Te bity mogą być zapisywane i odczytywane, sterują prescalerem bit rate.

REJESTR DANYCH TWI – TWDR

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

W trybie transmisji TWDR zawiera następny bajt przeznaczony do wysłania. W trybie odbiornika, TWDR zawiera ostatnio odebrany bajt. Jest zapisywalny gdy TWI nie jest w stanie przesuwania bajtu. Zdarza się to wtedy gdy TWINT jest ustawiona przez sprzęt. Należy zauważyć że rejestr danych nie może być zainicjowany przez użytkownika zanim pojawi się pierwsze przerwanie. Dane w TWDR pozostaną stabilne tak długo jak TWINT będzie ustawione. Kiedy dane są przesuwane na zewnątrz, dane na magistrali są przesuwane do wewnątrz. TWDR zawsze zawiera ostatni bajt obecny na magistrali, wyjątkiem jest sytuacja po wzbudzeniu z trybu uśpienia przez przerwanie TWI. W tym przypadku zawartość TWDR jest niezdefiniowana. W przypadku przegrania arbitrażu, żadne dane nie są tracone w transmisji z urządzenia master do slave. Bit ACK jest sterowany automatycznie przez układ TWI, CPU nie ma do niego bezpośredniego dostępu.

- **Bity 7..0 – TWD: rejestr danych TWI**

Tych 8 bitów tworzy następny bajt danych do przesłania lub ostatni bajt danych które zostały odebrane.

TWI (SLAVE) REJESTR ADRESU – TWAR

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

TWAR powinien być załadowany 7-bitowym adresem urządzenia slave (na 7 najbardziej znaczących bitach) do którego TWI będzie się odwoływał gdy zostanie zaprogramowane jako nadajnik lub odbiornik. W systemach multimaster, TWAR musi być ustawiony w urządzeniach master które mogą być adresowane jako urządzenia slave przez inne urządzenia master.

LSB rejestru TWAR jest wykorzystywane do aktywacji rozpoznania adresu wywołania ogólnego. Z rejestrem związany jest komparator adresów który szuka adresów urządzeń slave (lub adresu wywołania ogólnego jeżeli jest aktywne) w odebranym szeregowym adresie. Jeżeli porównanie wypadnie pomyślnie generowane jest żądanie przyjęcia przerwania.

- **Bity 7..1 – TWA: rejestr adresowy TWI (Slave)**

  Tych 7 bitów tworzy adres slave jednostki TWI

- **Bit 0 – TWGCE: Bit TWI aktywacji rozpoznania wywołania ogólnego**

  Jeżeli ten bit jest ustawiony aktywuje rozpoznanie wywołania ogólnego danego na magistralę.

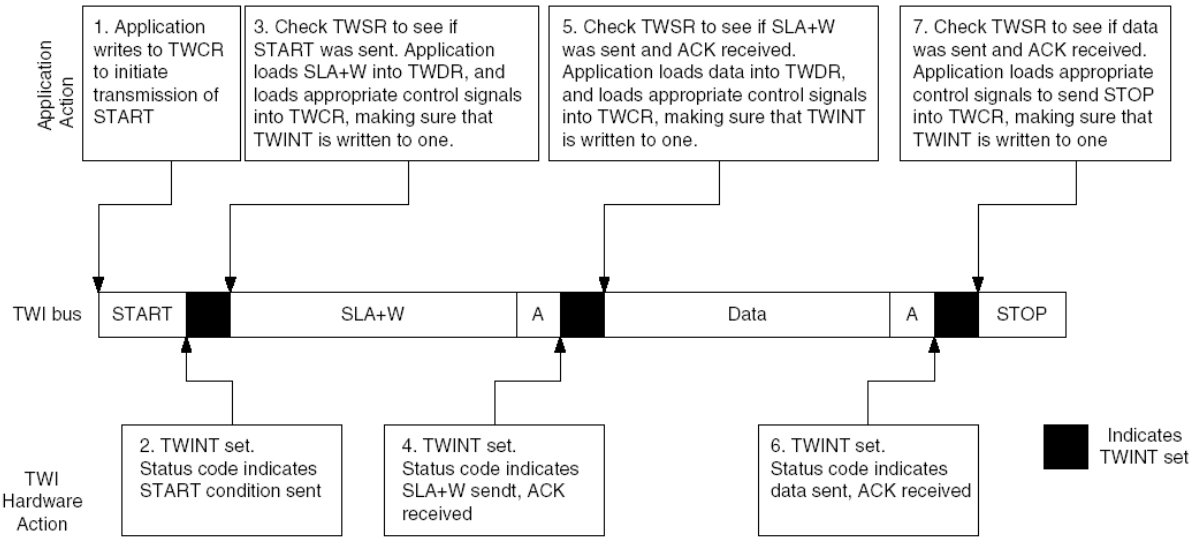
UŻYWANIE TWI

AVR TWI jest zorientowane bajtowo i bazuje na przerwaniach. Przerwania są wydawane po wszystkich zdarzeniach na magistrali takich jak przyjęcie bajtu transmisji lub warunek START. Ponieważ TWI bazuje na przerwaniach, oprogramowanie jest wolne od ciężaru wykonywania innych operacji podczas transferu danych. Należy zauważyć że bit TWIE w TWCR razem z bitem globalnej aktywacji przerwania w SREG pozwalają aplikacji zdecydować czy asercja flagi TWINT powinna generować żądanie przyjęcia przerwania czy nie. Jeżeli bit TWIE jest kasowany, aplikacja musi oddać TWINT żeby wykryć akcję na magistrali TWI.

Kiedy znacznik TWINT jest potwierdzony oznacza to że TWI zakończył operację i oczekuje na odpowiedź aplikacji. W tym przypadku TWSR zawiera wartość wskazującą bieżący stan magistrali TWI. Oprogramowanie może wtedy zdecydować jak TWI powinno się zachować w następnym cyklu magistrali TWI przez manipulację rejestrami TWCR i TWDR.

Rys. 94 jest prostym przykładem interfejsu pomiędzy aplikacją a sprzętem TWI. W tym przykładzie master życzy sobie przesłać pojedynczy bajt danych do urządzenia slave. Ten opis jest raczej abstrakcyjny, więcej szczegółowych wyjaśnień zostanie przedstawionych później. Prosty przykładowy kod implementacji żądanego zachowania również został przedstawiony.

Rys. 94 Interfejs pomiędzy aplikacją a TWI podczas typowej transmisji



1. Pierwszym krokiem w transmisji TWI jest przesłanie warunku START. Robi się to poprzez wpisanie określonej wartości do TWCR, informującej sprzęt TWI, że należy przesłać warunek START. To jaką wartość wpisać zostanie wyjaśnione później. Jednak ważne jest że bit TWINT jest ustawiony we wpisanej wartości. Wpisanie jedynki do TWINT kasuje flagę. TWI nie rozpocznie jakichkolwiek operacji tak długo jak bit TWINT w TWCR będzie ustawiony. Natychmiast po tym jak aplikacja wykasuje TWINT, TWI zainicjuje transmisję warunkiem START.
2. Kiedy warunek START został przesłany , znacznik TWINT w TWCR jest ustawiony i TWSR jest aktualizowany kodem stanu wskazującym że warunek START został pomyślnie wysłany.
3. Oprogramowanie może teraz testować wartość TWSR, aby upewnić się że warunek START został pomyślnie wysłany. Jeżeli TWSR wskaże coś odmiennego, oprogramowanie może podjąć pewne specjalne działania takie jak obsługa błędu. Przyjmując że kod stanu jest taki jak oczekujemy, aplikacja musi załadować SLA+W do TWDR. Należy pamiętać że TWDR jest używany zarówno przez adresy jak i dane. Po tym jak TWDR został załadowany żądanym SLA+W, określona wartość musi być wpisana do TWCR, informująca sprzęt TWI żeby przesłać SLA+W obecne w TWDR. To jaką wartość należy zostanie wpisać opisane później. Jednak ważne jest że bit TWINT jest ustawiony we wpisanej wartości. Wpisanie jedynki do TWINT kasuje flagę. TWI nie rozpocznie jakichkolwiek operacji tak długo jak bit TWINT w TWCR będzie ustawiony. Natychmiast po tym jak aplikacja wykasuje TWINT, TWI zainicjuje transmisję pakietu z adresem.
4. Kiedy pakiet z adresem zostanie przesłany flaga TWINT w TWCR zostanie ustawiona i TWSR jest aktualizowany kodem stanu wskazującym, że pakiet z adresem został pomyślnie wysłany. Kod stanu odzwierciedla również czy urządzenie slave potwierdziło odbiór przesyłki.

5. Oprogramowanie powinno teraz testować wartość TWSR, aby upewnić się że pakiet z adresem został pomyślnie wysłany i że wartość bitu ACK była taka jak oczekiwano. Jeżeli TWSR wskaże coś odmiennego, oprogramowanie może podjąć pewne specjalne działania takie jak obsługa błędu. Przyjmując że kod stanu jest taki jak oczekujemy, aplikacja musi załadować pakiet danych do TWDR. Następnie określona wartość musi być wpisana do TWCR, informująca sprzęt TWI żeby przesłać pakiet danych obecny w TWDR. To jaką wartość należy zostanie wpisać opisane później. Jednak ważne jest że bit TWINT jest ustawiony we wpisanej wartości. Wpisanie jedynki do TWINT kasuje flagę. TWI nie rozpocznie jakichkolwiek operacji tak długo jak bit TWINT w TWCR będzie ustawiony. Natychmiast po tym jak aplikacja wykasuje TWINT, TWI zainicjuje transmisję pakietu danych.
6. Kiedy pakiet danych zostanie przesłany flaga TWINT w TWCR zostanie ustawiona i TWSR jest aktualizowany kodem stanu wskazującym, że pakiet danych został pomyślnie wysłany. Kod stanu odzwierciedla również czy urządzenie slave potwierdziło odbiór przesyłki.
7. Oprogramowanie powinno teraz testować wartość TWSR, aby upewnić się że pakiet danych został pomyślnie wysłany i że wartość bitu ACK była taka jak oczekiwano. Jeżeli TWSR wskaże coś odmiennego, oprogramowanie może podjąć pewne specjalne działania takie jak obsługa błędu. Przyjmując że kod stanu jest taki jak oczekujemy, aplikacja musi wpisać określoną wartość do TWCR, informująca sprzęt TWI żeby przesłać warunek STOP. To jaką wartość należy zostanie wpisać opisane później. Jednak ważne jest że bit TWINT jest ustawiony we wpisanej wartości. Wpisanie jedynki do TWINT kasuje flagę. TWI nie rozpocznie jakichkolwiek operacji tak długo jak bit TWINT w TWCR będzie ustawiony. Natychmiast po tym jak aplikacja wykasuje TWINT, TWI zainicjuje transmisję warunku STOP. Należy zauważyć że TWINT NIE jest ustawiana po tym jak wysłano warunek STOP.

Chociaż ten przykład jest prosty, pokazuje stopień skomplikowania we wszystkich transmisjach. Podsumowując:

- Kiedy TWI zakończył operację i oczekuje odpowiedzi aplikacji, znacznik TWINT jest ustawiany. Linia SCL jest w stanie niskim dopóki TWINT nie zostanie skasowane.
- Kiedy znacznik TWINT jest ustawiony użytkownik musi zaktualizować wszystkie rejestry TWI wartością związaną z następnym cyklem magistrali TWI. W przykładzie TWDR musiał być załadowany wartością przeznaczoną do transmisji w następnym cyklu magistrali.
- Po aktualizacji wszystkich rejestrów TWI i zakończeniu zadań aplikacji TWCR jest zapisywany. Podczas zapisu TWCR bit TWINT powinien być ustawiony. Wpisanie jedynki do

TWINT kasuje flagę. TWI zacznie wtedy wykonywanie jakiejkolwiek operacji określonej przez ustawienie TWCR.

Poniżej podano implementację w asemblerze i C.

	Assembly Code Example	C Example	Comments
1	`ldi r16, (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)` `out TWCR, r16`	`TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)`	Send START condition

2	`wait1:`
	`in r16,TWCR`
	`sbrs r16,TWINT`
	`rjmp wait1`
		`while (!(TWCR & (1<<TWINT)))`
		`;`
			Wait for TWINT flag set. This indicates that the START condition has been transmitted

3	`in r16,TWSR`
	`andi r16, 0xF8`
	`cpi r16, START`
	`brne ERROR`
		`if ((TWSR & 0xF8) != START)`
		`ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR

	`ldi r16, SLA_W`
	`out TWDR, r16`
	`ldi r16, (1<<TWINT) | (1<<TWEN)`
	`out TWCR, r16`
		`TWDR = SLA_W;`
		`TWCR = (1<<TWINT) | (1<<TWEN);`
			Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address

4	`wait2:`
	`in r16,TWCR`
	`sbrs r16,TWINT`
	`rjmp wait2`
		`while (!(TWCR & (1<<TWINT)))`
		`;`
			Wait for TWINT flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.

5	`in r16,TWSR`
	`andi r16, 0xF8`
	`cpi r16, MT_SLA_ACK`
	`brne ERROR`
		`if ((TWSR & 0xF8) != MT_SLA_ACK)`
		`ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR

	`ldi r16, DATA`
	`out TWDR, r16`
	`ldi r16, (1<<TWINT) | (1<<TWEN)`
	`out TWCR, r16`
		`TWDR = DATA;`
		`TWCR = (1<<TWINT) | (1<<TWEN);`
			Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of address

6	`wait3:`
	`in r16,TWCR`
	`sbrs r16,TWINT`
	`rjmp wait3`
		`while (!(TWCR & (1<<TWINT)))`
		`;`
			Wait for TWINT flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.

7	`in r16,TWSR`
	`andi r16, 0xF8`
	`cpi r16, MT_DATA_ACK`
	`brne ERROR`
		`if ((TWSR & 0xF8) != MT_DATA_ACK)`
		`ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR

	`ldi r16, (1<<TWINT)|(1<<TWEN)|(1<<TWSTO)` `out TWCR, r16`	`TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);`	Transmit STOP condition

TRYBY TRANSMISJI

TWI może działać w jednym z czterech trybów: Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) i Slave Receiver (SR). Każdy z tych trybów może zostać użyty w tej samej aplikacji. Na przykład TWI może wykorzystywać tryb MT do zapisu danych do EEPROM TWI, tryb MR do odczytu danych z EEPROM. Jeżeli inne urządzenia master są obecne w systemie, niektóre z nich mogą przesyłać dane to TWI a potem tryb SR mógłby być użyty. Decyzja o tym które tryby są dozwolone należy do oprogramowania.

Kolejne rozdziały opisują każdy z tych trybów. Możliwe kody stanów są opisane poniżej razem ze schematami opisującymi szczegóły transmisji danych w każdym z trybów. Schematy zawierają następujące skróty:

S: warunek START

Rs: warunek REPEATED START

R: czytaj bit (wysoki poziom na SDA)

W: zapisz bit (niski poziom na SDA)

A: potwierdź bit (niski poziom na SDA)

/A: nie potwierdzaj bitu (wysoki poziom na SDA)

Data: bajt danych

P: warunek STOP

SLA: adres urządzenia slave

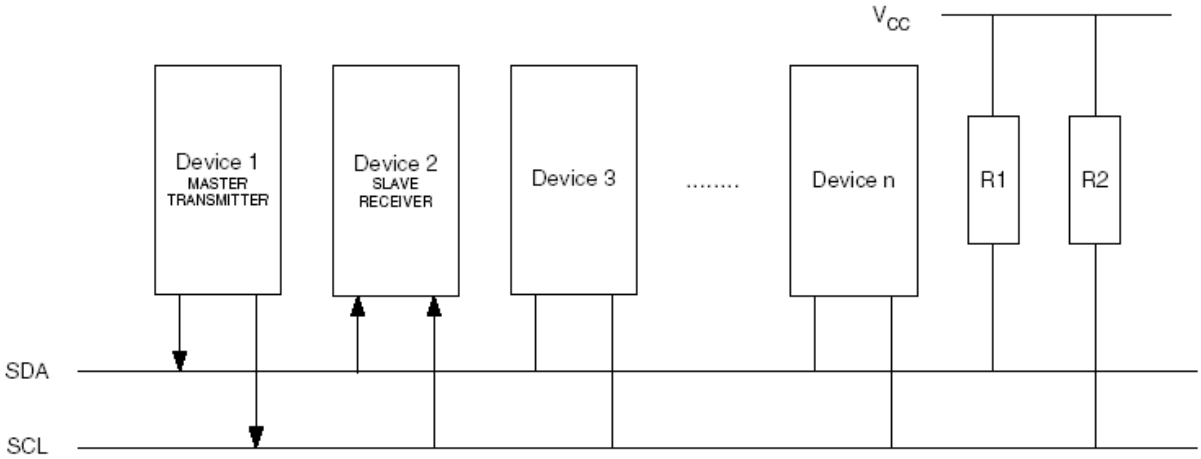
Na schematach 96 i 102 okręgi są wykorzystane aby wskazać że flaga TWINT jest ustawiona. Liczby w okręgach pokazują kod stanu zatrzymany w TWSR, z bitami prescalera zamaskowanymi zerami. W tym punkcie aplikacja musi podjąć działania aby kontynuować lub zakończyć transfer TWI. Transfer TWI jest zawieszony dopóki flaga TWINT nie zostanie skasowana przez oprogramowanie.

Kiedy flaga TWINT jest ustawiona, kod stanu w TWSR jest wykorzystany do określenia stosownych działań aplikacji. Dla każdego kodu stanu, wymagana akcja ze strony oprogramowania oraz szczegóły kolejnego szeregowego transferu są podane w tabelach od 87 do 90. Należy zwrócić uwagę że bity prescalera są zamaskowane zerami w tych tabelach.

TRYB MASTER TRANSMITTER

W trybie MT liczba bajtów danych jest przesyłana do odbiornika slave (zobacz rys. 95). Żeby wejść w tryb master, warunek START musi zostać przesłany. Format kolejnego pakietu z adresem określa w który z trybów (MT czy MR) wejdziemy. Jeżeli SLA+W jest przesyłane, wchodzimy w tryb MT. Wysyłając SLA+R wchodzimy w tryb MR. Wszystkie wspomniane kody stanu zakładają że bity prescalera są równe zero lub są zamaskowane zerami.

Rys. 95 Przesył danych w trybie MT



Warunek START jest wysłany przez wpisanie następujące wartości do TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

TWEN musi zostać ustawione aby aktywować TWI, TWSTA musi zostać przypisana jedynka aby przesłać warunek START oraz TWINT musi być równy jeden aby wykasować flagę TWINT. Wówczas TWI będzie testować magistralę i wygeneruje warunek START tak szybko jak tylko magistrala zostanie zwolniona. Po przesłaniu warunku START, flaga TWINT jest ustawiana sprzętowo i kod stanu w TWSR będzie wynosił $80 (zobacz tabela 87). Żeby wejść w tryb MT, SLA+W musi zostać przesłane. Odbywa się to przez wpisanie SLA+W do TWDR. Od tego czasu bit TWINT powinien być skasowany (przez przypisanie mu jedynki) aby móc kontynuować transfer. Dokonuje się tego przez wpisanie następującej wartości do TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

Kiedy SLA+W zostanie przesłane i otrzymany zostanie bit potwierdzenia, TWINT jest ustawiane ponownie i kod stanu w trybie master może wynosić $18, $20 lub $38. Dla każdej z tych wartości jest przewidziana stosowne działanie. Szczegóły są podane w tabeli 87.

Kiedy SLA+W zostanie pomyślnie przesłane powinien zostać wysłany pakiet danych. Odbywa się to przez wpisanie bajtu danych do TWDR. TWDR musi tylko być zapisane kiedy TWINT jest w stanie wysokim. Jeżeli nie dostęp zostanie zawieszony, a bit TWWC w TWCR zostanie ustawiony. Po aktualizacji TWDR, bit TWINT powinien zostać skasowany (przez przypisanie mu wartości jeden) aby kontynuacja transferu był możliwa. Dokona tego wpisanie następującej wartości do TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

Ten schemat jest powtarzany dopóki ostatni bajt nie zostanie wysłany i nie zostanie zakończona transmisja przez generacje warunku STOP lub REPEATED START. Warunek STOP jest generowany przez wpisanie poniższej wartości do TWCR.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

Warunek REPEATED START jest generowany przez wpisanie poniższej wartości do TWCR.

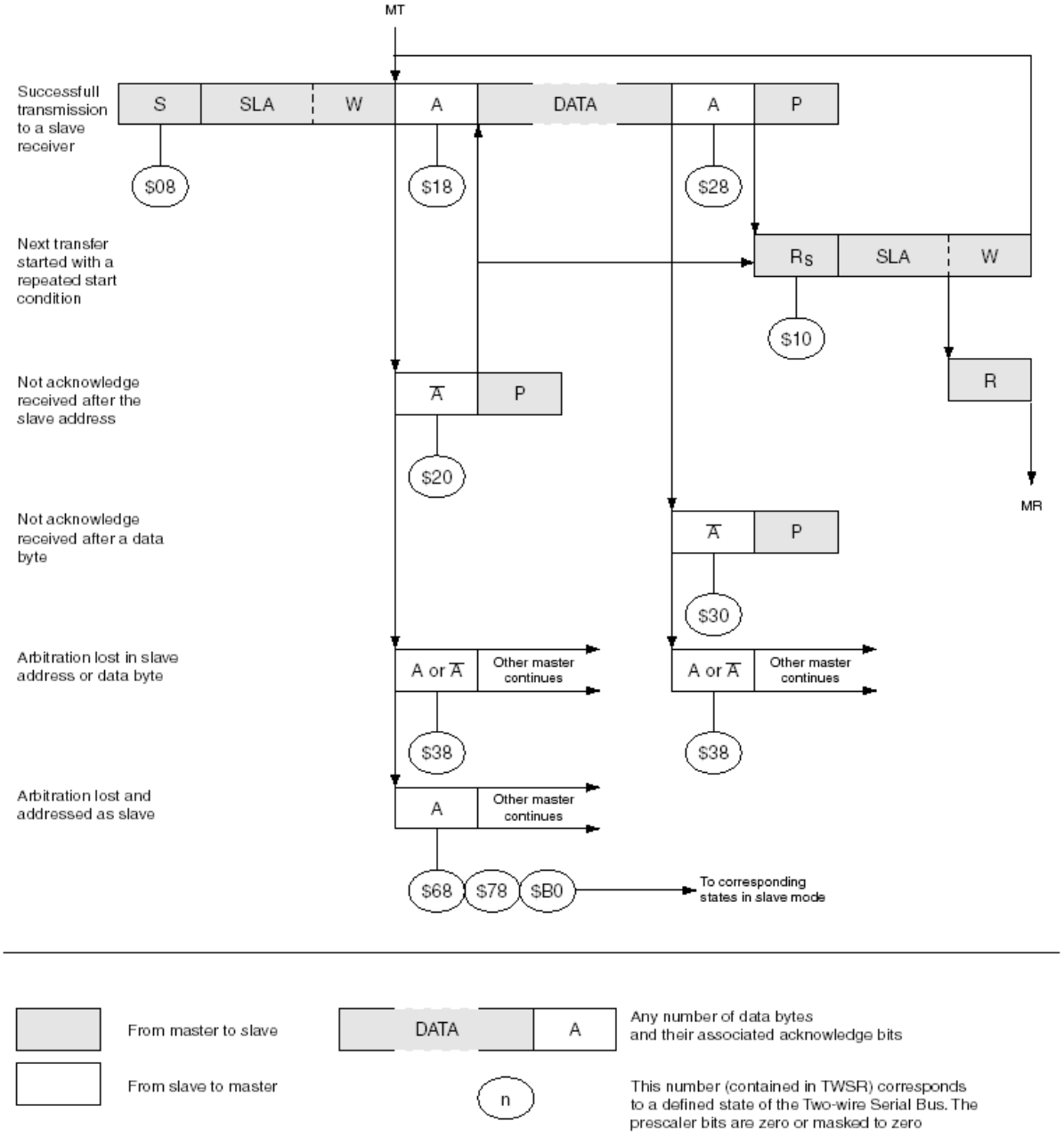
TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

Po warunku REPEATED START (stan $10) TWI może uzyskać dostęp do tego samego urządzenia slave lub do nowego urządzenia slave bez generacji warunku STOP. REPEATED START aktywuje urządzenie master aby przełączać między urządzeniami slave, tryb MT i MR bez utraty sterowania na magistrali.

Tabela 87 Kody stanów dla trybów MT

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$08	A START condition has been transmitted	Load SLA+W	X	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
$10	A repeated START condition has been transmitted	Load SLA+W or	X	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	X	0	1	X	SLA+R will be transmitted; Logic will switch to master receiver mode
$18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
$20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
$28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
$30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
$38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	Two-wire Serial Bus will be released and not addressed slave mode entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free

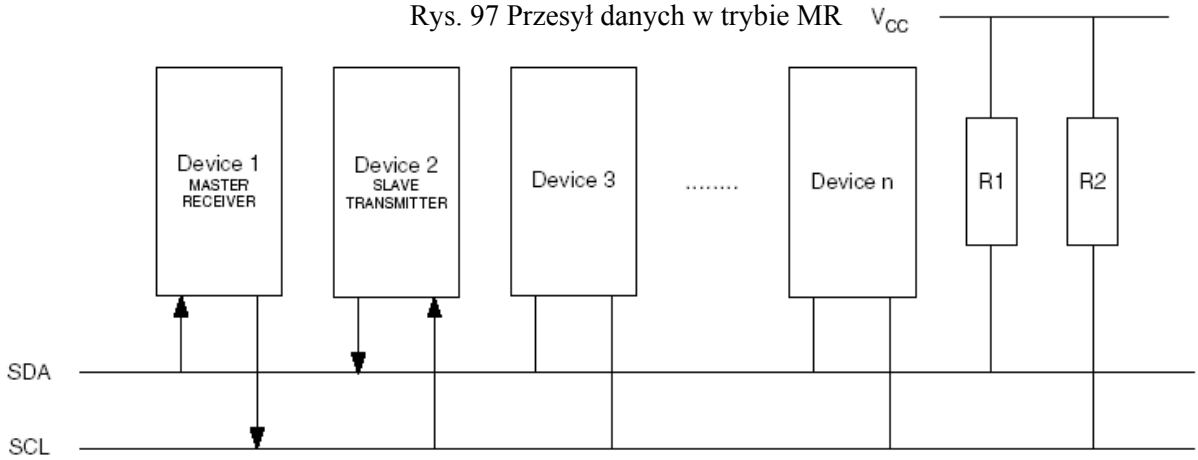
Rys. 96 Formaty i stany w trybie MT



Tryb Master Receiver

W trybie Master Receiver bajty danych są otrzymywane od nadajnika slave (zobacz rys. 97). Ąby wejść w tryb MR musi zostać wysłany warunek START. Format kolejnego pakietu z adresem określa w który z trybów (MT czy MR) wejdziemy. Jeżeli SLA+W jest przesyłane, wchodzimy w tryb MT. Wysyłając SLA+R wchodzimy w tryb MR. Wszystkie wspomniane kody stanu zakładają że bity prescalera są równe zero lub są zamaskowane zerami.

Rys. 97 Przesył danych w trybie MR



Warunek START jest wysłany przez wpisanie następujące wartości do TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

TWEN musi zostać ustawione aby aktywować TWI, TWSTA musi zostać przypisana jedynka aby przesłać warunek START oraz TWINT musi być równy jeden aby wykasować flagę TWINT. Wówczas TWI będzie testować magistralę i wygeneruje warunek START tak szybko jak tylko magistrala zostanie zwolniona. Po przesłaniu warunku START, flaga TWINT jest ustawiana sprzętowo i kod stanu w TWSR będzie wynosił $80 (zobacz tabela 87). Żeby wejść w tryb MR, SLA+R musi zostać przesłane. Odbywa się to przez wpisanie SLA+R do TWDR. Od tego czasu bit TWINT powinien być skasowany (przez przypisanie mu jedynki) aby móc kontynuować transfer. Dokonuje się tego przez wpisanie następującej wartości do TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

Kiedy SLA+R zostanie przesłane i otrzymany zostanie bit potwierdzenia, TWINT jest ustawiane ponownie i kod stanu w trybie master może wynosić $38, $40 lub $48. Dla każdej z tych wartości jest przewidziana stosowne działanie. Szczegóły są podane w tabeli 96. Odebrane dane mogą zostać odczytane z rejestru TWDR kiedy znacznik TWINT jest ustawiony sprzętowo na jeden. Ten schemat jest powtarzany dopóki ostatni bajt nie zostanie odebrany. Po tym jak ostatni bajt został otrzymany MR powinien poinformować ST poprzez wysłanie NACK po ostatnim otrzymanym bajcie danych. Transfer zostaje zakończony po wysłaniu warunku STOP lub REPEATED START. Warunek STOP jest generowany przez wpisanie poniższej wartości do TWCR.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

Warunek REPEATED START jest generowany przez wpisanie poniższej wartości do TWCR.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

Po warunku REPEATED START (stan $10) TWI może uzyskać dostęp do tego samego urządzenia slave lub do nowego urządzenia slave bez generacji warunku STOP. REPEATED START aktywuje urządzenie master aby przełączać między urządzeniami slave, tryb MT i MR bez utraty sterowania na magistrali.

Rys. 98 Formaty i stany w trybie MR

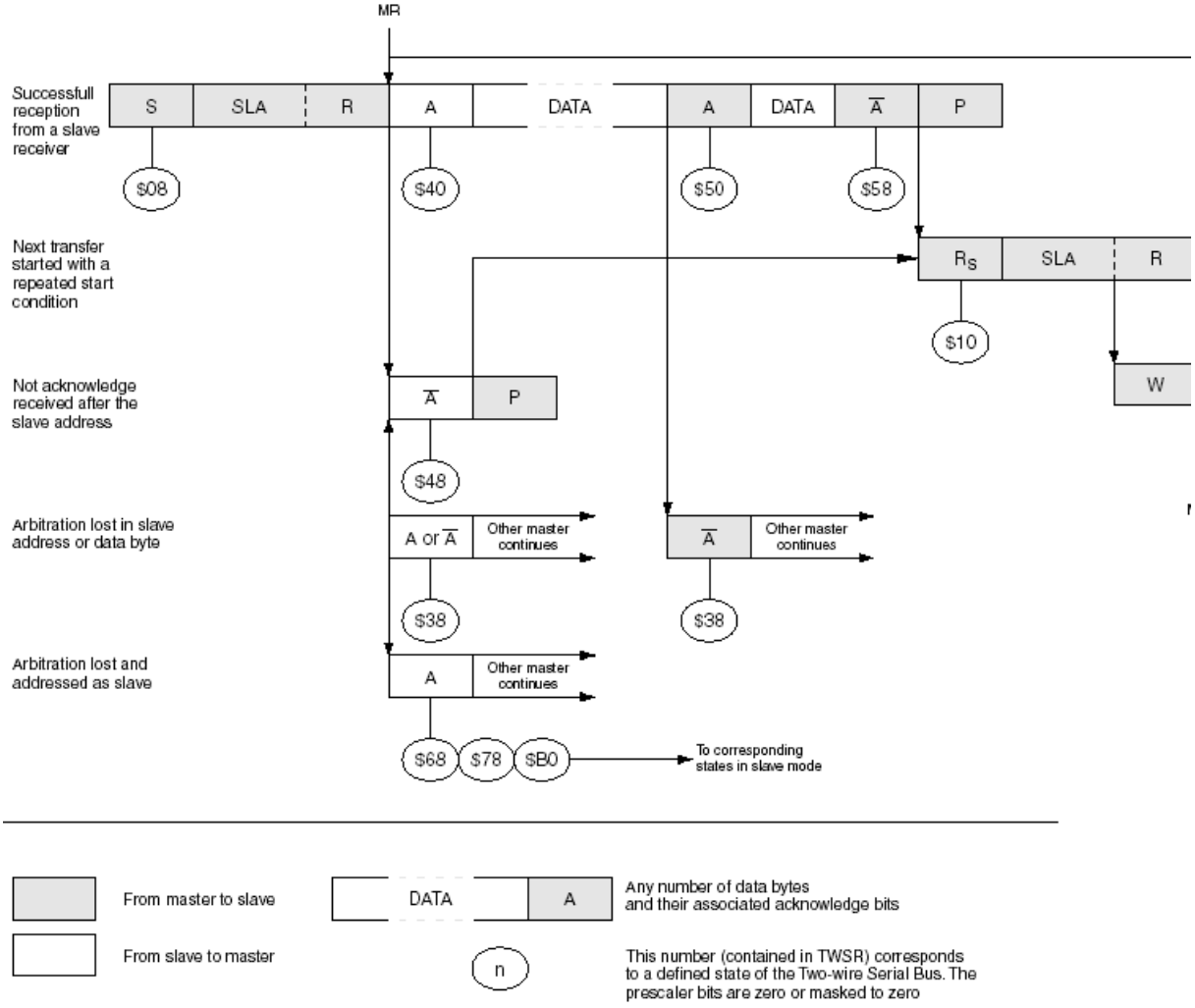


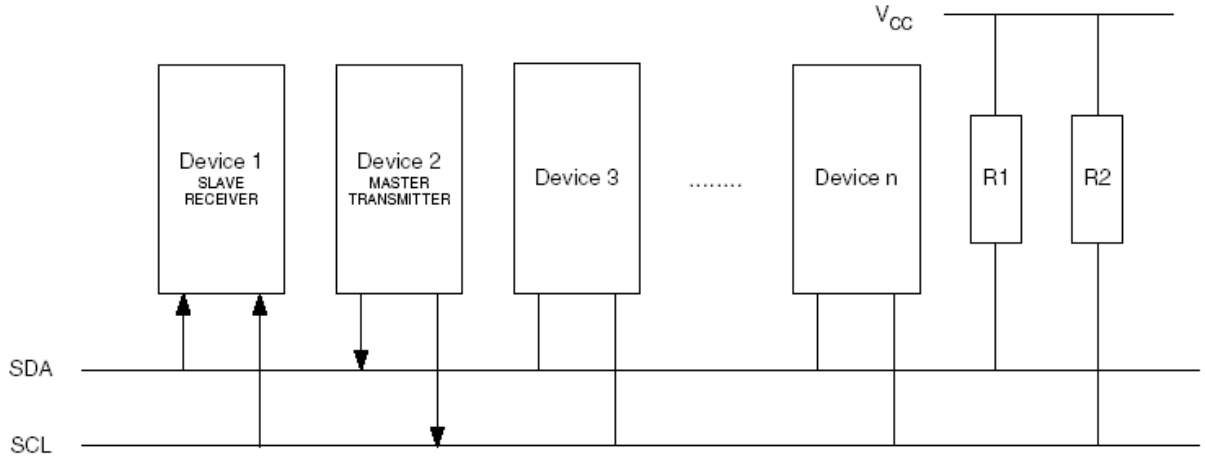
Tabela 88 Kody stanu dla trybu MR

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$08	A START condition has been transmitted	Load SLA+R	X	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
$10	A repeated START condition has been transmitted	Load SLA+R or	X	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	X	0	1	X	SLA+W will be transmitted Logic will switch to master transmitter mode
$38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	Two-wire Serial Bus will be released and not addressed slave mode will be entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free
$40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
$48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
$50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
$58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset

TRYB SLAVE RECEIVER

W trybie Slave Receiver bajty danych są otrzymywane od nadajnika master (zobacz rys. 99). Wszystkie wspomniane kody stanu zakładają że bity prescalera są równe zero lub są zamaskowane zerami.

Rys. 99 Przesył danych w trybie SR



Aby zainicjować tryb Slave Receiver, TWAR i TWCR muszą zostać zainicjowane w następujący sposób:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
value	Device's Own Slave Address							

Starsze 7-bitów jest adresem urządzenia slave, który jeżeli zostanie wygenerowany przez urządzenie master spowoduje że TWI odpowie. Jeżeli LSB jest ustawiony TWI będzie odpowiadał na adres wywołania ogólnego ($00), w przeciwnym wypadku wywołania ogólne będą ignorowane.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	0	1	0	0	0	1	0	X

Na bit TWEN musi zostać wpisana jedynka aby aktywować TWI. Bit TWEA również musi być ustawiony na jeden aby aktywować potwierdzenie własności adresu lub wywołania ogólnego. Na bity TWSTA i TWSTO należy wpisać zero.

Jeżeli TWAR i TWCR zostały zainicjowane, TWI czeka aż na magistrali pojawi się jego adres (lub adres wywołania ogólnego jeżeli jest aktywne) po którym przesłany zostanie bit kierunku przepływu danych. Jeżeli bit kierunku jest równy 0 (zapis) TWI będzie działać w trybie SR, w przeciwnym razie przejdziemy w tryb ST. Po otrzymaniu adresu i bitu kierunku znacznik TWINT zostanie ustawiony i ważny stan kodu będzie mógł być odczytany z TWSR. Kod stanu jest używany do określenia stosownej akcji ze strony oprogramowania. Szczegóły działań podjętych przez aplikację są zawarte w tabeli 89. Urządzenie może także wejść w tryb SR jeżeli przegra proces arbitrażu podczas gdy TWI jest w trybie master (zobacz stany $68 i $78).

Jeżeli bit TWEA zostanie zresetowany w czasie trwania transferu TWI zwróci „Not Acknowledge” („1”) na SDA po następnym otrzymanym bajcie danych. Może to zostać wykorzystane do wskazania że slave nie jest w stanie odebrać już więcej bajtów. Gdy TWEA jest równe 0 TWI nie potwierdzi swojego własnego adresu. Jednakże magistrala jest ciągle monitorowana i rozpoznawanie adresu może zostać w każdej chwili wznowione przez ustawienie TWEA. Oznacza to że bit TWEA może być użyty do tymczasowej izolacji TWI od magistrali.

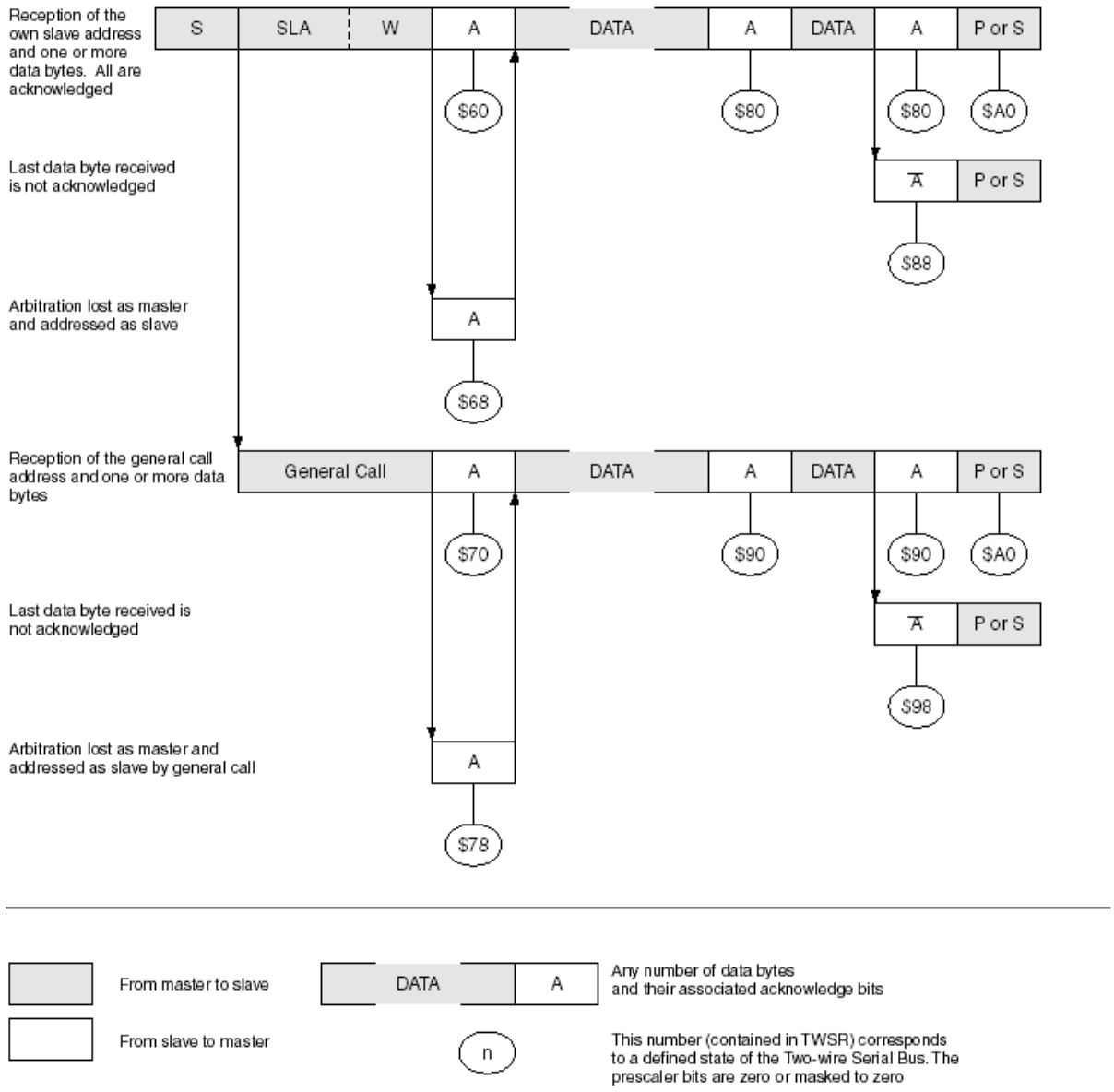
We wszystkich trybach uśpienia i w trybie bezczynności zegar systemowy do TWI jest wyłączony. Jeżeli bit TWEA jest ustawiony interfejs ciągle może potwierdzać swój własny adres lub wywołanie ogólne używając magistrali zegara jako źródła zegarowego. Część będzie wzbudzona ze stanu uśpienia i TWI utrzyma zegar SCL w stanie niskim podczas budzenia do momentu w którym flaga TWINT zostanie skasowana. Przyjmowanie przyszłych danych będzie wykonywane normalnie z normalnie uruchomionym zegarem AVR. Należy zwrócić uwagę że jeżeli AVR jest ustawione na długi czas uruchomienia, linia SCL może być utrzymana w stanie niskim przez długi czas blokując tym samym transmisję danych.

TWDR nie odzwierciedla ostatniego bajtu obecnego na magistrali podczas wzbudzania z trybów uśpienia.

Tabela 89 Kody stanów dla trybu SR

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$68	Arbitration lost in SLA+R/W as master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$78	Arbitration lost in SLA+R/W as master; General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
$88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
$90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
$98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
$A0	A STOP condition or repeated START condition has been received while still addressed as slave	Read data byte or	0	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free

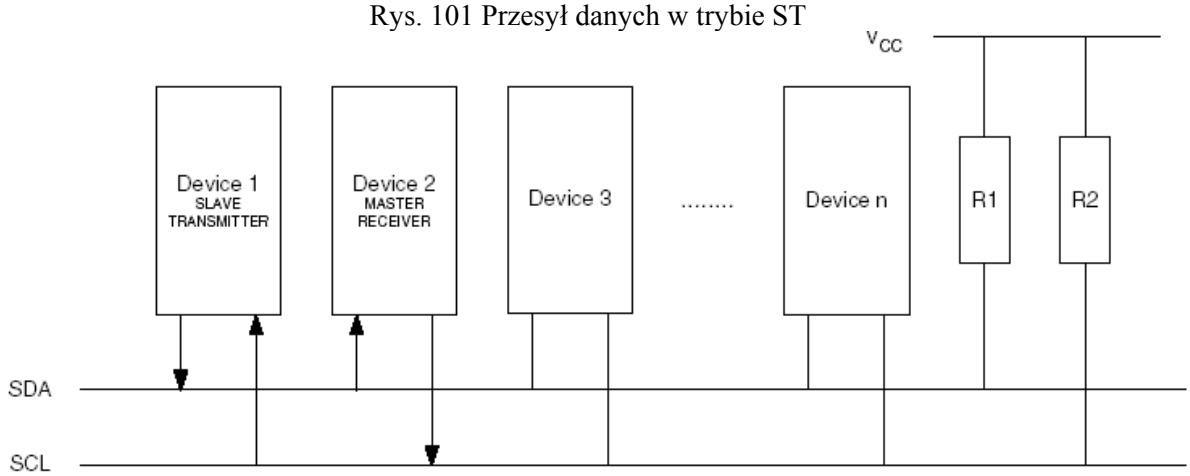
Rys. 100 Formaty i stany w trybie SR



TRYB SLAVE TRANSMITTER

W trybie Slave Transmitter bajty danych są wysyłane do odbiornika master (zobacz rys. 101). Wszystkie wspomniane kody stanu zakładają że bity prescalera są równe zero lub są zamaskowane zerami.

Rys. 101 Przesył danych w trybie ST



Aby zainicjować tryb Slave Transmitter, TWAR i TWCR muszą zostać zainicjowane w następujący sposób:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
value	Device's Own Slave Address							

Starsze 7-bitów jest adresem urządzenia slave, który jeżeli zostanie wygenerowany przez urządzenie master spowoduje że TWI odpowie. Jeżeli LSB jest ustawiony TWI będzie odpowiadał na adres wywołania ogólnego ($00), w przeciwnym wypadku wywołania ogólne będą ignorowane.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	0	1	0	0	0	1	0	X

Na bit TWEN musi zostać wpisana jedynka aby aktywować TWI. Bit TWEA również musi być ustawiony na jeden aby aktywować potwierdzenie własności adresu lub wywołania ogólnego. Na bity TWSTA i TWSTO należy wpisać zero.

Jeżeli TWAR i TWCR zostały zainicjowane, TWI czeka aż na magistrali pojawi się jego adres (lub adres wywołania ogólnego jeżeli jest aktywne) po którym przesłany zostanie bit kierunku przepływu danych. Jeżeli bit kierunku jest równy 1 (odczyt) TWI będzie działać w trybie ST, w przeciwnym razie przejdziemy w tryb SR. Po otrzymaniu adresu i bitu kierunku znacznik TWINT zostanie ustawiony i ważny stan kodu będzie mógł być odczytany z TWSR. Kod stanu jest używany do określenia stosownej akcji ze strony oprogramowania. Szczegóły działań podjętych przez aplikację są zawarte w tabeli 90. Urządzenie może także wejść w tryb ST jeżeli przegra proces arbitrażu podczas gdy TWI jest w trybie master (zobacz stan $B0).

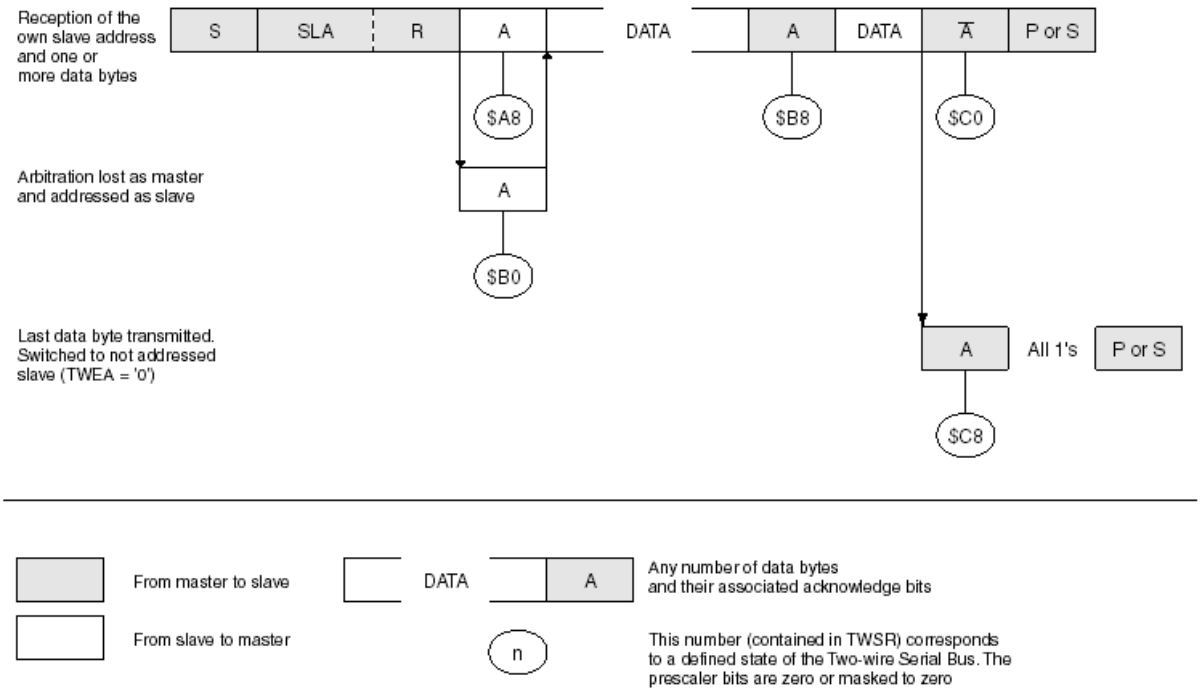
Jeżeli na bit TWEA zostanie wpisane 0 podczas transferu, TWI prześle ostatni bajt transferu. Wejdziemy w stan $C0 lub $C8 zależnie od tego czy odbiornik master prześle NACK czy ACK po końcowym bajcie. TWI zostaje przełączone do nieadresowalnego trybu slave i będzie ignorować urządzenie master podczas kontynuowania przez nie transmisji. Z tego powodu MR będzie otrzymywać jedynki. W stan $C8

wejdziemy gdy master wymaga dodatkowych bajtów danych (przez przesłanie ACK), nawet wtedy gdy slave przesłał wszystkie bajty (TWEA na zero i oczekiwanie na NACK od urządzenia master).
Gdy TWEA jest ustawione na 0 podczas transferu TWI nie odpowie na swój własny adres. Jednakże magistrala jest ciągle monitorowana i rozpoznawanie adresu może zostać w każdej chwili wznowione przez ustawienie TWEA. Oznacza to że bit TWEA może być użyty do tymczasowej izolacji TWI od magistrali.
We wszystkich trybach uśpienia i w trybie bezczynności zegar systemowy do TWI jest wyłączony. Jeżeli bit TWEA jest ustawiony interfejs ciągle może potwierdzać swój własny adres lub wywołanie ogólne używając magistrali zegara jako źródła zegarowego. Część będzie wzbudzona ze stanu uśpienia i TWI utrzyma zegar SCL w stanie niskim podczas budzenia do momentu w którym flaga TWINT zostanie skasowana. Przyjmowanie przyszłych danych będzie wykonywane normalnie z normalnie uruchomionym zegarem AVR. Należy zwrócić uwagę że jeżeli AVR jest ustawione na długi czas uruchomienia, linia SCL może być utrzymana w stanie niskim przez długi czas blokując tym samym transmisję danych.
TWDR nie odzwierciedla ostatniego bajtu obecnego na magistrali podczas wzbudzania z trybów uśpienia.

Tabela 90 Kody stanów dla trybu ST

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$A8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$B0	Arbitration lost in SLA+R/W as master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$B8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
$C0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
$C8	Last data byte in TWDR has been transmitted (TWEA = "0"); ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free

Rys. 102 Formaty i stany dla trybu ST



RÓŻNE STANY

Są dwa kody stanów nie odpowiadające zdefiniowanym stanom TWI (zobacz tabela 91).

Stan $F8 wskazuje że żadne istotne informacje nie są dostępne ponieważ znacznik TWINT nie jest ustawiony. Takie zdarzenie ma miejsce pomiędzy innymi stanami oraz gdy TWI nie jest uwikłany w szeregowy transfer.

Stan $00 wskazuje że pojawił się błąd magistrali podczas transferu. Tego typu bład zdarza się gdy warunki START i STOP pojawią się na niedozwolonej pozycji w formacie ramki. Przykłady takich niedozwolonych pozycji zdarzają się podczas szeregowego transferu bajtu adresu, bajtu danych czy bitu potwierdzenia. Kiedy ma miejsce błąd TWINT zostaje ustawione. Aby powrócić ze stanu błędnego, należy ustawić flagę TWSTO i skasować TWINT. Spowoduje to że TWI wejdzie w tryb nieadresowanego slave'a i skasuje znacznik TWSTO (inne bity w TWCR nie ulegną zmianie). Linie SDA i SCL zostaną zwolnione i zostanie wygenerowany warunek STOP.

Tabela 91 Różne stany

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$F8	No relevant state information available; TWINT = "0"	No TWDR action	No TWCR action				Wait or proceed current transfer
$00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

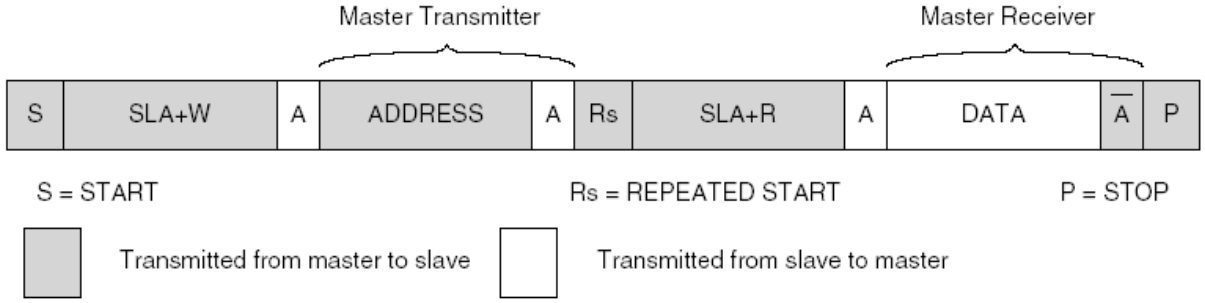
TRYBY ZŁOŻONE Z KILKU TRYBÓW TWI

Istnieją sytuacje, w których tryby TWI muszą zostać połączone żeby zakończyć pewne działanie. Rozważmy na przykład odczyt danych z szeregowego EEPROM-u. Zwykle tego typu transfer przebiega w następujący sposób:

1. Przesył musi zostać zainicjowany
2. EEPROM musi zostać poinstruowany, które miejsce powinno być odczytane
3. Odczyt musi być przygotowany
4. Transfer musi się zakończyć

Należy zauważyć że dane są przesyłane zarówno od urządzenia master do urządzenia slave jak i odwrotnie. Master musi poinstruować slave'a która lokację chce odczytać, co wymaga użycia trybu MT. Następnie dane musza zostać odczytane ze slave'a co oznacza użycie trybu MR. Dlatego kierunek przesyłu musi zostać zmieniony. Master musi utrzymać sterowanie nad magistralą podczas tych wszystkich kroków, które należy uważać za operacje niepodzielne. Gdy ta zasada nie jest przestrzegana w systemach multimaster inne urządzenie master może zmienić wskaźnik danych w EEPROM-ie pomiędzy krokami 2 i 3, wówczas master odczyta dane ze złego miejsca. Taka zmiana kierunku przesyłu jest dokonana poprzez przesłanie REPEATED START pomiędzy transmisją bajtu adresu a przyjęciem danych. Po REPEATED START utrzymuje magistralę w swoim posiadaniu. Poniższy rysunek pokazuje przepływ w tym przesyle.

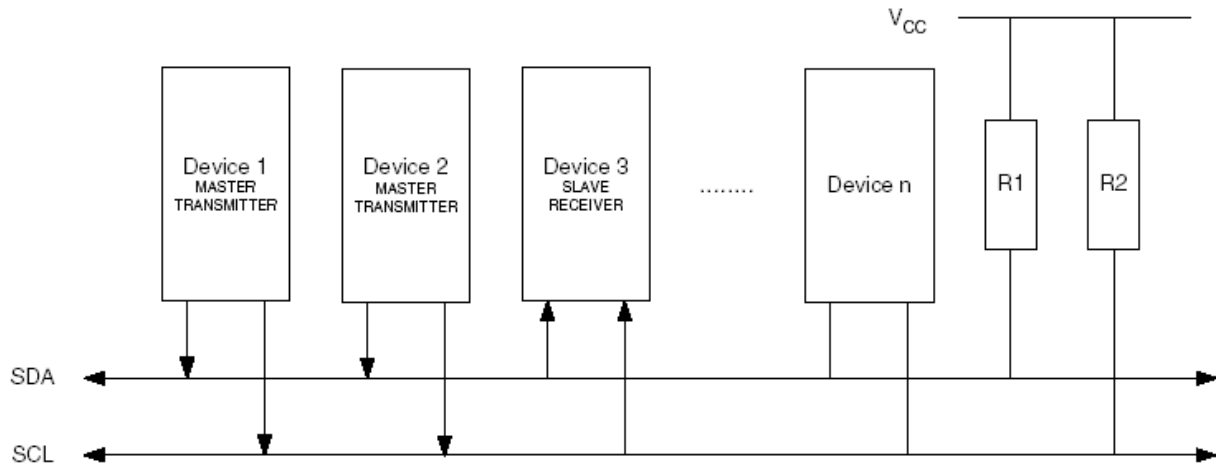
Rys. 103 Tryby złożone z kilku trybów TWI uzyskujące dostęp do EEPROM-u



SYSTEMY MULTI-MASTER I ARBITRAŻ

Jeżeli wiele urządzeń master jest podpiętych do tej samej magistrali, transmisja może zostać zainicjowana równocześnie przez jedno lub kilka z nich. Standard TWI radzi sobie z tego typu sytuacją i zapewnia że tylko jednemu urządzeniu master pozwoli się kontynuować transfer oraz że żadne dane nie zostaną utracone w tym procesie. Przykład sytuacji arbitrażu jest narysowany poniżej, gdzie dwa urządzenia master próbują przesyłać dane do odbiornika slave.

Rys. 104 Przykład arbitrażu

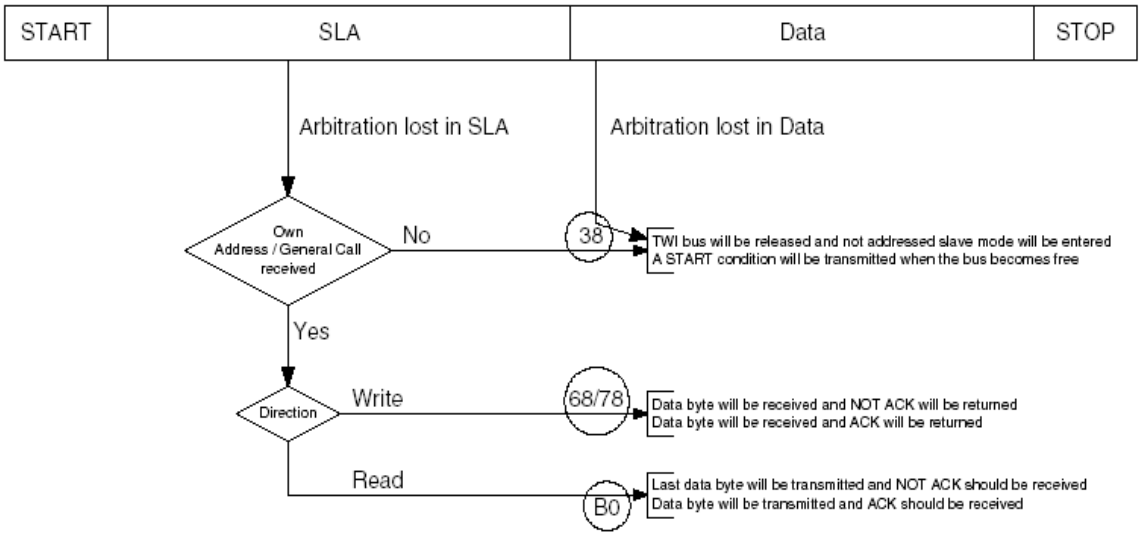


Jak opisano poniżej, kilka różnych sytuacji może się pojawić podczas arbitrażu:

- Dwa lub więcej urządzeń master rozpoczęło identyczną komunikację z tym samym urządzeniem slave. W tym przypadku ani slave ani żadne z urządzeń master nie będą wiedziały o konflikcie na magistrali.
- Dwa lub więcej urządzeń master chce uzyskać dostęp do tego samego slave'a z różnymi danymi i różnymi bitami kierunku. W tym przypadku pojawi się arbitraż. Urządzenia master starają się wyprowadzić jedynkę na SDA podczas gdy inne urządzenie master wyprowadzi zero i utraci arbitraż. Przegrywające urządzenia master przełączą się w tryb nieadresowanych urządzeń slave lub będą czekać na zwolnienie magistrali i prześlą nowy warunek START co zależy od rozwiązania przyjętego w oprogramowaniu.
- Dwa lub więcej urządzeń master chce uzyskać dostęp do różnych slave'ów. W tym przypadku arbitraż będzie dotyczył bitów SLA. Urządzenia master próbują wyprowadzić jedynkę na SDA podczas gdy inne urządzenie masterwyprowadza zero i przegrywa arbitraż. Przegrywające urządzenia master przełączą się w tryb slave aby sprawdzić czy są adresowane przez zwycięskie urządzenie master. Jeżeli tak przełączą się w tryb SR do ST zależnie od wartości bitu READ/WRITE. Jeżeli te urządzenia nie zostały zaadresowane przełączą się w tryb nieadresowanych urządzeń slave lub będą czekać na zwolnienie magistrali i prześlą nowy warunek START co zależy od rozwiązania przyjętego w oprogramowaniu.

Rys. 105 zawiera podsumowanie wszystkich powyższych sytuacji. Możliwe wartości stanów są podane w okręgach.

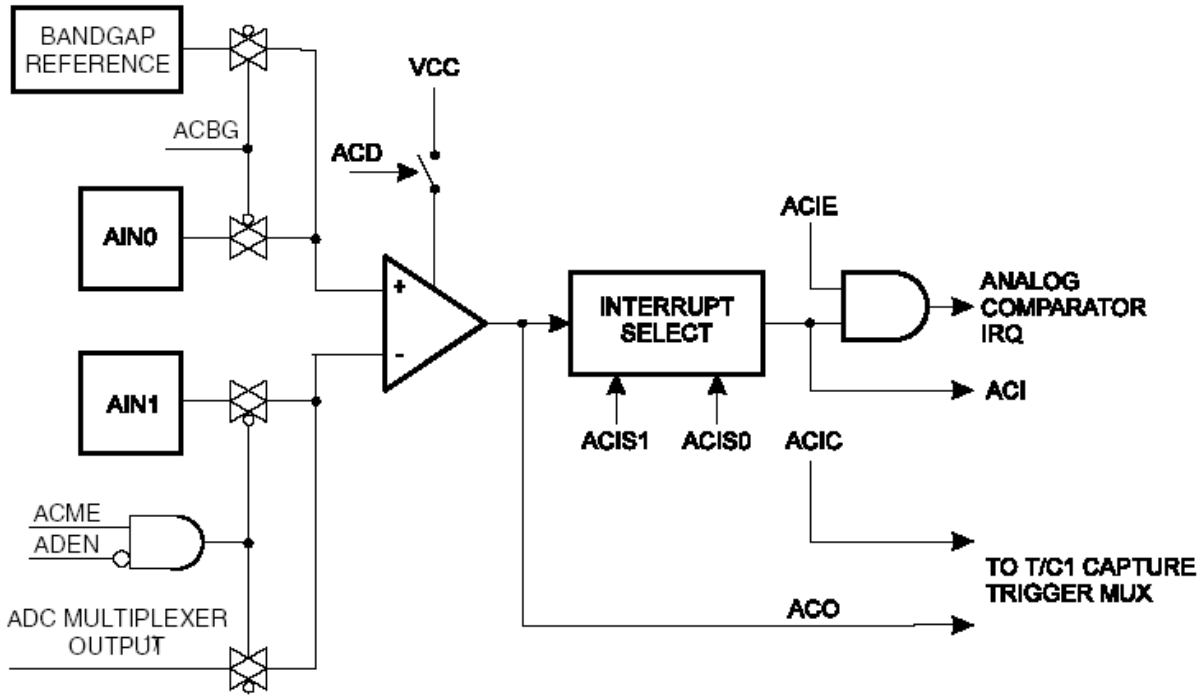
Rys. 105 Możliwe kody stanu spowodowane przez arbitraż



KOMPARATOR ANALOGOWY

Komparator analogowy porównuje wartości wejściowe na pinie dodatnim AIN0 oraz ujemnym AIN1. Kiedy napięcie na pinie AIN0 jest większe niż na ujemnym pinie AIN1, wyjście analogowego komparatora ACO zostanie ustawione. Wyjście komparatora może zostać użyte do wyzwolenia wejścia układu czasowo-licznikowego 1. Dodatkowo komparator może wyzwalać poszczególne przerwania. Użytkownik może wybrać wyzwalanie przerwań na wyjściu komparatora zboczem narastającym, opadającym lub poziomem. Schemat blokowy komparatora i otaczających go układów został przedstawiony na rys. 106.

Rys. 106 Schemat blokowy komparatora analogowego



REJESTR WEJŚCIA/WYJŚCIA SPECJALNEGO PRZEZNACZENIA - SFIOR

Bit	7	6	5	4	3	2	1	0	
	TSM	–	–	ADHSM	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 3 – ACME: Aktywacja multipleksera komparatora analogowego**

Kiedy temu bitowi jest przypisana logiczna jedynka i ADC jest wyłączone (ADEN w ADCSRA jest równe zero) multiplekser ADC wybiera ujemne wejście komparatora analogowego. Kiedy ten bit jest ustawiony na zero, AIN1 jest przyłożony do ujemnego wejścia komparatora analogowego. Aby uzyskać dodatkowe informacje na temat tego bitu należy zapoznać się z tabelą 93.

REJESTR STERUJĄCY I STANU KOMPARATORA ANALOGOWEGO – ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

Bit 7 ACD Analog Comparator Disable

Jeśli ten bit ma wartość jeden zasilanie komparatota jest wyłączone. Jedynka może zostać wpisana w każdej chwili, co pozwala na zmniejszenie poboru mocy. Przy zmianie tego bitu przerwanie komparatora analogowego musi byś wyłączone (w rejestrze ACSR bit ACIE = 0)

Bit 6 ACBG Analog Comparator Bandgap Select
Jeśli ACBG = 1 napięcie odniesienia zastępuje dodatnie wejście komparatora analogowego.

Bit 5 ACO Analog Comparator Output
Wyjście komparatora jest synchronizowane a następnie połączone wprost z ACO. Synchronizacja zajmuje 1-2 cykle zegarowe.

Bit 4 ACI Analog Comparator Interrupt Flag
Ten bit jest ustawiany sprzętowo jeśli wystąpi przerwanie zdefiniowane przez bity **ACIS1 i ACIS0.** Procedura obsługi przerwania jest wykonana jeśli zawartości bitów ACIE i I (w rej. SREG) są równe jeden. ACI jest zerowany sprzętowo lub przez wpisanie logicznej jedynki do flagi.

Bit 3 ACIE Analog Comparator Interrupt Enable
Jeśli ACIE = 1 i bit I w rejestrze stanu jest równy 1 przerwanie komparatora analogowego jest aktywne. Jeśli ACIE = 0 przerwanie jest wyłączone.

Bit 2 ACIC Analog Comparator Input Capture Enable
Jeśli ten bit jest równy jeden wyjście komparatora analogowego jest wprost połączone z wejściem licznika 1. Po wpisaniu w ten bit zera nie ma połączenia między licznikien1 a komparatorem analogowym

Bity 1 i 0 ACIS1, ACIS0: Analog Comparator Interrupt Mode Select

Tabela 92. Ustawienia bitów ACIS1 i ACIS0.

ACIS1	ACIS0	Tryb przerwania
0	0	Przerwanie komparatora
0	1	ZAREZERWOWANE
1	0	Przerwanie komparatora na opadającym zboczu wyjścia
1	1	Przerwanie komparatora na narastającym zboczu wyjścia

Kiedy zmieniamy bity ACIS1/ACIS0 przerwanie komparatora analogowego musi być wyłączone przez skasowanie bitu aktywacji przerwań w rejestrze ACSR. W przeciwnym wypadku przerwanie może się pojawić kiedy bity są zmieniane.

WEJŚCIE MULTIPLEKSOWE KOMPARATORA ANALOGOWEGO
Istnieje możliwość wyboru któregokolwiek z pinów ADC7..0 w celu zastąpienia ujemnego wejścia komparatora analogowego. Multiplekser ADC jest wykorzystywany do wyboru tego wejścia i w konsekwencji ADC musi być wyłączone aby wykorzystać tą cechę. Jeżeli bit aktywacji multipleksera komparatora analogowego (ACME w rejestrze SFOR) jest ustawiony i

ADC jest wyłączone (ADEN w ADCSRA jest równe zero), MUX2..0 w ADMUX wybierają pin wejścia zastępujący ujemne wejście komparatora analogowego jak pokazano w tabeli 93. Jeżeli ACME jest skasowane lub ADEN jest ustawione, AIN1 jest podłączone do ujemnego wejścia komparatora analogowego.

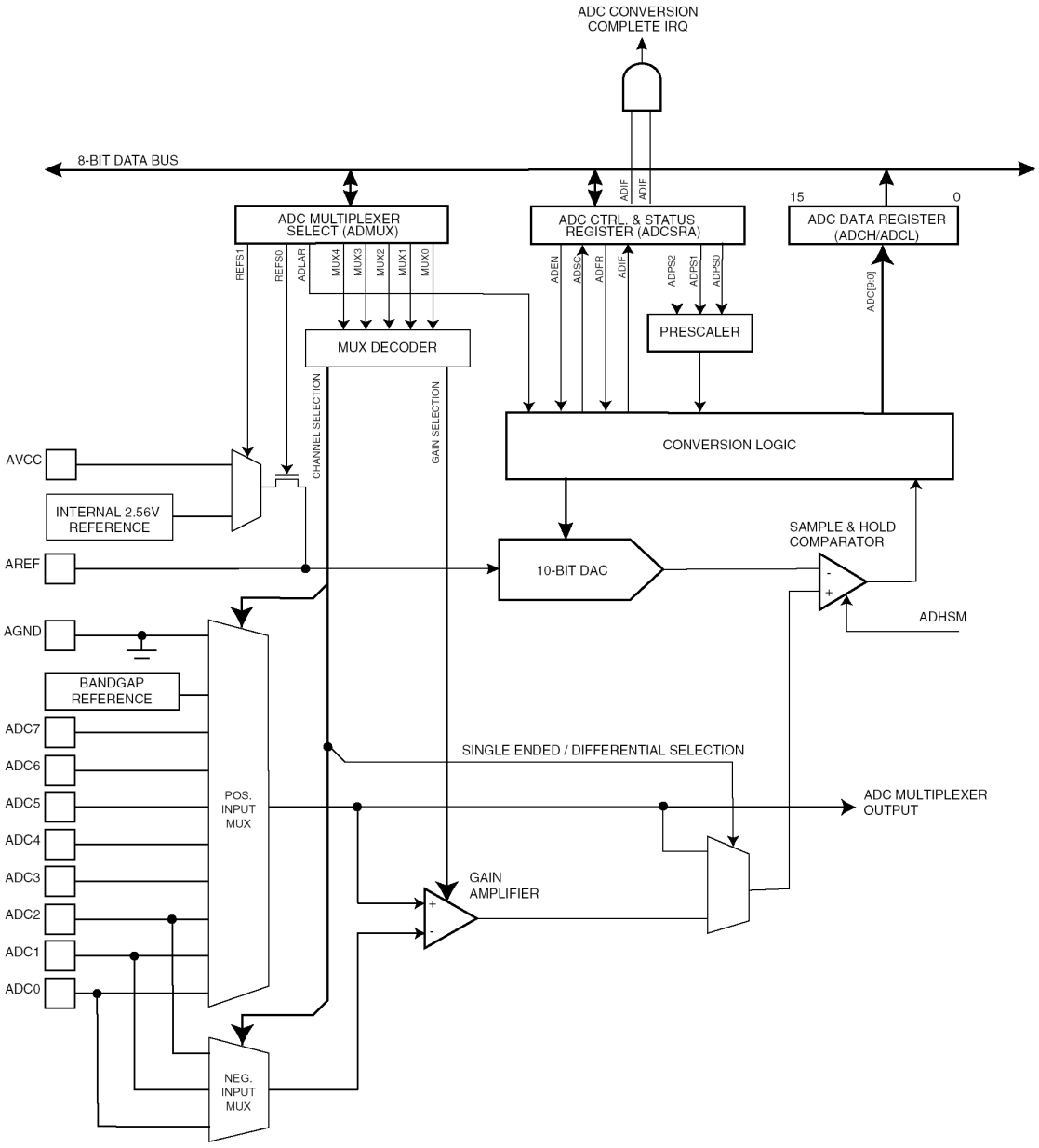
TABELA 93. Multipleksowane wejście komparatora analogowego

ACME	ADEN	MUX2..0	Analog Comparator Negative Input
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

KONWERTER ANALOGOWO CYFROWY (ADC- Analog to Digital Converter)

Konwerter analogowo cyfrowy jest połączony z 8 kanałowym multiplekserem analogowym, na którego 8 niesymetrycznych wejść są podawane sygnały napięciowe z wyprowadzeń portu F. Napięcia wejść niesymetrycznych są podawane względem masy. ADS wspiera także 16 kombinacji napięciowych wejść różnicowych. Dwa z wejść różnicowych (ADC1, ADC0 i ADC3, ADC2) są wyposażone w programowalny stopień wzmacniający pozwalający na wzmocnienie sygnału z wejścia różnicowego : 0 dB (1x) , 20 dB (10x) lub 46 dB (200x). Wzmocnienie jest dokonywane przed konwersją. Siedem analogowych różnicowych wejść używa wspólnej ujemnej końcówki (ADC1),, natomiast wyprowadzenie dodatnie może być wybrane z pozostałych wejść ADC
Jeśli sygnał wejściowy jest wzmacniany 1x lub 10x należy spodziewać się 8 bitowej rozdzielczości, jeśli wzmocnienie wynosi 200x rozdzielczość jest 7 bitowa.
Napięcie wejściowe jest utrzymywane na stałym poziomie podczas konwersji.
Konwerter analogowo cyfrowy posiada oddzielne wyprowadzenie zasilania AVCC. Napięcie AVCC nie może różnić się więcej niż ± 0.3V od VCC

Rys. 107. Schemat blokowy konwertera analogowo cyfrowego



OBSŁUGA

ADC konwertuje analogowy sygnał wejściowy do 10 bitowej wartości wyjściowej. Minimalna wartość wejścia odpowiada masie, wartość maksymalna wartość jest równa napięciu na wyprowadzeniu AREF minus 1 LSB. Opcjonalnie AVCC lub wewnętrzne napięcie odniesienia może być połączone z wyprowadzeniem AREF. Aby tak było należy wpisać „1” do bitów REFSn w rejestrze ADMUX. Analogowe wejście i wzmocnienie różnicowe są wybierane przez wpisanie odpowiednich wartości do rejestru ADMUX. Wszystkie wyprowadzenia wejściowe ADC (w tym masa)
mogą być wybrane jako niesymetryczne wejścia konwertera. Wyprowadzenie wejściowe może być wybrane jako dodatnie lub ujemne wejście wzmacniacza różnicowego.

Jeśli są wybrane kanały różnicowe, napięcie między tą wybraną parą wejść zostaje wzmocniona i przekazana na analogowe wejście ADC. Jeśli wybrane są wejścia niesymetryczne wzmacniacz różnicowy jest pomijany.

Przetwornik analogowo cyfrowy jest włączany przez ustawienie bitu ADEN = 1 w rejestrze ADCSRA. Przetwornik nie pobiera energii jeśli ADEN = 0 dlatego zalecane jest wyłączenie ADC przed przejściem do trybu uśpienia.

ADC generuje 10-bitowy wynik, który zostaje umieszczony w rejestrach danych ADC: ADCH i ADCL. Domyślnie wynik jest wyrównany do prawej jednak może być wyrównany do lewej przez ustawienie na 1 bitu ADLAR w ADMUX.

Jeśli wynik jest wyrównany do lewej i nie jest wymagana precyzja większa od 8 bitowej wystarczający jest odczyt rejestru ADCH. W przeciwnym wypadku najpierw należy odczytać rejestr ADCL a następnie ADCH aby upewnić się czy zawartość obu rejestrów pochodzi z tej samej konwersji.

Zaraz po odczycie ADCL dostęp konwertera do rejestrów danych jest blokowany. Oznacza to, że jeśli zawartość ADCL została odczytana i konwersja została ukończona przed odczytem ADCH zawartość żadnego z rejestrów się zostanie zaktualizowana i wyniki konwersji zostaną utracone. Po odczycie ADCH konwerter odzyskuje dostęp do rejestrów danych(ADCH i ADCL)

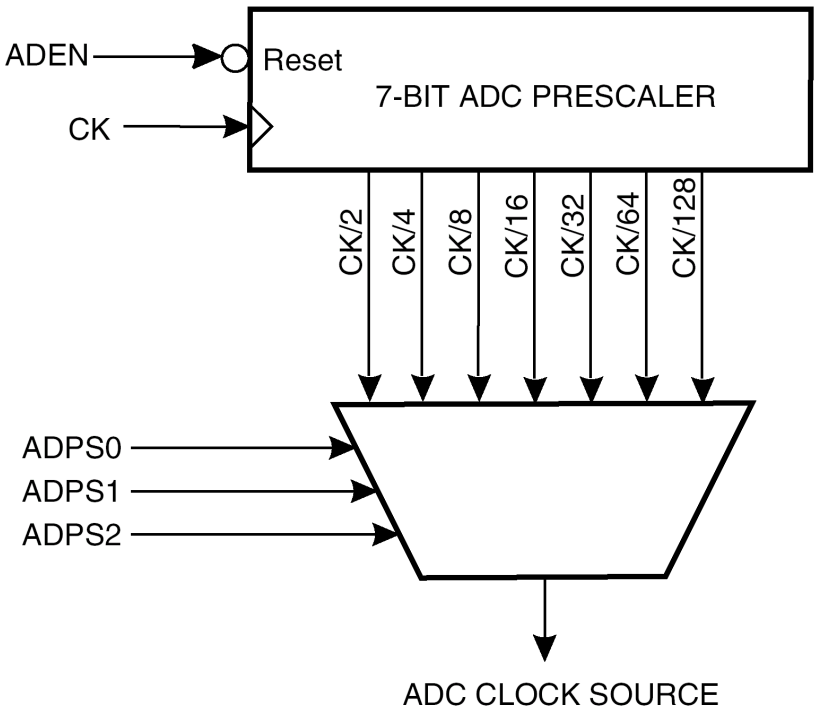
ADC posiada własne przerwanie, które może być wywołane po ukończeniu konwersji. W czasie kiedy ADC nie ma dostępu do rejestrów danych(między odczytem ADCH i ADCL) przerwanie zostanie wywołane nawet jeśli wynik konwersji zostanie utracony

ROZPOCZĘCIE KONWERSJI

Pojedynczą konwersję rozpoczyna się przez wpisanie 1 do bitu ADC Start Conversion w rejestrze ADSC. Wartość tego bitu pozostaje = 1 tak długo, jak długo trwa proces konwersji, i zostaje zmieniona na 0 po ukończeniu konwersji. Jeśli kanał danych jest wybrany podczas konwersji, ADC ukończy bieżącą konwersję przed zmianą kanału.

W trybie “Free Running Mode” ADC stale próbkuje i uaktualnia rejestry danych. Ten tryb jest wybierany przez wpisanie jedynki do bitu ADFR w rejestrze ADCSRA. Pierwsza konwersja musi się rozpocząć od wpisania jedynki do bitu ADSC w rejestrze ADCSRA.

Rys. 108. Prescaler przetwornika ADC



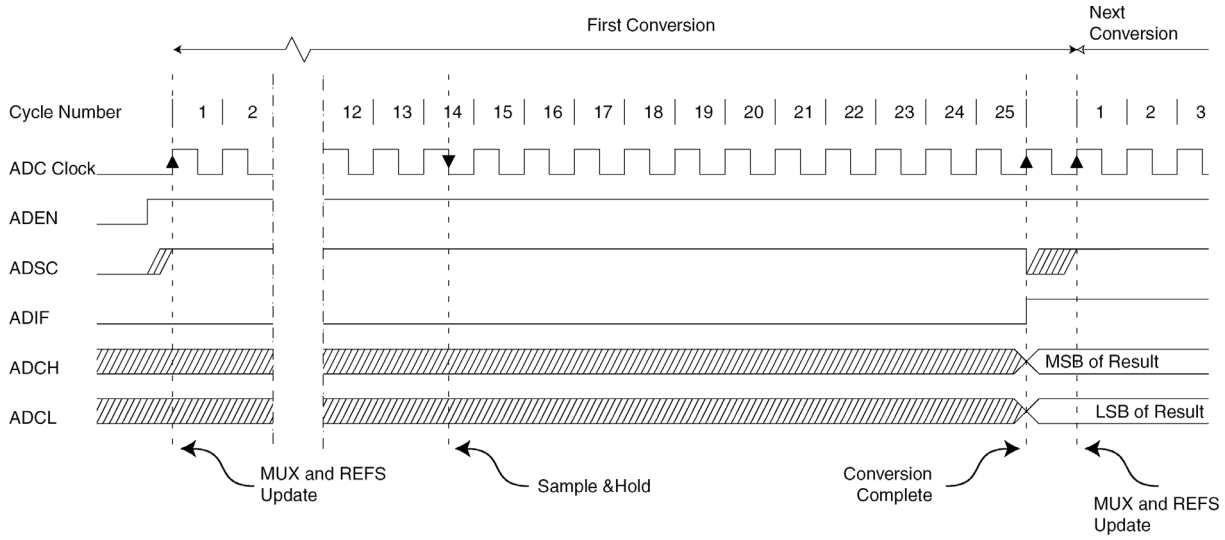
Domyślnie, w celu uzyskania największej rozdzielczości ADC wymaga wejściowego impulsu zegarowego o częstotliwości 50 kHz – 200 kHz. Dla rozdzielczości niższej od 10 bitowej częstotliwość zegara może być wyższa do 200 kHz w celu uzyskania wyższej częstotliwości próbkowania. Można również, przez ustawienie na 1 bitu ADHSM w rej. SFIOR, zwiększyć częstotliwość zegarową, co powoduje jednak zwiększony pobór mocy.
Moduł ADC zawiera prescaler, który generuje odpowiednią dla ADC częstotliwość zegarową z dowolnego przebiegu zegarowego procesora powyżej 100 kHz. Prescaler jest uaktywniany przez wpisanie jedynki do bitu ADPS w rejestrze ACDSRA. Prescaler rozpoczyna zliczanie po włączeniu konwertera (wpisanie do bitu ADEN jedynki), kończy po wpisaniu do ADEN zera.

Po wpisaniu wartości 1 do bitu ADSC w rejestrze ADCSRA konwersja rozpoczyna się po narastającym zboczu sygnału zegarowego ADC

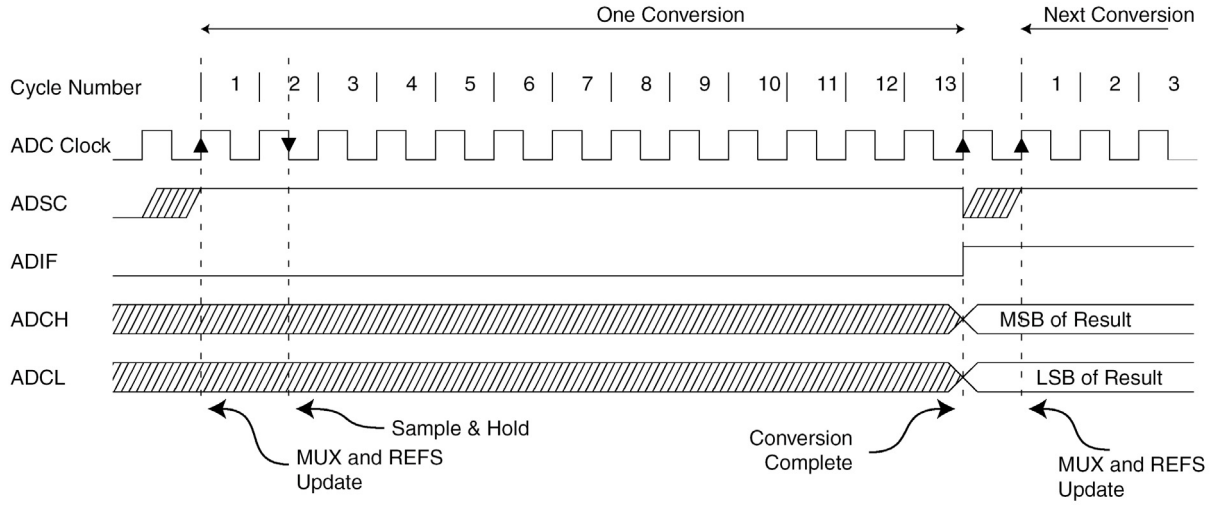
Konwersja zajmuje 13 cykli zegara konwertera, wyjątkiem jest pierwsza konwersja po włączeniu przetwornika, która trwa 25 cykli zegara.

Rzeczywiste próbkowanie ma miejsce po 1,5 cyklu zegara ADC od rozpoczęcia konwersji (13,5 cykli od rozpoczęcia pierwszej konwersji po włączeniu przetwornika analogowo cyfrowego). Po zakończeniu konwersji wynik jest wpisywany do rejestrów danych konwertera a bit ADIF jest ustawiany na jeden; w trybie pojedynczej konwersji równocześnie jest zerowany bit ADSC. Możliwe jest ponowne programowe ustawienie bitu ADSC w wyniku czego po pierwszym narastającym zboczu sygnału zegarowego ADC rozpocznie się następna konwersja. W trybie “Free Running Mode” następna konwersja rozpoczyna się bezpośrednio po ukończeniu poprzedniej konwersji.

Rys. 109. Przebieg czasowy pierwszej konwersji (tryb pojedynczej konwersji)



Rys. 110. Przebieg czasowy pojedynczej konwersji.



Rys 111. Tryb “Free Running Mode” przebieg czasowy

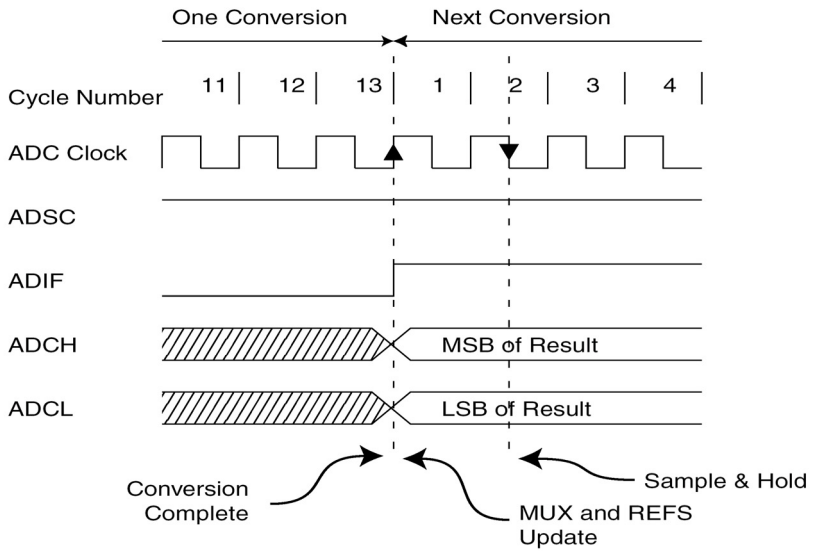


Tabela 94. Czasy konwersji

	Próbkowanie (ilość cykli od rozpoczęcia konwersji)	Czas konwersji (cykle zegara ADC)
Pierwsza konwersja	14,5	25
Normalna konwersja, wejście niesymetryczne	1,5	13
Normalna konwersja, wejście różnicowe	1,5/2,5	13/14

Konwersje różnicowe są synchronizowane przez wewnętrzny zegar CKADC2 o częstotliwości równej połowie częstotliwości zegara ADC. Synchronizacja jest automatyczna.

Stopień wzmacniający jest optymalizowany dla częstotliwości 4 kHz, wyższa częstotliwość może spowodować nieliniowość wzmocnienia. Zewnętrzny filtr dolnoprzepustowy powinien być używany jeśli sygnał wejściowy zawiera składowe o częstotliwości większej od częstotliwości optymalnej stopnia wzmacniającego. Uwaga: częstotliwość zegara ADC nie zależy od częstotliwości stopnia wzmacniającego.

ZMIANA WYBORU KANAŁU LUB ODNIESIENIA

Bity MUXn i REFS1:0 w rejestrze ADMUX są buforowane przez tymczasowy rejestr do którego procesor ma swobodny dostęp. Zapewnia to, że zmiany kanałów i odniesienia będą wykonane w bezpiecznym momencie podczas konwersji. Wybór kanału oraz odniesienia jest stale aktualizowany aż do rozpoczęcia konwersji. Od momentu rozpoczęcia konwersji wybór kanału oraz odniesienia jest zablokowany w celu zapewnienia wystarczającego czasu próbkowania dla ADC. Stałe aktualizowanie jest odblokowywane w ostatnim cyklu zegara przetwornika przed ukończeniem konwersji (ADIF i ADCSRA są ustawiane na 1)

Ostrożność należy zachować przy zmianie kanałów różnicowych. Po wyborze kanału różnicowego stopień wzmacniający potrzebuje 125 µs na stabilizację nowej wartości. Konwersja nie powinna być rozpoczynana w czasie tych 125 µs. Wyniki uzyskane w tym przedziale czasu powinny zostać odrzucone.

Ta sama sytuacja ma miejsce w przypadku pierwszej konwersji różnicowej po zmianie odniesienia ADC

Przy zmianie kanału należy postępować według następujących wskazówek:

W trybie konwersji niesymetrycznej wybór kanału musi zawsze zostać dokonany przed rozpoczęciem konwersji. Wybór kanału może być zmieniony jeden cykl zegara ADC po wpisaniu jedynki do bitu ADSC, jednak prostszą metodą jest poczekać na zakończenie konwersji przed zmianą kanału.

W trybie "Free Running Mode" wybór kanału należy wykonać przed rozpoczęciem pierwszej konwersji. Podobnie jak w trybie konwersji niesymetrycznej wybór kanału może być zmieniony jeden cykl zegara ADC po wpisaniu jedynki do bitu ADSC, jednak prostszą metodą jest poczekać na zakończenie pierwszej konwersji przed zmianą kanału.

Przy przełączaniu kanału różnicowego przy aktywnym wzmacniaczu rezultaty pierwszej konwersji mogą mieć dużą niedokładność z powodu czasu stabilizacji wzmacniacza.

.

NAPIĘCIE ODNIESIENIA

Napięcie odniesienia dla przetwornika analogowo cyfrowego (V_{REF}) wskazuje skalę konwersji dla przetwornika. W przypadku kanału niesymetrycznego napięcie przekraczające V_{REF} prowadzi do otrzymania kodu zbliżonego do 0x3FF. V_{REF} może być wybrane z: AV_{CC}, wewnętrznego napięcia odniesienia 2,56V lub z zewnętrznego wyprowadzenia AREF. AVCC jest połączone zADC przez przełącznik. Wewnętrzne napięcie odniesienia 2,56V jest generowane przez wewnętrzny wzmacniacz z napięcia V_{BG} . W innym przypadku zewnętrzne wyprowadzenie AREF jest wprost połączone z ADC w wyniku czego napięcie odniesienia jest bardziej odporne na zakłócenia (połączenie wyprowadzenia AREF przez kondensator z masą)

V_{REF} może być mierzony na wyprowadzeniu AREF za pomocą woltomierza o wysokiej impedancji.

Wynik pierwszej konwersji po zmianie napięcia odniesienia może być błędny i powinien zostać odrzucony.

REDUKTOR ZAKŁÓCEŃ

Konwerter analogowo cyfrowy posiada reduktor zakłóceń umożliwiający konwersję w trybie uśpienia w celu zmniejszenia zakłóceń pochodzących z rdzenia procesora oraz z urządzeń peryferyjnych. Reduktor zakłóceń może być użyty w trybie redukcji zakłóceń i w trybie bezczynności. W celu użycia reduktora należy wykonać następujące czynności:

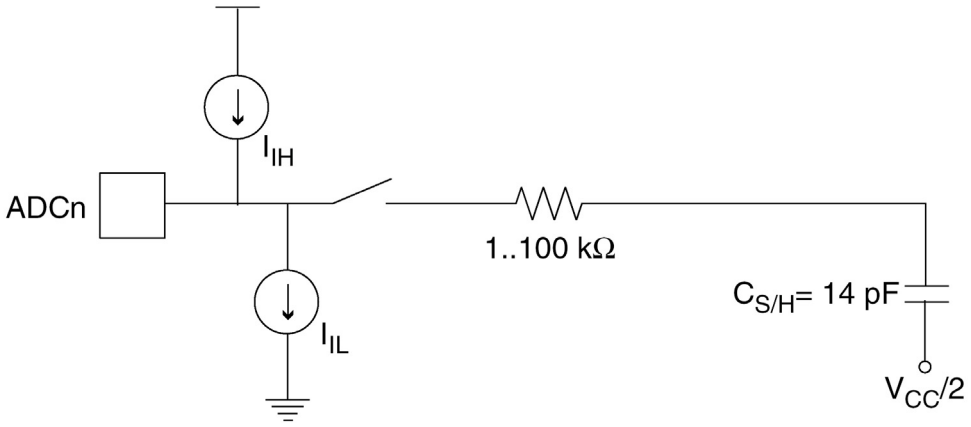
- Upewnić się, czy ADC jest włączony i nie jest zajęty konwersją. Przetwornik analogowo cyfrowy musi znajdować się w trybie pojedynczej konwersji, przerwanie ukończenia konwersji musi być włączone.
- Włączyć tryb redukcji zakłóceń (lub tryb bezczynności). ADC rozpocznie konwersję kiedy procesor zostanie zatrzymany.
- Jeśli nie wystąpią żadne przerwania przed ukończeniem konwersji, przerwanie ukończenia konwersji „obudzi" procesor i zostanie wykonana procedura obsługi przerwania ukończenia konwersji. Jeśli wystąpiło jakieś przerwanie podczas konwersji zostanie ono wykonane, po czym konwersja zostanie dokończona. Po ukończeniu konwersji zostanie wygenerowane przerwanie ukończenia konwersji.

ADC nie jest automatycznie wyłączany po wejściu do trybu uśpienia lub bezczynności. Użytkownik powinien wpisać zero do bitu ADEN przed wejściem do tych trybów w celu uniknięcia nadmiernego poboru mocy. Jeśli ADC jest włączony w trybach uśpienia i użytkownik chcę przeprowadzić konwersją różnicową zalecane jest wyłączenie a następnie włączenie i wyjście z trybu uśpienia co pozwala na uzyskanie poprawnych wyników.

OBWÓD WEJŚCIA ANALOGOWEGO

Obwód wejścia analogowego dla kanału niesymetrycznego jest przedstawiony na rysunku 112.
Źródło sygnału analogowego
Przetwornik analogowo cyfrowy jest optymalizowany dla źródeł sygnałów analogowych o impedancji wyjściowej 10 kΩ lub mniejszej. Jeśli takie źródło jest używane czas próbkowania jest nieistotny. Jeśli źródło ma większą impedancję wyjściową czas próbkowania zależy od czasu jaki źródło potrzebuje do naładowania kondensatora S/H. Zalecane jest używanie źródeł o wysokiej impedancji jedynie w przypadku sygnałów wolnozmiennych ponieważ minimalizuje to przepływ ładunku do kondensatora S/H.
Jeśli są używane kanały różnicowe obwód wejściowy wygląda nieco inaczej mino, że zalecana impedancja wyjściowa źródła wynosi kilkaset kΩ lub mniej. Nie powinny występować składowe sygnału o częstotliwości większej od $f_{ADC}/2$ z powodu możliwości wystąpienia zniekształceń. Zalecane jest usuniecie składowych o zbyt wysokich częstotliwościach za pomocą filtru dolnoprzepustowego.

Rys. 112.Obwód wejścia analogowego.

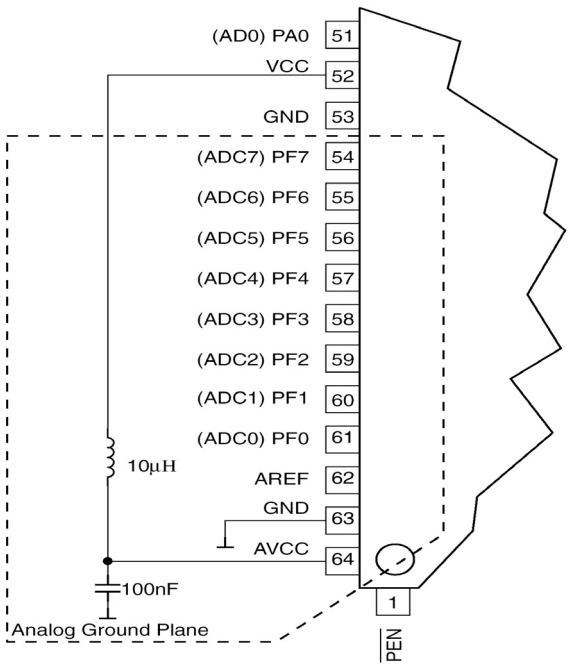


SPOSOBY REDUKCJI ZAKŁÓCEŃ SYGNAŁÓW ANALOGOWYCH

Cyfrowe obwody wewnątrz I na zewnątrz urządzenia generują zakłócenia, które mogą wpłynąć na dokładność konwersji. Poziom zakłóceń może zostać zmniejszony za pomocą następujących sposobów.:

- Ścieżki sygnałów analogowych powinny być jak najkrótsze, powinny bać jak najdalej położone od szybko przełączanych ścieżek cyfrowych.
- Wyprowadzenie AVCC powinien być połączony z zasilaniem V_{CC} za pomocą obwodu LC przedstawionego na rysunku 113
- Użycie reduktora zakłóceń
- Jeśli którekolwiek z wyprowadzeń portu ADC są używane jako wyjścia cyfrowe jest niezwykle istotne, by nie były przełączane podczas konwersji.

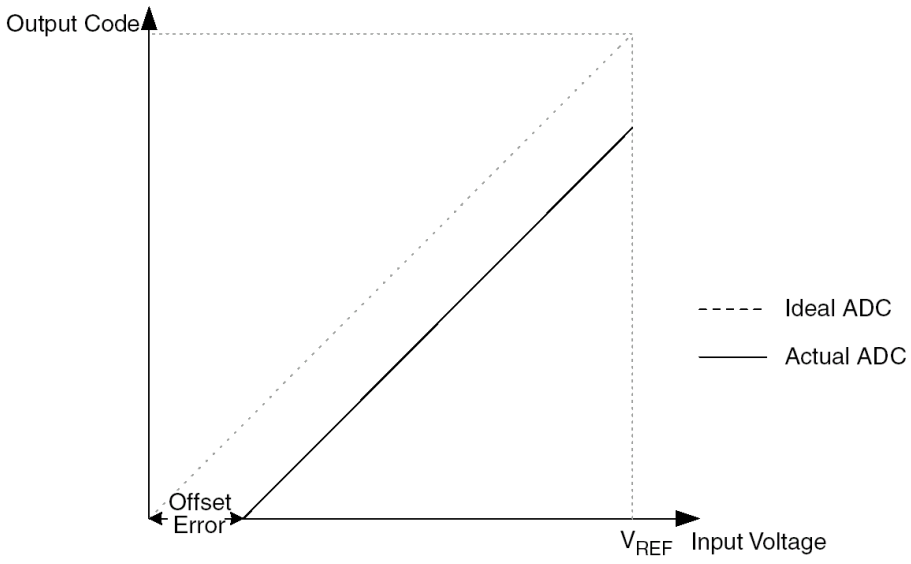
Rys. 113.



Konwersja n-bitowa polega na przekształceniu napięcia o wartości miedzy 0 (masą) a V_{REF} w 2^n krokach. Najniższy kod wynosi 0, najwyższy 2^{n-1}. Następujące parametry określają odchylenie od konwersji idealnej:

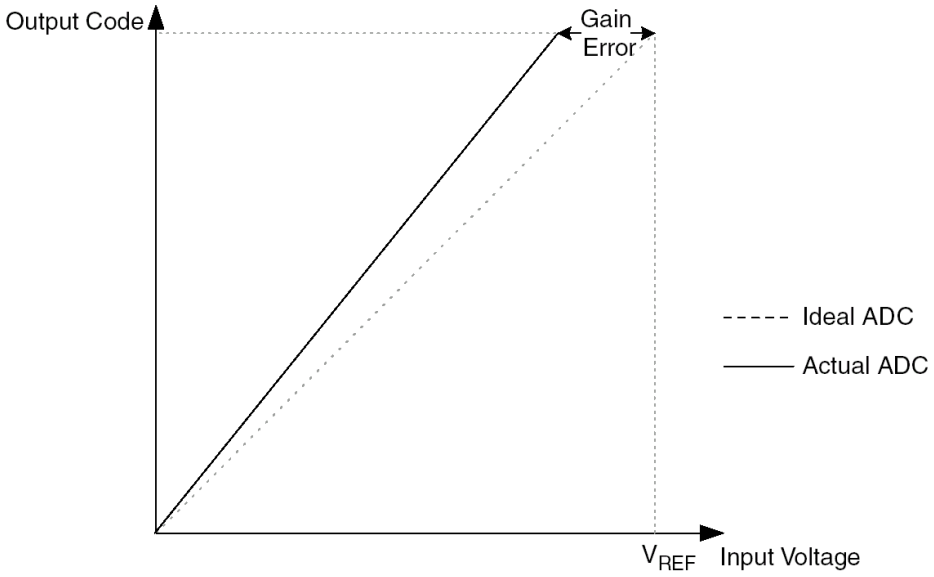
- Offset: błąd pierwszego przejścia(0x000 do 0x001) w porównaniu z idealnym przejściem. Idealna wartość: 0 LSB

Rys. 114. Błąd „offset”



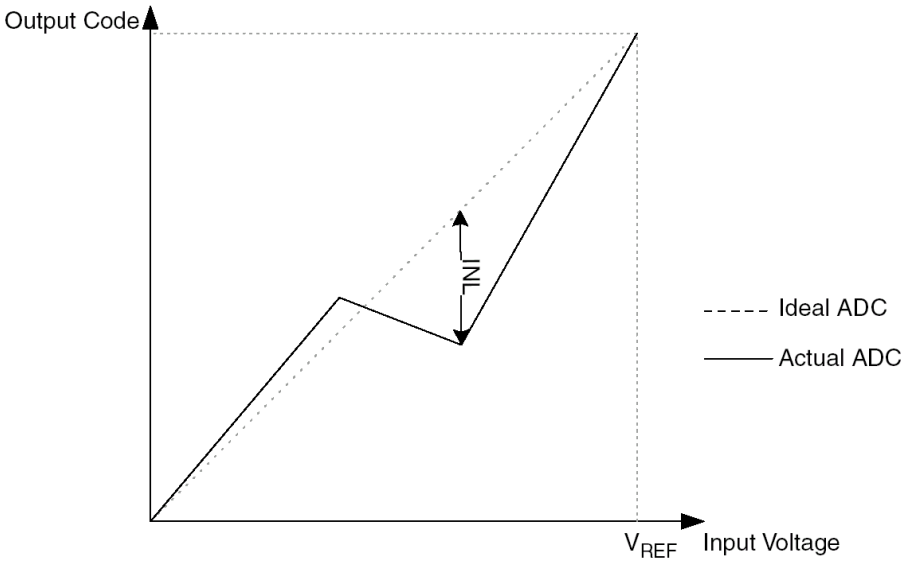
- Błąd ”Gain Error” odchylenie miedzy ostatnim przejściem (0x3FE to 0x3FF) a idealnym przejściem(1,5 LSB poniżej maksimum). Wartość idealna: 0 LSB.

Rys .115.

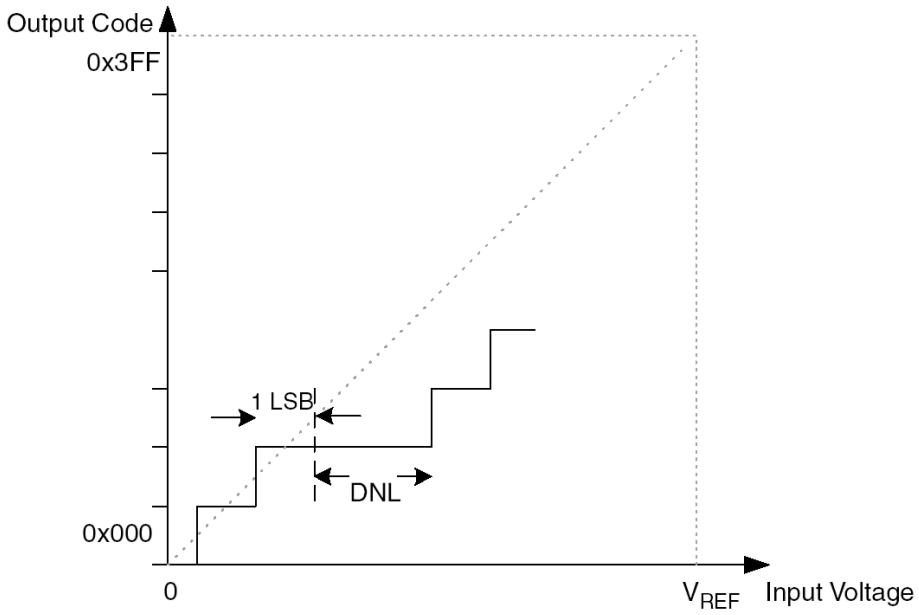


- Błąd nieliniowości całkowej: po wyrównaniu błędów “offset” i “gain error” błąd nieliniowości jest to maksymalne odchylenie między wartością rzeczywistej konwersji a konwersją idealną. Idealna wartość 0 LSB.

Rys. 116.



- Błąd nieliniowości różniczkowej: maksymalne odchylenie między rzeczywistą szerokością kodu (odstęp między dwoma sąsiednimi przejściami) a idealną szerokością kodu (1 LSB)
  Idealna wartość 0 LSB
  Rys .117.



- Błąd kwantyzacji : przy przekształcaniu napięcia wejściowego do kodu wyjściowego pewien zakres napięcia wejściowego (zawsze ± 0,5 LSB) jest kodowany ą samą wartością.
- Bezwzględna dokładność: maksymalne odchylenie rzeczywistego i idealnego przejścia. Na to odchylenie składają się: „offset error", „gain error", nieliniowość oraz błąd kwantyzacji. Wartość idealna ±0.5 LSB.

WYNIK KONWERSJI

Po zakończeniu konwersji wynik znajduje się w rejestrach ADCL I ADCH. Dla konwersji niesymetrycznej wynik wynosi:

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

gdzie:
V_{IN} - napięcie na wybranym wyprowadzeniu wejściowym
V_{REF} - napięcie odniesienia
Wynik 0x000 odpowiada analogowej masie, 0x3FF reprezentuje napięcie odniesienia minus jeden LSB.

Dla konwersji różnicowej wyniki jest równy

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot WZM \cdot 512}{V_{REF}}$$

gdzie:

V_{POS} – napięcie na dodatnim wejściu różnicowym
V_{NEG} – napięcie na ujemnym wejściu różnicowym
V_{REF} - napięcie odniesienia
WZM – wzmocnienie stopnia wzmacniającego

Tabela 95. Związek między napięciem wejściowym a kodem wyjściowym

V_{ADCn}	Kod	Wartość dziesiętna kodu
$V_{ADCm} + V_{REF}/GAIN$	0x1FF	511
$V_{ADCm} + 0.999\ V_{REF}/GAIN$	0x1FF	511
$V_{ADCm} + 0.998\ V_{REF}/GAIN$	0x1FE	510
...	...	...
$V_{ADCm} + 0.001\ V_{REF}/GAIN$	0x001	1
V_{ADCm}	0x000	0
$V_{ADCm} - 0.001\ V_{REF}/GAIN$	0x3FF	-1
...	...	...
$V_{ADCm} - 0.999\ V_{REF}/GAIN$	0x201	-511
$V_{ADCm} - V_{REF}/GAIN$	0x200	-512

REJESTR ADMUX

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Zapis/odczyt	Z/O	Z/O	Z/O	Z/O	Z/O	Z/O	Z/O	Z/O
Wartość początkowa	0	0	0	0	0	0	0	0

Bity 7 I 6 – REFS1 I REFS0 (Reference Selection Bits)
Te bity określają napięcie odniesienia ADC. Jeśli wartość tych bitów zostanie zmieniona podczas konwersji nie przyniesie to efektu do zakończenia bieżącej konwersji.

Bit 5 – ADLAR (ADC Left Adjust Result)

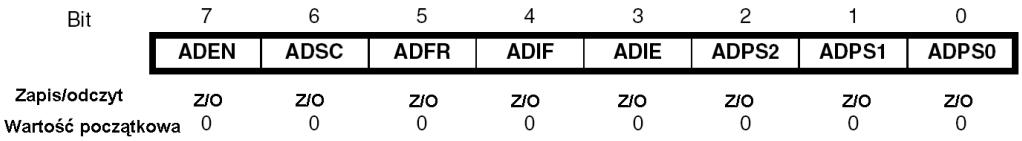
Wpisanie jedynki do tego bitu powoduje wyrównanie wyniku do lewej, w przeciwnym przypadku wynik jest wyrównany do prawej. Zmiana wartości tego bitu przynosi efekt natychmiast, niezależnie od bieżącej konwersji.

Bity 4 do 0 – MUX4 do0: (Analog Channel and Gain Selection Bits)
Wartości tych bitów określają jaka kombinacja wyprowadzeń wejściowych jest połączona z przetwornikiem analogowo cyfrowym; określa również wzmocnienie dla kanałów różnicowych. Zmiana wartości tych bitów nie przynosi efektu do czasu ukończenia bieżącej konwersji

Tabela 97. Wybór wejść ADC i wzmocnienia

MUX4..0	Wejście niesymetryczne	Wejście różnicowe dodatnie	Wejście różnicowe ujemne	Wzmoc - nienie
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	N/A	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1.22 V (V_{BG})	N/A		
11111	0 V (GND)			

REJESTR ADCSRA



Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Zapis/odczyt	Z/O	Z/O	Z/O	Z/O	Z/O	Z/O	Z/O	Z/O
Wartość początkowa	0	0	0	0	0	0	0	0

Bit 7 ADEN (ADC Enable)
Jeśli ADEN = 1 przetwornik analogowe cyfrowy jest włączony, jeśli ADEN = 0 ADC jest wyłączony. Jeśli ADC zostanie wyłączony podczas konwersji konwersja zostanie przerwana.

Bit 6 ADSC (ADC Start Conversion)
W trybie pojedynczej konwersji wpisanie jeden powoduje rozpoczęcie konwersji, w trybie „Free Running Mode" powoduje to rozpoczęcie pierwszej konwersji

Bit 5 – ADFR (ADC Free Running Select)
Jeśli ten bit = 1 ADC pracuje w trybie „Free Running Mode", w tym trybie ADC stale wykonuje próbkowanie i wpis do rejestrów danych. Wpisanie zera powoduje zakończenie pracy w trybie „Free Running Mode".

Bit jest równy jeden po zakończeniu konwersji i po zaktualizowaniu rejestrów danych. Jeśli ADIF = 1 i bit I w rejestrze SREG =1 występuje przerwanie zakończenia konwersji.

Bit 4 – ADIF (ADC Interrupt Flag)

Jeśli ten bit jest równy jeden przerwanie ukończenia konwersji jest aktywne

Bit 3 – ADIE (ADC Interrupt Enable)

Bity 2, 1, 0 –ADPS2, ADPS1, ADPS0: ADC Prescaler Select Bits
Bity ustalają czynnik dzielenia między częstotliwością XTAL a zegarem przetwornika analogowo cyfrowego

ADPS2	ADPS1	ADPS0	Czynnik dzielenia
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

REJESTRY DANYCH ADCL i ADCH

ADLAR=0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
Bit	7	6	5	4	3	2	1	0	
Zapis/odczyt	O	O	O	O	O	O	O	O	
	O	O	O	O	O	O	O	O	
Wartość początkowa	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
Bit	7	6	5	4	3	2	1	0	
Zapis/odczyt	O	O	O	O	O	O	O	O	
	O	O	O	O	O	O	O	O	
Wartość początkowa	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

REJESTR SFIOR

Bit	7	6	5	4	3	2	1	0
	TSM	–	–	ADHSM	ACME	PUD	PSR2	PSR10
Zapis/odczyt	Z/O	Z/O	Z/O	Z/O	Z/O	Z/O	Z/O	Z/O
Wartość początkowa	0	0	0	0	0	0	0	0

INTERFEJS JTAG I WYBUDOWANY SYSTEM DEBUGOWANIA

CECHY

- **Interfejs JTAG (IEEE standart 1149.1)**
- **Możliwości skanowania urządzeń peryferyjnych zgodnie ze standartem IEE 1149.1**
- **Dostęp debugera do:**
  - **Wszystkich peryferyjnych jednostek**
  - **Wewnętrznego i zewnętrznego RAM**
  - **Wewnętrznego zestawu rejestrów**
  - **Licznika programu**
  - **Pamięci EEPROM i Flash**
- **Obszerne wbudowane wsparcie debugera dla łamania warunków, zawierające**
  - **Przerywanie instrukcji AVR**
  - **Przerwanie przy zmianie przepływu pamięci programu**
  - **Pojedynczy krok przerywający**
  - **Breakpointy w pamięci programu na poszczególnych adresach lub zakresie adresów**
  - **Breakpoint na pamięci danych na pojedynczym adresie lub zakresie adresów**
- **Programowanie Flash, EEPROM, bezpieczników i bitów blokujących przez interfejs JTAG**
- **Wbudowany debuger wspierany przez AVR studio**

- Krótki opis

AVR standard IEEE 1149.1 podatny interfejsowi JTAG może być wykorzystywany do:

- **Testowania PCB przy użyciu zdolności JTAG do skanowania urządzeń peryferyjnych**
- **Programowania stałych pamięci, bezpieczników i bitów blokujących**
- **Debugowania wewnętrznego**

Krótki opis jest przedstawiony w następujących sekcjach. Szczegółowy opis programowania przez interfejs JTAG i wykorzystanie łańcucha skanowania urządzeń zewnętrznych znajduje się w sekcji „Programowanie przez interfejs JTAG" na stronie 301 i „IEEE 1149.1 (JTAG) skanowanie urządzeń peryferyjnych" na stronie 248. Wbudowane wsparcie debugera jest oparte na prywatnych instrukcjami JTAG i rozprowadzany przez ATMEL i wybranych sprzedawców.

Rysunek 119 przestawia diagram blokowy interfejsu JTAG z systemem degugera. Kontroler TAP jest urządzeniem kontrolowanym przez sygnały TCK i TMS. Kontroler TAP wybiera albo rejestr instrukcji JTAG albo jeden z wielu rejestrów danych jako łańcuch skanowania (rejestr przesuwny) pomiędzy wejściem TDI i wyjściem TDO.

Rejestr instrukcji przetrzymuje rozkaz JTAG kontrolujący zachowanie rejestru danych.

Rejestr ID, rejestr bocznikujący i łańcuch skanowania urządzeń peryferyjnych są rejestrami danych wykorzystywanymi do poziomowego testowania. Interfejs programowania JTAG (składający się z wielu fizycznych i wirtualnych rejestrów danych) jest wykorzystywany do szeregowego programowania poprzez interfejs JTAG. Wewnętrzny łańcuch skanujący i łańcuch skanujący breakpoint są wykorzystywane tylko przez wbudowany debuger.

- Dostęp do portu testującego TAP

Interfejs JTAG jest dostępny przez cztery nóżki AVR. W terminologii JTAG, nóżki te są częścią składową portu testowego dostępu – TAP i są to:

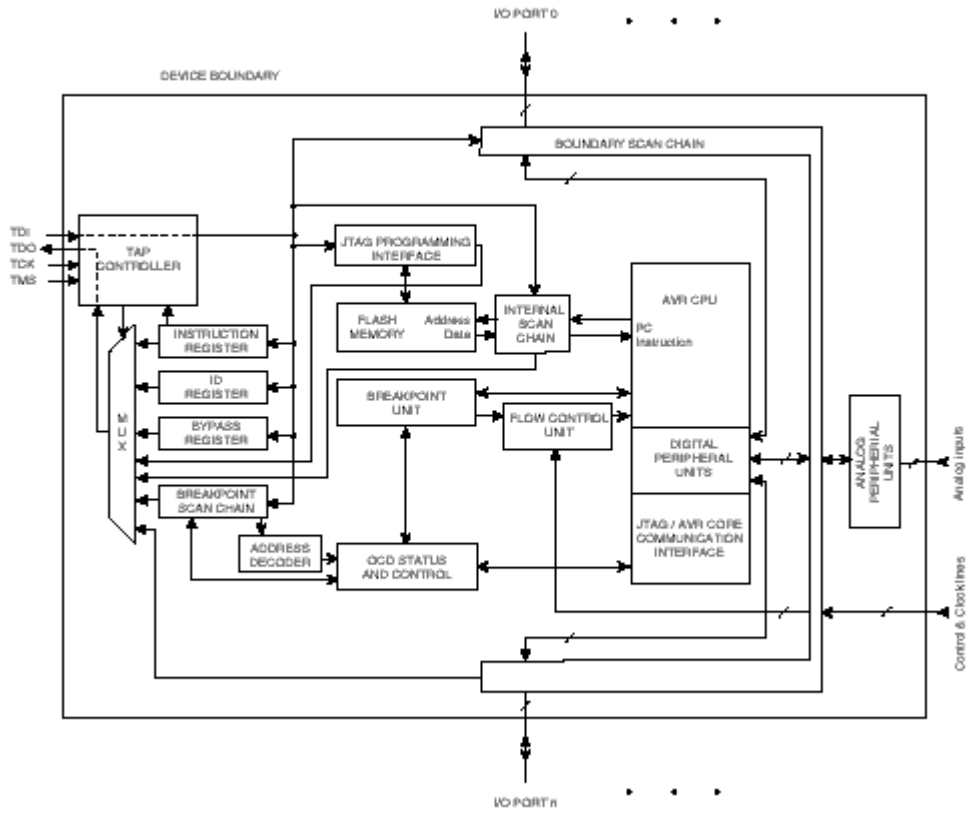
- **TMS: wybór trybu testowania. Służy do nawigacji po stanach pracy maszyny kontrolera TAP**
- **TCK: test zegara. Operacje JTAG są synchroniczne do TCK.**
- **TDI: testowanie danych wejściowych. Szeregowe wejście danych przesuwane jest do rejestru instrukcji lub rejestru danych (łańcuch skanowania)**
- **TDO: test danych wychodzących. Szeregowe wyjście z rejestru instrukcji lub rejestru danych/**

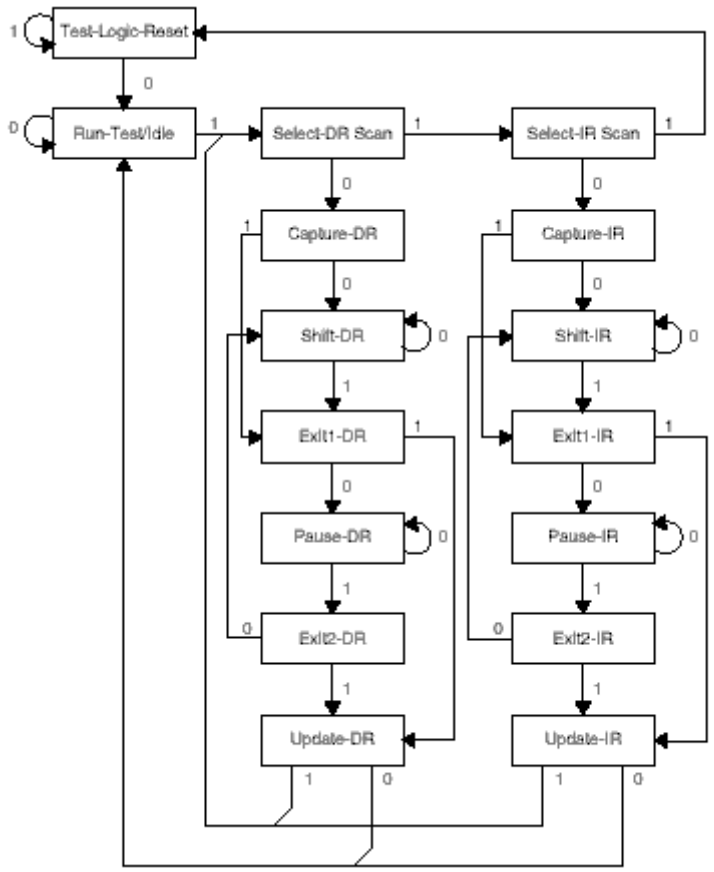
IEEE standard 1149.1 również specyfikuje opcjonalne sygnały TAP; TRST – reset testu, który nie jest przewidziany.

Kiedy bezpiecznik JTAGEN nie jest zaprogramowany, te cztery nóżki są normalnymi wyjściami portu i TAP kontroler jest zresetowany. Kiedy bezpiecznik ten jest zaprogramowany, sygnały wejściowe TAP są wewnętrznie ustawiane w stan wysoki i JTAG jest odblokowany dla skanowanie urządzeń zewnętrznych i programowania. Urządzenie jest ładowane zaprogramowanymi bezpiecznikami.

Dla wbudowanego systemu debugowania, w dodatku do nóżek interfejsu JTAG, nóżka RESET jest monitorowana przez debuger w celu wykrycia zewnętrznego resetu źródeł. Debuger może również ustawić reset w stan niski w celu zresetowania całego systemu, zakładając tylko, że linia resetu jest OC (open collector).

Figure 119. Block Diagram





- Kontroler TAP
  Kontroler TAP jest 16 stanową skończoną maszyną, która kontroluje operacje obwodu skanowania urządzeń peryferyjnych, obwodu programowania JTAG lub wbudowanego systemu debugera. Przemiana stanów opisanych na rysunku 120 zależy od sygnałów obecnych na TMS (pokazanych przy każdej przemianie stanu) w czasie narastającego zbocza TCK. Początkowy stan po resecie przy włączonej mocy jest reset testu logicznego.

  Jako definicja w tym dokumencie LSB jest przesuwany do i z pierwszego z wszystkich rejestrów przesuwnych.

  Biorąc pod uwagę, że tryb działania-testowania/idyl jest bieżącym stanem, typowym sposobem postępowania z interfejsem JTAG jest był:
  - **Na wejście TMS należy podać sekwencję 1, 1, 0, 0 na narastającym zboczu TCK aby wejść w tryb przesuwania rejestru instrukcji – Shift-IR. W tym stanie przesuń 4 bity instrukcji JTAG do rejestru instrukcji JTAG z wejścia TDI na narastającym zboczu TCK. Wejście TMS należy utrzymywać w stanie niskim podczas wejścia trzech najmłodszych bitów w celu pozostania w stanie Shift-IR. Najbardziej znaczący bit instrukcji jest przesuwany kiedy ten stan zostaje opuszczony poprzez ustawienie TMS na stan wysoki. Kiedy instrukcja jest wsuwana z wejścia TDI, złapany stan IR 0x01 jest wysuwany na wyjściu TDO.**

Rozkaz JTAG wybiera konkretny rejestr danych jako ścieżkę między TDI i TDO i kontroluje obwody otaczające w wybranych rejestrach danych.

- **Zastosuj sekwencje 1, 1, 0 na TMS aby powtórnie wejść w tryb działania-testowania/idyl. Rozkaz jest złapany na równoległym wyjściu ze ścieżki rejestru przesuwnego w stanie Update-IR. Stany Exit-IR, Pause-IR, Exit2-IR są wykorzystywane tylko dla nawigacji maszyny statycznej.**
- **Na wejście TMS podaj sekwencje 1, 0, 0 na narastającym zboczu TCK aby wejść w stan przesuwnego rejestru danych – Shift-DR. W tym stanie załaduj wybrany rejestr danych(wybrany przez obecną instrukcje w rejestrze instrukcji) z wejścia TDI przy narastającym zboczu TCK. W celu pozostania w tym stanie, wejście TMS musi być w stanie niskim podczas wsuwania wszystkich bitów z wyjątkiem MSB. MSB danych jest wsunięte kiedy ten stan jest opuszczony przez ustawienie wysokiego stanu na wejściu TMS. Podczas gdy rejestr danych jest wsuwany z wejścia TDI, równoległe wejścia do rejestrów danych łapane są w stanie Capture-DR i przesuwane na wyjście TDO.**
- **Podaj sekwencje 1, 1, 0 na TMS aby wejść powtórnie do stanu działania-testownia/idyl. Jeśli wybrany rejestr danych będzie miał zatrzaśnięte równoległe wyjścia, zatrzaskiwanie ma miejsce w stanie Update-DR. Stany Exit-DR, Pause-DR, Exit2-DR są wykorzystywane tylko dla nawigacji maszyny statycznej.**

Jak pokazano w diagramie stanów, do stanu dziłania-testowania/idyl nie trzeba przechodzić między wyborem rozkazu JTAG i wykorzystaniem rejestru danych, niektóre rozkazy mogą wybrać konkretne funkcje, które zostaną wykonane w trybie działania-testowania/idyl, powodując, że jest to niewygodny stan jako idyl.
Uwaga: Niezależnie od stanu początkowego do kontrolera TAP, można zawsze wejść poprzez utrzymanie stanu wysokiego na TMS przez 5 cykli TCK.

Szczegółowe informacje w specyfikacji JTAG „Bibliografia" na stronie 247.

- Wykorzystanie łańcucha skanującego peryferyjne urządzenia

Pełny opis możliwości łańcucha skanującego urządzenia zewnętrzne jest dany w sekcji „ IEEE 1149.1 (JTAG) skanowanie urządzeń peryferyjnych" na stronie 248.

- Wykorzystanie wewnętrznego systemu debugera

Jak przedstawia rysunek 119 sprzętowe wsparcie dla wbudowanego debugera składa się z:

- **Łańcucha skanującego na interfejsie pomiędzy wewnęrznym AVR CPU i wewnętrznymi obrzeżnymi jednostkami.**
- **Jednostki break point**
- **Interfejs komunikacyjny między CPU i systemem JTAG.**

Wszystkie operacje odczytu lub zapisu/ modyfikowania potrzebne do zaimplementowania debugera są zapewnione poprzez odwoływanie się do instrukcji AVR poprzez wewnętrzny łańcuch skanowania AVR CPU. CPU wysyła rezultaty do

odwzorowanych lokacji pamięci I/O , które są częścią interfejsu komunikacyjnego pomiędzy CPU i systemem JTAG.

Jednostka break point implementuje przerwanie działania programu przy zmianie, pojedynczy krok, dwa programowe breakpointy pamięci, i dwa łączone breakpointy. Razem, te cztery break pointy mogą zostać skonfigurowane następująco:

- **4 pojedyncze break pointy pamięci programu**
- **3 pojedyncze break pointy pamięci programu i 1 pojedynczy break point pamięci danych**
- **2 pojedyncze break pointy pamięci programu i 2 pojedynczy break point pamięci danych**
- **2 pojedyncze break pointy pamięci programu i 1 break point pamięci programu z maską (break point zakresu)**
- **2 pojedyncze break pointy pamięci programu i 1 break point pamięci danych z maską (break point zakresu)**

Debuger tak jak i AVR studio mogą używać jedno lub więcej z tych źródeł dla wewnętrznych potrzeb, pozostawiając mniej możliwości końcowemu użytkownikowi.

Lista specyficznyhc rozkazów JTAG jest podana w „ Specyficzne rozkazy JTAG ...” na stronie 246.

Bezpiecznik JTAGEN musi być zaprogramowany aby odblokować port dostępu do testowania JTAG. W dodatku, bezpiecznik OCDEN musi być zaprogramowany i nie mogą być ustawione żadne bity blokujące aby system wbudowanego debugera mógł działać. Jako cech bezpieczeństwa, system wbudowanego debugera jest zablokowany kiedy jakikolwiek bit bezpieczeństwa jest ustawiony. W przeciwnym wypadku, system wbudowanego debugera byłby tylnymi drzzwiami umożliwiającymi obejście zabezpieczeń.

Studio AVR pozwala użytkownikowi w pełni kontrolować wykonanie programów na urządzeniu AVR ze wszystkimi możliwościami wbudowanego debugera, Emulatora wejściowego obwodu AVR lub wbudowany symulator zestawu rozkazów AVR. Studio AVR wspiera poziom źródła wykonania programów asemblerowych.

Studio AVR działa w systemie Mictosoft Windows 95/98/2000 i Windows NT.

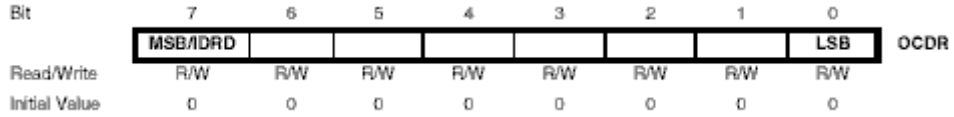
Pełany opis studia AVR znajduje się w przewodniku po studiu AVR.
Wszystkie potrzebne do wykonania rozkazy są dostępne w AVR studio, zarówno na poziomie źródeł jak i poziomie zdeasemblowanym. Użytkownik może wykonać program pojedynczymi krokami lub wykonując oddzielnie poszczególne funkcje i inne możliwość dostępne w pospolitym debugerze. W dodatku użytkownik może mieć nielimitowaną liczbę break pointów i do dwóch break pointów pamięci, alternatywnie połączoneych jako maskowalny break point.

- Wewnętrzne specyficzne instrukcje debugera JTAG

Wsparcie systemu debugera ma prywatne instrukcji JTAG sprzedawane przez ATMELa i wybranych sprzedawców. Rozkazy są następujące:

PRIVATE0; $8 **prywatny rozkaz JTAG dla dostępu do systemu debugowania.**
PRIVATE1; $9 **prywatny rozkaz JTAG dla dostępu do systemu debugowania.**
PRIVATE2; $A **prywatny rozkaz JTAG dla dostępu do systemu debugowania.**
PRIVATE3; $B **prywatny rozkaz JTAG dla dostępu do systemu debugowania.**

- Wewnętrzny związany rejestr debugera w pamięci I/O
    - Wbudowany rejestr debugera – OCDR



Rejestr OCDR zapewnia kanał komunikacyjny z działającego programu w mikrokontrolerze z debugerem. CPU może wysłać bajt do debugera poprzez wpisanie do tej lokacji. W tym samym czasie wewnętrzna flaga; I/O rejestr debugera – IDRD – jest ustawiony tak, że wskazuje debugerowi, że rejestr został zapisany. Kiedy CPU odczytuje rejestr OCDR 7 LBS będzie pochodził z tego rejestru, podczas gdy MSB jest bitem IDRD. Debuger zeruje bit IDRD po przeczytaniu informacji.

W niektórych urządzeniach AVR rejestr ten jest dzielony ze standardową lokacją I/O. W tym przypadku, rejestr OCDR jest dostępny tylko jeśli bezpiecznik OCDEN jest zaprogramowany i debuger odblokuje dostęp do rejestru OCDR. We wszystkich innych przypadkach standardowe lokacje I/O są dostępne.

W celu dokładniejszych informacji należy przejrzeć dokumentacje debugera.

- Wykorzystanie zdolności do programowania JTAG

Programowanie części AVR poprzez JTAG jest wykonywane przy wykorzystaniu czterech nóżek portu JTAG: TCK, TMS, TDI i TDO. To są jedyne nóżki, które muszą być kontrolowane przy programowaniu JTAG. Nie jest wymagane by podać 12V z zewnątrz. Bezpiecznik JTAGEN musi być zaprogramowany i bit JTD w rejestrze MCUSR musi być wyzerowany aby odblokować port dostępu do testowania JTAG.

Programowanie JTAG wspiera:
- **Programowanie i weryfikacje Flash.**
- **Programowanie i weryfikacje EEPROM**
- **Programowanie i weryfikacje bezpieczników**
- **Programowanie i weryfikacje bitów blokujących**

Zabezpieczenie bitów blokujących jest takie samo jak w trybie programowania równoległego. Jeżeli bity blokujące LB1 lub LB2 są zaprogramowane bezpiecznik

OCDEN nie może być zaprogramowany, chyba że najpierw wykona się wymazanie modułu. Jest to cech mająca na celu zabezpieczenie przed umożliwieniem tylnego wejścia dla odczytu danych.

Szczegóły opisane są w sekcji „Programowanie przez interfejs JTAG" na stronie 301.

- Bibliografia

W celu uzyskania większej ilości informacji na temat łańcuch skanowania urządzeń peryferyjnych polecamy:

- IEEE: IEEE Std 1149.1-1990. IEEE Standard Test Access Port and Boundary-scan Architecture, IEEE, 1993
- Colin Maunder: The Board Designers Guide to Testable Logic Circuits, Addison-Wesley, 1992

- IEEE 1149.1 (JTAG) skanowanie urządzeń peryferyjnych
    - Cechy
        - **Interfejs JTAG (IEE 1149.1)**
        - **Możliwość skanowania łańcucha urządzeń peryferyjnych zgodnie ze standartami JTAG**
        - **Pełne skanowanie wszystkich funkcji portów jak również obiektu analogowego**
        - **Wsparcie opcjonalnych rozkazów IDCODE**
        - **Dodatkowe publiczne rozkazy AVR_RESET służące resetowaniu AVR.**

- Krótki opis systemu

Łańcuch skanowania urządzeń zewnętrznych ma zdolności sterowania i obserwowania logicznych poziomów na binarnych wejściach I/O jak również granice pomiędzy binarnym i analogowym poziomem logicznym dla obwodów mających połączenie odległe z modułem. Na poziomie systemu wszystkie ICs, które mają możliwości JTAG są podłączone szeregowo poprzez sygnały TDO/TDI w celu stworzenia długiego rejestru przesuwnego. Zewnętrzny kontroler ustawia urządzenia aby wyprowadzały wartości na nóżki wyjściowe i obserwuje wartości wejściowe otrzymane z innych urządzeń. Kontroler porównuje otrzymane dane z oczekiwanymi wynikami. W ten sposób łańcuch skanowania urządzeń peryferyjnych zapewnia mechanizm dla testowania połączeń i integralności składników na drukowanych płytkach poprzez wykorzystanie tylko czterech sygnałów TAP.

Cztery zdefiniowane instrukcje obowiązujące w JTAG: IDCODE, BYPASS< SAMPLE/PRELOAD i EXTEST jak również specyficzne publiczne rozkazy JTAG – AVR_RESET mogą być wykorzystane do testowania płytek drukowanych. Początkowe skanowanie ścieżek rejestrów danych pokaże kod identyfikacyjny urządzenia, odkąd IDCODE jest domyślną instrukcją JTAG. Jeżeli urządzenie jest zresetowane podczas

trybu testowego może dać to pożądane skutki. Jeżeli nie jest zresetowane wejścia do urządzenia mogą być zdeterminowane przez operacje skanowania i wewnętrzne oprogramowania może znaleźć się w nie zamierzonym stanie przy wykonywaniu trybu testowania. Wchodząc w reset, na wyjścia portu nóżek zostaną natychmiastowo podane wysokie stany napięć sprawiając, że rozkaz HIGHZ będzie zbędny. Jeżeli potrzeba instrukcja BYPASS może spowodować stworzenie najkrótszego możliwego łańcucha skanującego całe urządzenie. Urządzenie może zostać ustawione w stan resetu albo poprzez podanie stanu niskiego na wejście RESET albo przez wykonanie rozkazu AVR_RESET z odpowiednim ustawieniem danych w rejestrze resetu danych.

Rozkaz EXTEST jest wykorzystywany do próbkowania zewnętrznych nóżek i ładowania wyjść danymi. Dane zatrzaśnięte na wyjściu zostaną wyprowadzone na wyjścia jak tylko rozkaz EXTEST zostanie wprowadzona do rejestru-IR. Dlatego SAMLPLE/PRELOAD również powinny być wykorzystywane do początkowego ustawiania wartości w pierścieniu skanowania w celu uniknięcia szkodliwego wpływu na płytkę przy pierwszym wykonywaniu instrukcji EXTEST. SAMPLE/PRELOAD może być również wykorzystywana do robienia migawek wyjść zewnętrznych podczas normalnych operacji na tej części urządzenia.

Bezpiecznik JTAGEN musi być zaprogramowany i bity JTD w rejestrze I/O MCUCSR muszą być wyzerowane, aby odblokować port dostępu do testowania JTAG.

Korzystając z interfejsu JTAG przy skanowaniu urządzeń peryferyjnych wykorzystywanie zegara TCK o większej częstotliwości niż wewnętrzna częstotliwość układu jest możliwe. Zegar wewnętrzny nie musi pracować.

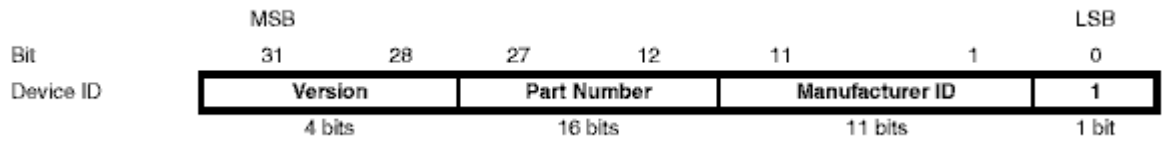
- Rejestry danych

Rejestry danych odpowiadające odpowiednim operacjom skanowania urządzeń zewnętrznych:
  - **Rejestr bocznikujący**
  - **Rejestr identyfikacji urządzenia**
  - **Rejestr resetu**
  - **Łańcuch skanowania urządzeń peryferyjnych**

  - Rejestr bocznikujący

Rejestr bocznikujący składa się z pojedynczego poziomu rejestru przesuwnego. Kiedy rejestr ten jest wybrany jako ścieżka między TDI i TDO jest on resetowany na 0 przy opuszczaniu stanu kontroli Capture-DR. Rejestr bocznikujący może być wykorzystywany do skracania łańcucha skanowania kiedy mają być testowane inne urządzenia.

  - Rejestr identyfikacji urządzenia



- **Wersja**

Wersja jest to cztero bitowy numer identyfikujący weryfikujący część składową. Odpowiadające numery są podane w tabeli 99.

Tabele 99

Wersja	Numer seryjny JTAG (heksadecymalnie)
ATmega128 nowelizacja C	0x3
ATmega128 nowelizacja F	0x5
ATmega128 nowelizacja G	0x6

- **Numer części**

Numer części jest szesnastobitowym kodem identyfikującym część składową. Numery JTAG dla części składowych są zamieszczone w tabeli 100.

Numer części	Numer części JTAG
ATmega128	0x9702

- **ID wykonawcy**

Id wykonawcy jest 11-bitowym kodem identyfikującym producenta. Tabela 101.

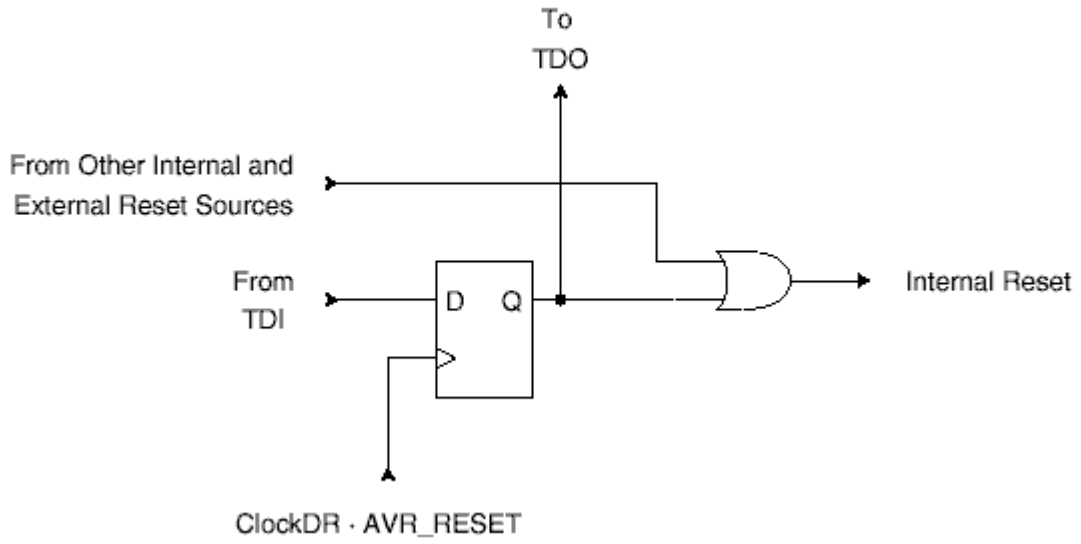
Producent	Producent JTAG
Atmel	0x01F

- Rejestr resetu

Rejestr resetu jest rejestrem testowania danych wykorzystywanym do resetowania części . Odkąd AVR określa nóżki portu przy resecie rejestr resetu może również podmieniać funkcje nie zaimplementowanych opcjonalnych rozkazów HIHGZ JTAG.

Wysoka wartość w rejestrze resetu odpowiada ustawieniu zewnętrznego resetu na stan niski. Część jest resetowana tak długo jak w rejestrze resteu znajduje się wysoka wartość. W zależności od ustawień bezpieczników dla funkcji zegara, cześć ta pozostanie zresetowana dla resetu okresu czasu przerwy („Źródła zegara” na stronie 34) po zwolnieniu rejestru resetu. Wyjście z rejestru danych nie jest zatrzaskiem, więc reset nastąpi natychmiast co pokazuje rysunek 122.

Figure 122. Reset Register



- Łańcuch skanowania urządzeń peryferyjnych

Łańcuch skanowania urządzeń peryferyjnych ma możliwość wprowadzania i obserwowania poziomów logicznych na binarnych wejściach I/O, jak również granic między binarnymi i analogowymi locznymi poziomami dla analogowego obiegu mającego zewnętrzne połączenia.

Przejrzyj opsi „Łańcuch skanowania urządzeń peryferyjnych „ na stronie 252.

- Specyficzne instrukcje JTAG skanujące urządzenia peryferyjne

Rejestr instrukcji jest czterobitowy dlatego wspiera 16 rozkazów Poniżej zostały wypisane rozkazy JTAG przydatne przy operacjach skanowania. Warto zauważyć, że opcjonalny rozkaz HIGHZ nie jest zaimplementowany, ale wszystkie wyjścia trzystanowe mogą być ustawiane w stan wysokiej impedancji poprzez wykorzystanie rozkazu AVR_RESET, odkąd początkowym stanem dla wszystkich portów jest trzeci stan.

Jako definicje w tym arkuszu danych przyjmuje się, że LSB jest wsuwany i wysuwany jako pierwszy we wszystkich rejestrach przesuwnych.

Kod rozkazu jest pokazany za każdą nazwą rozkazu w formacie szesnastkowym. Tekst opisuje, który rejestr danych jest wybrany jako ścieżka między TDI i TDO dla każdej instrukcji.

- EXTEST; $0

Obowiązkowe instrukcje dla wyboru łańcucha skanowania urządzeń peryferyjnych jako rejestru danych dla testowania zewnętrznych obwodów dla pakietów AVR. Dla

nóżek portów, Pull-up Disable, Output Control, Output Data i Input Date wszystkie są dostępne w łańcuchu skanowania. Dla analogowych obwodów mających zewnętrzne połączenia interfejs między analogową a binarną logiką jest w łańcuchu skanowania. Zawartość zatrzaśniętych wyjść łańcuch skanowania urządzeń peryferyjnych jest wyprowadzana jak tylko rejestr IR jest załadowany rozkazem EXTEST.

Aktywne stany to:

- **Capture-DR: dane na zewnętrznych nóżkach są próbkowane przez łańcuch skanowania urządzeń peryferyjnych**
- **Shift-DR: wewnętrzny łańcuch skanowania jest przesuwany przez wejście TCK**
- **Update-DR: dane z łańcucha skanowania jest podana na wyjścia**

    - IDCODE ; $1

Opcjonalne rozkazy JTAG wybierające 32 bitowy rejestr ID jako rejestr danych. Rejestr ID składa się z numeru wersji, numeru urządzenia i kodu producenta, który jest wybrany przez JEDEC. Jest to domyślna instrukcja po włączeniu zasilania.

Aktywne stany to:

- **Capture-DR: dane w rejestrze IDCODE są próbkowane w łańcuchu skanowania**
- **Shift-DR: łańcuch skanowania jest przesuwany przez wejście TCK**

    - SAMPLE_PRELOAD; $2

Obowiązującym rozkazem JTAG służącym do przeładowania zatrzasków na wyjściach I pobrania migawek nóżek I/O bez wpływania na działanie systemu. Jednakże, zatrzaśnięte wyjścia nie są podłączone do nóżek. Łańcuch skanowania jest wybierany jako rejestr danych.

Aktywne stany to:

- **Capture-DR: dane na zewnętrznych nóżkach są próbkowane przez łańcuch skanowania**
- **Shift-DR: wewnętrzny łańcuch skanowania jest przesuwany przez wejście TCK**
- **Update-DR: dane z łańcucha skanowania jest podana na wyjścia. Chociaż zatrzaski wyjść nie są połącznoe z nóżkami.**

    - AVR_RESET; $C

Rozkaz JTAG dla wprowadzania urządzenia AVR w tryb resetu lub zwalniający źródła resetu JTAG. Kontroler TAP nie jest resetowany przez ten rozkaz. Jeden bit w rejestrze resetu jest wybierany jako rejestr danych. Zauważ, że reset będzie aktywny tak długo jak jest logiczna jedynka w łańcuchu resetu.

Aktywne stany:

- **Shift-DR: rejestr resetu jest wsuwany przez wejście TCK.**

    - BYPASS; $F

Obowiązujący rozkaz JTAG wybierający rejestr bocznikujący dla rejestru danych.

Aktywne stany to:

- **Capture-DR: ładuje logiczne 0 do rejestru bocznikującego**
- **Shift-DR: komórka rejestru bocznikującego znajdujące się między TDI i TDO jest przesuwana.**

- Rejestr spokrewniony ze skanowaniem urządzeń peryferyjnych w pamięci I/O
  - Rejestr kontroli i stanu MCU – MCUCSR

Rejestr kontroli i stanu MCU zawiera bity kontrolne dla powszechnych funkcji MCU i zapewnia informacje, które źródło resetu spowodowało reset MCU.

Bit	7	6	5	4	3	2	1	0	
	JTD	–	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

a: bit 7- JDT: zablokowanie interfejsu JTAG

kiedy bit ten jest zerem, interfejs JTAG jest odblokowany przy zaprogramowanym bezpieczniku JTAGEN. Jeżeli bit ten jest jedynką interfejs JTAG jest zablokowany. W celu uniknięcia nie zamierzonego blokowania lub odblokowywania interfejsu JTAG sekwencja czasowa musi być wykonana przy zmianie tego bitu. Oprogramowanie musi wpisać do tego bitu zamierzoną wartość dwa razy w przeciągu czterech cykli maszynowych, aby zmienić wartość tego bitu.

b: bit 4 – JTRF: flaga resetu JTAG

Flaga ta jest ustawiana jeśli reset został spowodowany poprzez logiczną jedynkę w rejestrze resetu wybraną poprze instrukcję AVR_RESET. Bit ten jest reseetowany przy resecie uruchamiania mocy lub poprzez wpisanie zera do tej flagi.

- Łańcuch skanowania urządzeń zewnętrznych

Łańcuch skanowania urządzeń peryferyjnych ma możliwość wprowadzania i obserwowania poziomów logicznych na binarnych nóżkach I/O, jak również granic między binarnymi i analogowymi stanami logicznymi obwodów mających zewnętrzne połączenia.

  - Skanowanie cyfrowych nóżek portów

Rysunek 123 przedstawia łańcuch skanowania urządzeń peryferyjnych dla dwukierunkowych nóżek portów z funkcjami podciągającymi. Komórka taka składa się ze standartowej komórki skanowania dla odblokowanej funkcji podciągania – PUExn – i dwukierunkową komórkę nóżek, które wytwarzają trzy sygnały: Output Control –

Ocnx, Output Data – Odxn i Input Data – Idxn tylko w dwupoziomowym rejestrze przesuwnym. Port i indeksy nóżek nie są wykorzystywane w poniższym opisie.

Łańcuch skanowania poziomów logicznych nie jest zamieszczony na rysunku. Rysunek 124 pokazuje przykładowy binarny port nóżek co opisano w sekcji „porty I/O" na stronie 60. Szczegóły łańcucha skanowania z rysunku 123 podmienia rysowany przerywaną linią prostokąt z rysunku 124.

Jeśli żadna alternatywna funkcja portu nie jest obecna, wejściowe dane – ID odpowiadają wartościom rejestru PINxn (ale ID nie ma żadnego synchronizatora), wyjściowe dane odpowiadają rejestrowi PORT, wyjście kontrolne odpowiada rejestrowi kierunku danych DD i odblokowanie podciągania – PUExn – odpowiada logficznym wyrażeniom $\overline{PUD} \bullet \overline{DDxn} \bullet PORTxn$ **.**
Binarne alternatywne funkcje portów są połączone na zewnątrz prostokąta rysowanego przerywaną linią na rysunku 124 w celu odczytania przez łańcuch skanowania aktualnych wartości nóżek. Dla funkcji analogowych jest bezpośrednie połączenie z zewnętrznych nóżek do analogowego obiegu i łańcuch skanowania jest włączony w interfejs między binarną logiką a analogowym obwodem.

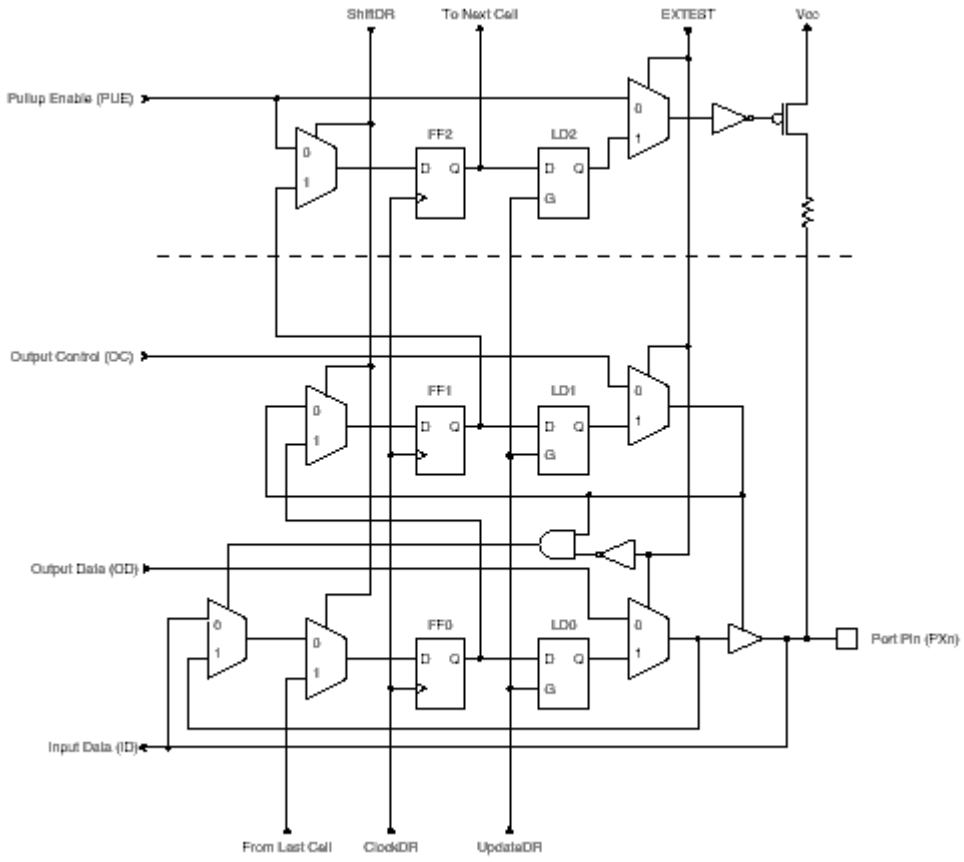
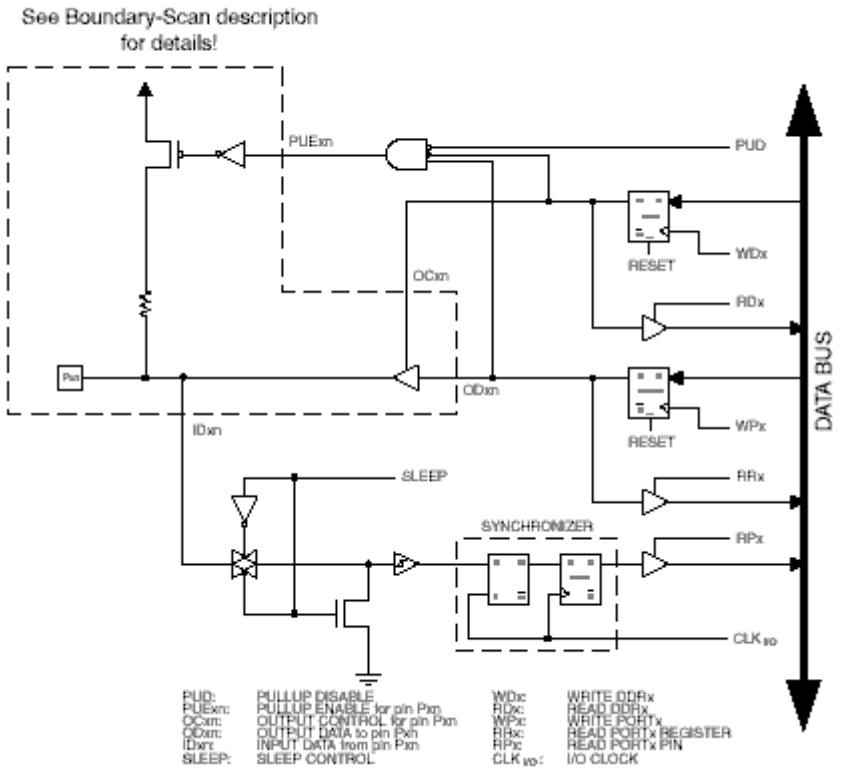


Figure 124. General Port Pin Schematic diagram



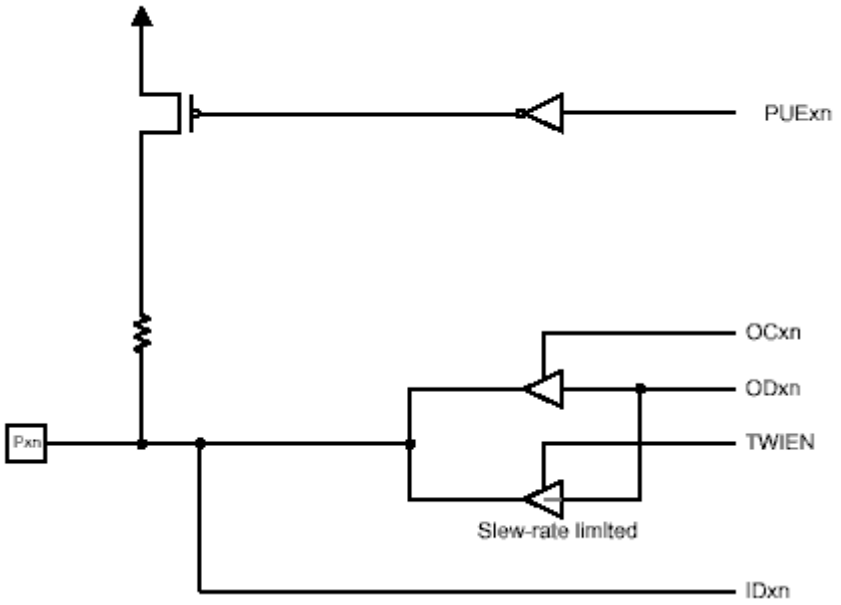
- Skanowanie urządzeń zewnętrznych i dwu przewodowy interfejs

Nóżki interfejsu dwu przewodowego SCL i SDA mają dodatkowy sygnał kontrolny w łańcuchu skanowania – odblokowanie dwu przewodowego interfejsu – TWIEN. Jak pokazuje rysunek 125 sygnał TWIEN odblokowuje bufor trzy stanowy z szybkością reakcji kontrolowaną równolegle ze zwykłymi binarnymi nóżkami portów. Główna komórka skanowania jest przedstawiona na rysunku 129 ma dołączony sygnał TWIEN.

Uwagi: 1. Oddzielny łańcuch skanowania dla 50 ns filtru krótkiego impulsu nie jest zapewniony. Zwykłe wsparcie dla binarnych nóżek portów wystarcza do TRESTÓW POŁĄCZEŃ. Jedynym powodem posiadania TWIEN w ścieżce skanowania jest możliwość rozłączenia buforu kontroli filtru krótkiego impulsu przy skanowaniu urządzeń peryferyjnych.

2. Upewnij się, czy sygnały OC i TWIEN nie są zapewnione jednocześnie bo spowoduje to spór wyprowadzeń.

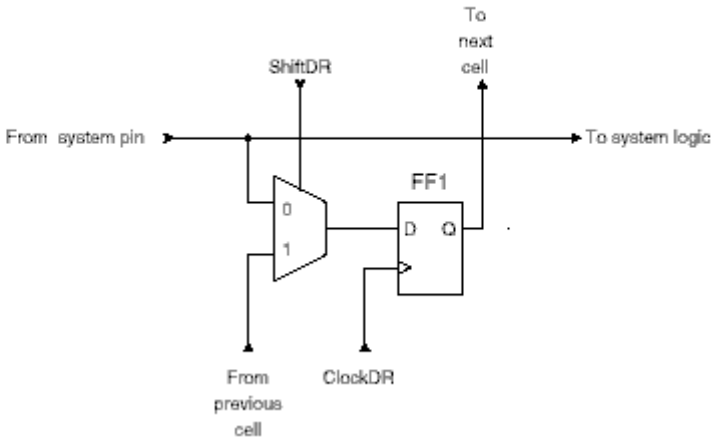
Figure 125. Additional Scan Signal for the Two-wire Interface



- Skanowanie nóżki resetu

Nózki resetu przyjmują 5V aktywnych niskim poziomem dla standartowych operacji resetu i 12V aktywne wysokim poziomem dla wysoko napięciowego równoległego programowania. Komórka obserwujące jest przedstawiona na rysunku 126 i jest do niej pryłożone napięcie 5V RSTT i 12 V RSTHV.

Figure 126. Observe-only Cell



- Skanowanie nóżek zegara

Urządzenie AVR ma wiele różnych źródeł zegara wybieranych przez odpowiednie ustawienia bezpieczników. Są to: wewnętrzny oscylator RC, zewnętrzne RC, zewnętrzny

zegar, (wysoko częstotliwościowy) kwarcowy oscylator, niskoczęstotliwościowy oscylator krzemowy, i ceramiczny rezonator.

Rysunek 127pokazuje jak każdy z oscylatorów z zewnętrznymi połączeniami jest wspierany w łańcuchu skanowania. Sygnał odblokowujący jest wspierany przez główną komórkę łańcucha skanowania podczas, gdy wyjściowy oscylator/zegar jest podłączony do komórki obserwacji. W dodatku do głównego zegara, oscylator licznika jest skanowany w ten sam sposób. Wyjście z wewnętrznego oscylatora RC nie jest skanowane, bo ten oscylator nie ma zewnętrznych połączeń.

Figure 127. Boundary-scan Cells for Oscillators and Clock Options

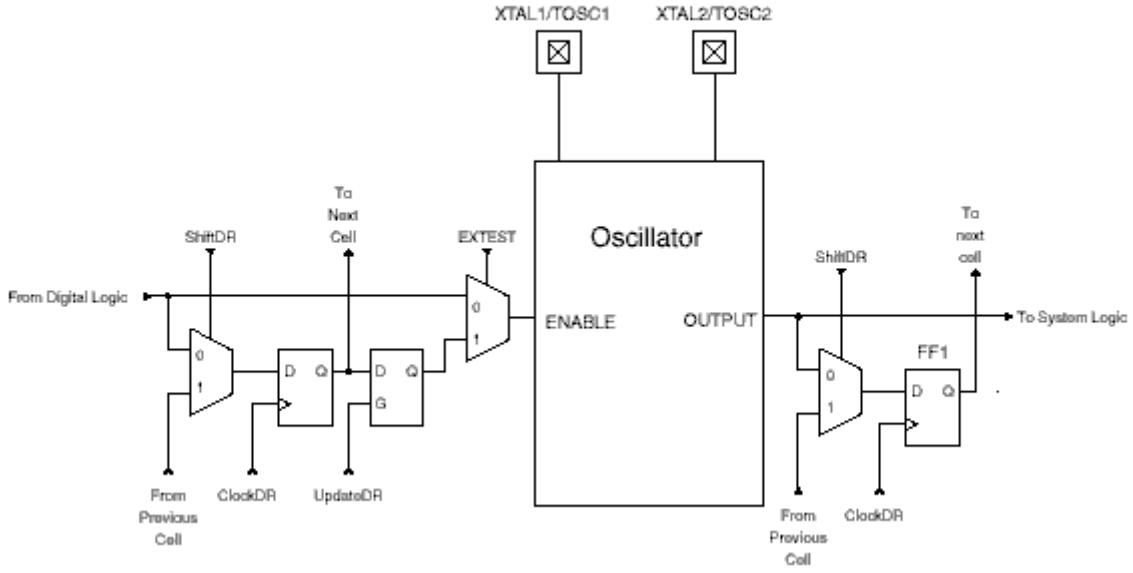


Tabela 102 streszcza rejestry skanujące dla zewnętrznego zegara XTAL1, oscylatory połączone z XTAL1 i XTAL2 jak również z 32kHz oscylatorem licznika.

Tabela102

Sygnał odblokowujący	Linia skanowanego zegara	Opcja zegara	Skanowana linia zegara gdy nie jest on używany
EXTCLKEN	EXTCLK (XTAL1)	Zewnętrzny zegar	0
OSCON	OSCCK	Zewnętrzny kwarc i ceramiczny rezonator	1
RCOSCEN	RCCK	Zewnętrzne RC	1
OSC32EN	OSC32CK	Niskoczęstotliwościowy zewnętrzny kwarc	1
TOSKON	TOSCK	32 kHz oscylator licznika	1

Uwagi: 1. nie należy odblokowywać więcej niż jednego zegara jako głównego źródła zegara.

2. Skanowanie wyjść oscylatora daje nie przewidywalne rezultaty ponieważ jest tam przesunięcie częstotliwości pomiędzy wewnętrznym oscylatorem i zegarem TCK JTAG. Jeżeli jest taka możliwość skanowanie zewnętrznego zegara jest preferowane.

3. Konfiguracja zegara jest programowana przez bezpieczniki. Ponieważ bezpiecznik nie można zmieniać podczas działania zegara ustawienia zegara są stałe dla danej aplikacji. Zaleca się skanowanie opcji zegara, która ma być wykorzystana w wersji końcowej. Sygnały odblokowujące są wspierane w łańcuchu skanowania ponieważ logika systemu może zablokować opcje zegara w trybie snu, w wyniku czego następuje rozłączenie nóżek oscylatora ze ścieżki skanowania. Bezpieczniki INITCAP nie są wspierane przez łańcuch skanowania, więc łańcuch skanowania urządzeń zewnętrznych nie może uruchomić oscylatora XTAL, który wymaga wewnętrznych kondensatorów jeśli bezpieczniki te nie są poprawnie zaprogramowane.

- Skanowanie analogowego komparatora

Odpowiadający sygnał komparatora mający na względzie skanowanie urządzeń zewnętrznych jest przedstawione na rysunku 128. Komórka skanowania urządzeń z rysunku 129 jest podłączona do tych sygnałów. Sygnały opisuje tabela 103.

Komparator nie musi być używany do czystego porównywania testowania odkąd wszystkie analogowe wejścia są dzielone z binarnymi nóżkami portów.

Figure 128. Analog comparator

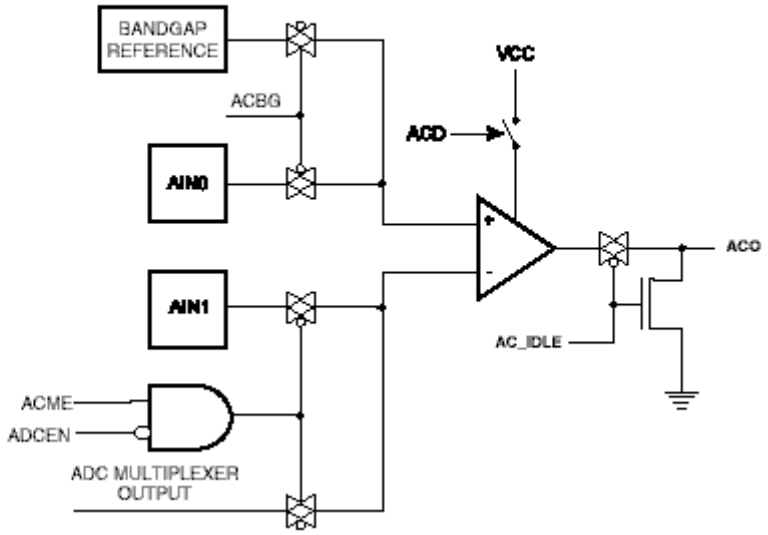


Figure 129. General Boundary-scan Cell used for Signals for Comparator and ADC

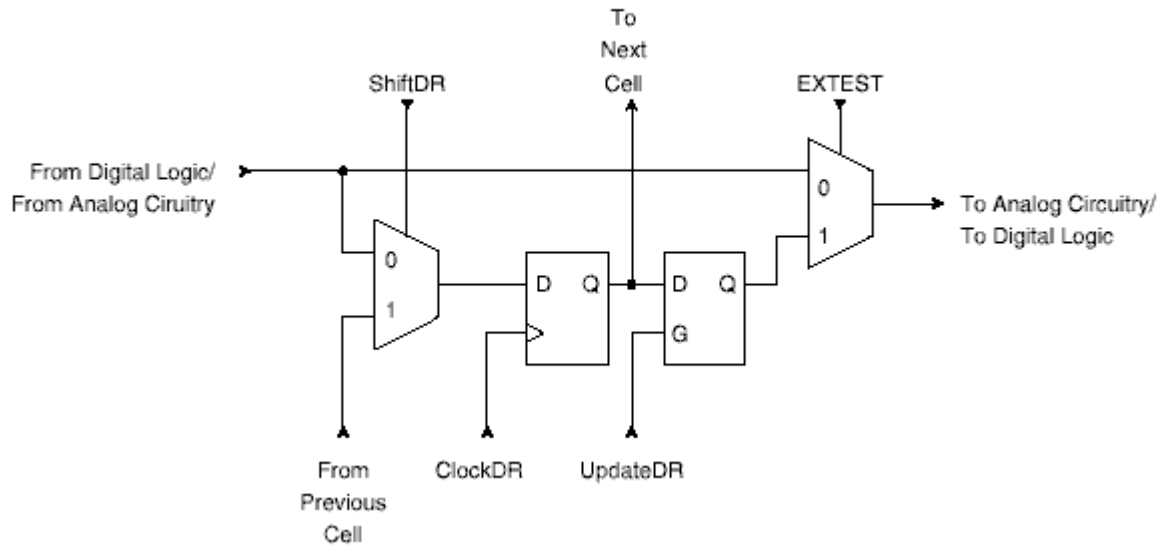


Tabela103

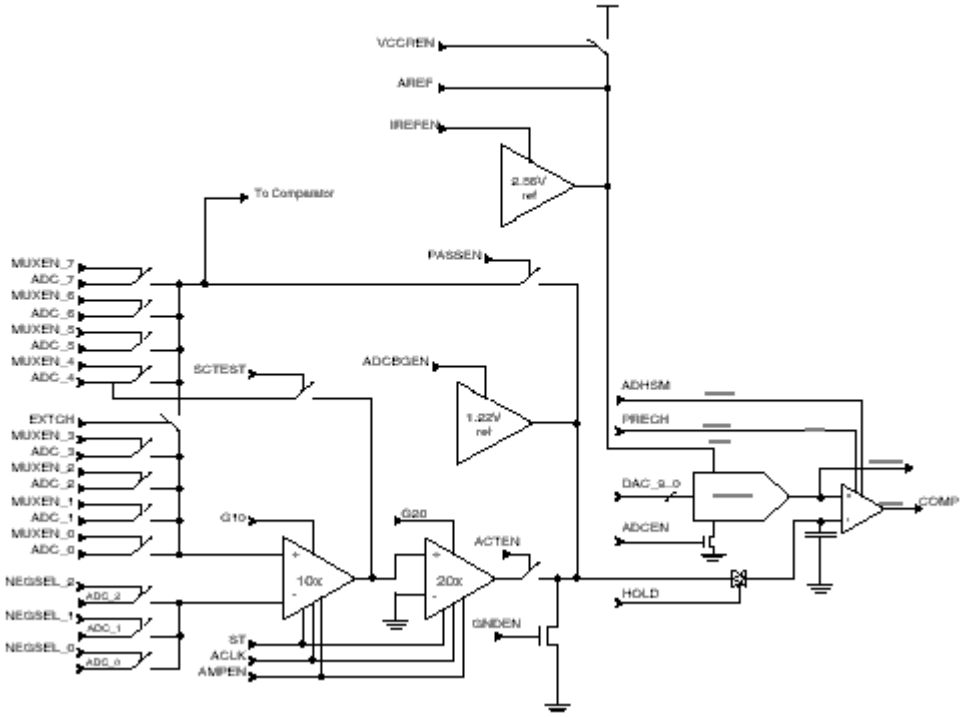
Nazwa sygnału	Kierunek widziany z komparatora	Opis	Zalecane wejście, jeżeli nie jest używane	Wyjścia, kiedy rekomendowane wejścia są wykorzystane
AC_IDYL	**Wejście**	**Wyłącza analogowy komparator jeśli jest ustawiony na true**	**1**	**Zależy od kodu mikroC, który był wykonany**
ACO	**Wyjście**	**Wyjście analogowego komparatora**	**Będzie wejściem jeśli kod mikroC był wykonany**	**0**
ACME	**Wejście**	**Wykorzystuje sygnały wyjścia ADC multipleksowany gdy jest true**	**0**	**Zależy od kodu mikroC, który był wykonany**
ACBG	**Wejście**	**Odblokowanie odwołania do przerwy(bandgap)**	**0**	**Zależy od kodu mikroC, który był wykonany**

- Skanowanie ADC

Rysunek 130 przedstawia schemat blokowy ADC z odpowiednimi sygnałami kontroli i obserwacji. Komórka skanowania urządzeń z rysunku 126 jest dołączona do każdego z

tych sygnałów. ADC nie jest potrzebne do testów podłączenia odkąd wszystkie analogowe wejścia dzielą binarne nóżki portów.

Figure 130. Analog to Digital Converter



Sygnały te są opisane w tabeli 104.
Tabela 104.

Nazwa sygnału	Kierunek widziany od ADC	Opis	Zalecane wejście kiedy nie jest używany	Wartości wyjścia kiedy są wykorzystywane zalecane wejścia i CPU nie korzysta z ADC
COMP	**Wyjście**	**Wyjście komparatora**	**0**	**0**
ACLK	**Wejście**	**Sygnał zegara dla stopnia wzmocnienia zaimplementowanego jako przełączniki filtrów**	**0**	**0**
ACTEN	**Wejście**	**Odblokowanie ścieżki dla stopnia wzmocnienia dla komparatora**	**0**	**0**

ADHMS	**Wejście**	**Zwiększenie szybkości komparatora przy zwiększonym poborze mocy**	**0**	**0**
ADCBGEN	**Wejście**	**Odblokowanie odnośnika band-gap jako ujemnego wejśćia do komparatora**	**0**	**0**
ADCEN	**Wejście**	**Sygnał włączenia mocy dla ADC**	**0**	**0**
AMPEN	**Wejście**	**Sygnał włączenia mocy dla szybkości generatora**	**0**	**0**
DAC_9	**Wejście**	**9 bit wartości do DAC**	**1**	**1**
DAC_8	**Wejście**	**8 bit wartości do DAC**	**0**	**0**
DAC_7	**Wejście**	**7 bit wartości do DAC**	**0**	**0**
DAC_6	**Wejście**	**6 bit wartości do DAC**	**0**	**0**
DAC_5	**Wejście**	**5 bit wartości do DAC**	**0**	**0**
DAC_4	**Wejście**	**4 bit wartości do DAC**	**0**	**0**
DAC_3	**Wejście**	**3 bit wartości do DAC**	**0**	**0**
DAC_2	**Wejście**	**2 bit wartości do DAC**	**0**	**0**
DAC_1	**Wejście**	**1 bit wartości do DAC**	**0**	**0**
DAC_0	**Wejście**	**0 bit wartości do DAC**	**0**	**0**
EXTCH	**Wejście**	**Podłączenie kanałów 0..3 ADC do ścieżki bocznikującej szybkość generacji**	**1**	**1**
G10	**Wejście**	**Odblokowuje 10x przyspieszenie**	**0**	**0**
G20	**Wejście**	**Odblokowuje 20x przyspieszenie**	**0**	**0**
GNDEN	**Wejście**	**Masa ujemne wejście kiedy jest ustawione true**	**0**	**0**
HOLD	**Wejście**	**Sygnały próbkowania i wstrzymania, Próbkuj analogowy sygnał kiey jest stan niski. Wstrzymaj sygnał dla stanu wysokiego. Jeśli wzmocnienie na stopień jest wykorzystywane te sygnały muszą być aktywne jeśli ACLK**	**1**	**1**

		jest w stanie wysokim		
IREFEN	**Wejście**	**Odblokowuje odniesienie do Band-gap jako sygnał AREF do DAC**	**0**	**0**
MUXEN_7	**Wejście**	**Wejście multipleksera 7**	**0**	**0**
MUXEN_6	**Wejście**	**Wejście multipleksera 6**	**0**	**0**
MUXEN_5	**Wejście**	**Wejście multipleksera 5**	**0**	**0**
MUXEN_4	**Wejście**	**Wejście multipleksera 4**	**0**	**0**
MUXEN_3	**Wejście**	**Wejście multipleksera 3**	**0**	**0**
MUXEN_2	**Wejście**	**Wejście multipleksera 2**	**0**	**0**
MUXEN_1	**Wejście**	**Wejście multipleksera 1**	**0**	**0**
MUXEN_0	**Wejście**	**Wejście multipleksera 0**	**1**	**1**
NEGSEL_2	**Wejście**	**Wejście multipleksera dla ujemnego wejscia dla różnicy sygnałów, bit 2**	**0**	**0**
NEGSEL_1	**Wejście**	**Wejście multipleksera dla ujemnego wejscia dla różnicy sygnałów, bit 1**	**0**	**0**
NEGSEL_0	**Wejście**	**Wejście multipleksera dla ujemnego wejscia dla różnicy sygnałów, bit 0**	**0**	**0**
PASSEN	**Wejście**	**Odblokowanie bramy pasmowej dla wzmocnienia na stopień**	**1**	**1**
PRECH	**Wejście**	**Wcześniejsza zmiana zatrzasku wyjścia dla komparatora (aktywny niskim stanem)**	**1**	**1**
SCTEST	**Wejście**	**Odblokowanie przełącznika TEST. Wyjście z 10x przyspieszenia**	**0**	**0**

		wzmocnienia na stopień wysłane od portu mającego ADC_4		
ST	**Wejście**	**Wyjście wzmocnienia na stopień ustawi szybsze jeśli ten sygnał jest wysoki przez pierwsze dwa cykle ACLK po ustawieniu stanu wysokiego na AMPEN**	**0**	**0**
VCCREN	**Wejście**	**Wybiera V_{cc} jako odniesienie ACC do napięcia**	**0**	**0**

Uwagi: Niepoprawne ustawienie przełączników na rysunku 130 spowoduje spór i może uszkodzić część. Jest tu wiele wyborów wejść do obwodu S&H na ujemnym wejściu wyjścia komparatora na rysunku 130. Upewnij się, że tylko jedna część jest wybrana przez każdą nóżkę ADC, źródło odniesienia Band-gap, lub masę.

Jeśli ADC nie ma być wykorzystywane podczas skanowania , zalecane wartości wejść z tabeli 104 powinny być zastosowane. Zaleca się, aby użytkownik nie korzystał ze zmiennych stanów wzmocnień na stopień podczas skanowania. Wzmocnienie na stopień oparta na przełączniku wymaga szybkich operacji i odpowiedniego taktowania, co jest trudno utrzymać przy wykorzystaniu w łańcuchu skanowania. Z tych samych powodów bit trybu wysokiej szybkości ADC (ADHSM) nie ma sensu podczas skanowania.

ACD w AVR jest oparte o obieg analogowy pokazany na rysunku 130 w sukcesywną aproksymacją zaimplementowanych algorytmów w logice binarnej. Kiedy skanowania urządzeń zewnętrznych jest wykorzystywane, problem zazwyczaj zapewnia, że podawane analogowe napięcie jest mierzone w pewnych ograniczeniach. Co może być prosto zrobione bez wykorzystania algorytmu sukcesywnej aproksymacji: podaj dolny limit napięcia na linie binarne DAC[0..9], upewnij się, że wyjście z komparatora jest w stanie niskim następnie podaj wyższy limit napięcia na linie DAC[0..9] i sprawdź czy wyjście z komparatora jest w stanie wysokim.

Komparator ADC nie musi być wykorzystywany do testowania połączeń odką dzieli nóżki wejściowe z portem binarnym.

Przy korzystaniu z ADC należy pamiętać o następujących rzeczach:

- **Nóżki portów wykorzystywane dla kanałów ADC muszą być skonfigurowane jako wejście z zablokowanym podciąganiem w celu uniknięcia najścia na siebie dwóch sygnałów.**
- **W trybie normalnym, sztuczna konwersja (składająca się z 10 porównań) jest wykonywana przy odblokowywaniu ADC. Zaleca się by użytkownik czekał**

przynajmniej 200 ns po odblokowaniu ADC przed kontrolą/obserwowaniem sygnałów ADC lub wykonaniem sztucznej konwersji przed wykorzystaniem pierwszego rezultatu.

- **Wartości DAC muszą być stabilne dla wartości środkowej 0x200 kiedy sygnał HOLD jest w stanie niskim (tryb próbkowania).**

Jak o przykład weźmy pod uwagę zadanie sprawdzenia 1.5V z tolerancją 5% na wejściu sygnału na kanał 3 ADC kiedy moc podaje 5V i AREF jest zewnętrznie podłączony do V_{cc}.

Dolny limit : [1024*1.5V*0.95/5V] = 291 = 0x123
Górny limit : [1024*1.5V*1.05/5V] = 323 = 0x143

Zalecane wartości z tabeli 104 są wykorzystywane jeśli inne wartości nie są podane w algorytmie z tabeli 5. Tylko wartości DAC, nóżek portów i łańcucha skanowania są podane. Kolumna „akcja" opisuje jaki rozkaz JTAG powinien zostać wykorzystany przed wypełnieniem skanowania urządzeń peryferyjnych w następujących kolumnach. Sprawdzenia powinno zostać zrobione poprzez skanowanie wychodzących danych przy skanowaniu danych w tym samym wierszu tabeli.

Tabela 105.

Table 105. Algorithm for Using the ADC

Step	Actions	ADCEN	DAC	MUXEN	HOLD	PRECH	PA3. Data	PA3. Control	PA3. Pullup_ Enable
1	SAMPLE_ PRELOAD	1	0x200	0x08	1	1	0	0	0
2	EXTEST	1	0x200	0x08	0	1	0	0	0
3		1	0x200	0x08	1	1	0	0	0
4		1	0x123	0x08	1	1	0	0	0
5		1	0x123	0x08	1	0	0	0	0
6	Verify the COMP bit scanned out to be 0	1	0x200	0x08	1	1	0	0	0
7		1	0x200	0x08	0	1	0	0	0
8		1	0x200	0x08	1	1	0	0	0
9		1	0x143	0x08	1	1	0	0	0
10		1	0x143	0x08	1	0	0	0	0
11	Verify the COMP bit scanned out to be 1	1	0x200	0x08	1	1	0	0	0

Korzystając z tego algorytmu, ograniczenia taktowania dla sygnału HOLD ograniczają częstotliwość zegara TCK. Ponieważ algorytm utrzymuje HOLD w wysokim stanie przez pięć kroków, częstotliwość zegara TCK musi być przynajmniej pięć razy podzielona prze maksymalny czas hold $t_{hold,max}$.

- Kolejność skanowania urządzeń zewnętrznych w ATmega128

Tabela 106 przedstawia kolejność skanowania między TDI i TDO kiedy łańcuch skanowania jest wybrany jako ścieżka danych. Bit 0 jest LSB – pierwszy bit wsuwany i wysuwany. Kolejność skanowania przebiega zgodnie z kolejnością wyjścia nóżek w takim stopniu jak było ot możliwe. Dlatego bity portu A są skanowane w odwrotnej kolejności niż bity pozostałych portów. Wyjątki z tej reguły stanowią łańcuch skanowania dla analogowego obiegu, który skanuje zgodnie ze starszością bitów nie zależnie od ich fizycznego ułożenia. Na rysunku 123, PXn. Dane odpowiadają FF0, PXn. Kontrola odpowiada FF1 i PXn. Pullup_enable odpowiada FF2. Bity 2, 3, 4 i 5 portu C nie należą do łańcucha skanowania odkąd są one częścią wejść TAP dla odblokowanego JTAG.

Tabela 106

Numer bitu	Nazwa sygnału	Moduł
204	AC_IDLE	Komparator
203	ACO	
202	ACME	
201	AINBG	
200	COMP	ADC
199	PRIVATE_SIGNAL1[(1)]	
198	ACLK	
197	ACTEN	
196	ADHSM	
195	ADCBGEN	
194	ADCEN	
193	AMPEN	
192	DAC_9	
191	DAC_8	
190	DAC_7	
189	DAC_6	
188	DAC_5	
187	DAC_4	
186	DAC_3	
185	DAC_2	
184	DAC_1	
183	DAC_0	
182	EXTCH	
181	G10	
180	G20	
179	GNDEN	
178	HOLD	
177	IREFEN	
176	MUXEN_7	

175	**MUXEN_6**	
174	**MUXEN_5**	
173	**MUXEN_4**	
172	**MUXEN_3**	
171	**MUXEN_2**	
170	**MUXEN_1**	
169	**MUXEN_0**	
168	**NEGSEL_2**	
167	**NEGSEL_1**	
166	**NEGSEL_0**	
165	**PASSEN**	
164	**PRECH**	
163	**SCTEST**	
162	**ST**	
161	**VCCREN**	
160	**PEN**	**Odblokowanie programowania (tylko obserwowany)**
159	**PE0.Data**	**PORT E**
158	**PE0.Control**	
157	**PE0.Pullup_Enable**	
156	**PE1.Data**	
155	**PE1.Control**	
154	**PE1.Pullup_Enable**	
153	**PE2.Data**	
152	**PE2.Control**	
151	**PE2.Pullup_Enable**	
150	**PE3.Data**	
149	**PE3.Control**	
148	**PE3.Pullup_Enable**	
147	**PE4.Data**	
146	**PE4.Control**	
145	**PE4.Pullup_Enable**	
144	**PE5.Data**	
143	**PE5.Control**	
142	**PE5.Pullup_Enable**	
141	**PE6.Data**	
140	**PE6.Control**	
139	**PE6.Pullup_Enable**	
138	**PE7.Data**	
137	**PE7.Control**	
136	**PE7.Pullup_Enable**	
135	**PB0.Data**	**PORT B**
134	**PB0.Control**	
133	**PB0.Pullup_Enable**	
132	**PB1.Data**	

131	PB1.Control	
130	PB1.Pullup_Enable	
129	PB2.Data	
128	PB2.Control	
127	PB2.Pullup_Enable	
126	PB3.Data	
125	PB3.Control	
124	PB3.Pullup_Enable	
123	PB4.Data	
122	PB4.Control	
121	PB4.Pullup_Enable	
120	PB5.Data	
119	PB5.Control	
118	PB5.Pullup_Enable	
117	PB6.Data	
116	PB6.Control	
115	PB6.Pullup_Enable	
114	PB7.Data	
113	PB7.Control	
112	PB7.Pullup_Enable	
111	PG3.Data	PORT G
110	PG3.Control	
109	PG3.Pullup_Enable	
108	PG4.Data	
107	PG4.Control	
106	PG4.Pullup_Enable	
105	TOSC	32 kHz oscylator licznika
104	TOSCON	
103	RSTT	Logiczny poziom resetu (tyko obserwowany)
102	RSTHV	
101	EXTLKEN	Odblokowanie sygnałów dla głównego zegara/oscylatora
100	OSCON	
99	RCOSCEN	
98	OSC32EN	
97	EXTCLK (XTAL1)	Wejścia zegara i oscylatora dla głównego zegara (tylko obserwowane)
96	OSCCK	
95	RCCK	
94	OSC32CK	
93	TWIEN	TWI
92	PD0.Data	PORT D
91	PD0.Control	
90	PD0.Pullup_Enable	
89	PD1.Data	
88	PD1.Control	
87	PD1.Pullup_Enable	

86	**PD2.Data**	
85	**PD2.Control**	
84	**PD2.Pullup_Enable**	
83	**PD3.Data**	
82	**PD3.Control**	
81	**PD3.Pullup_Enable**	
80	**PD4.Data**	
79	**PD4.Control**	
78	**PD4.Pullup_Enable**	
77	**PD5.Data**	
76	**PD5.Control**	
75	**PD5.Pullup_Enable**	
74	**PD6.Data**	
73	**PD6.Control**	
72	**PD6.Pullup_Enable**	
71	**PD7.Data**	
70	**PD7.Control**	
69	**PD7.Pullup_Enable**	
68	**PG0.Data**	**PORT G**
67	**PG0.Control**	
66	**PG0.Pullup_Enable**	
65	**PG1.Data**	
64	**PG1.Control**	
63	**PG1.Pullup_Enable**	
62	**PC0.Data**	**PORT C**
61	**PC0.Control**	
60	**PC0.Pullup_Enable**	
59	**PC1.Data**	
58	**PC1.Control**	
57	**PC1.Pullup_Enable**	
56	**PC2.Data**	
55	**PC2.Control**	
54	**PC2.Pullup_Enable**	
53	**PC3.Data**	
52	**PC3.Control**	
51	**PC3.Pullup_Enable**	
50	**PC4.Data**	
49	**PC4.Control**	
48	**PC4.Pullup_Enable**	
47	**PC5.Data**	
46	**PC5.Control**	
45	**PC5.Pullup_Enable**	
44	**PC6.Data**	
43	**PC6.Control**	
42	**PC6.Pullup_Enable**	

41	**PC7.Data**	
40	**PC7.Control**	
39	**PC7.Pullup_Enable**	
38	**PG2.Data**	**PORT G**
37	**PG2.Control**	
36	**PG2.Pullup_Enable**	
35	**PA7.Data**	**PORT A**
34	**PA7.Control**	
33	**PA7.Pullup_Enable**	
32	**PA6.Data**	
31	**PA6.Control**	
30	**PA6.Pullup_Enable**	
29	**PA5.Data**	
28	**PA5.Control**	
27	**PA5.Pullup_Enable**	
26	**PA4.Data**	
25	**PA4.Control**	
24	**PA4.Pullup_Enable**	
23	**PA3.Data**	
22	**PA3.Control**	
21	**PA3.Pullup_Enable**	
20	**PA2.Data**	
19	**PA2.Control**	
18	**PA2.Pullup_Enable**	
17	**PA1.Data**	
16	**PA1.Control**	
15	**PA1.Pullup_Enable**	
14	**PA0.Data**	
13	**PA0.Control**	
12	**PA0.Pullup_Enable**	
11	**PF3.Data**	**PORT F**
10	**PF3.Control**	
9	**PF3.Pullup_Enable**	
8	**PF2.Data**	
7	**PF2.Control**	
6	**PF2.Pullup_Enable**	
5	**PF1.Data**	
4	**PF1.Control**	
3	**PF1.Pullup_Enable**	
2	**PF0.Data**	
1	**PF0.Control**	
0	**PF0.Pullup_Enable**	

Uwaga: 1. PRIVATE_SIGNAL1 powinien być skanowany na wejściu jako 1.

- Języka opisujący w plikach skanowania urządzeń zewnętrznych

Język opisujący skanowanie urządzeń peryferyjnych (BSDL) opisuje zdolność skanowania urządzeń w standardowym formacie wykorzystującym przez zautomatyzowane generacje oprogramowania testującego. Kolejność i funkcje bitów rejestru danych skanowania urządzeń peryferyjnych jest zawarty w opisie.

26. Wsparcie programu ładującego - odczyt podczas zapisu, samo-programowanie

Wsparcie programu ładującego zapewnia rzeczywisty mechanizm zapisu podczas odczytu samo zaprogramowywania się dla ładowania i załadowywania kodów programów przez MCU. Ta cech umożliwia dogodne aktualizowanie oprogramowania kontrolowane przez MCU wykorzystujące program ładujący znajdujący się w pamięci Flash. Program ładujący może wykorzystywać jakiekolwiek dostępne interfejsy danych i związane z nimi protokoły odczytu i zapisu kodu do pamięci Flash lub odczytu tego kodu z pamięci Flash. Kod programu znajdujący się w sekcji programu ładującego pamięci ma możliwość do zapisu po całym Flash włączając w to zapis w sekcji programu ładującego. W związku z tym program ładujący może sam siebie modyfikować i wymazać z pamięci urządzenia jeśli nie będzie więcej potrzebny. Rozmiar pamięci z programem ładującym jest konfigurowana przez bezpieczniki i Program ładujący ma dwa zestawy bitów blokujących program ładujący, które mogą być ustawiane niezależnie. Daje to użytkownikowi unikalne dopasowanie do wyboru różnych poziomów ochrony.

- **Cech programu ładującego**
  1. zapis podczas odczytu samo programujący się
  2. zmienny rozmiar pamięci przeznaczonej na program ładujący
  3. wysokie bezpieczeństwo (gwarantowane poprzez oddzielne bity blokujące program ładujący)
  4. oddzielny bezpiecznik wybierający wektor resetu
  5. zoptymalizowany rozmiar strony
  6. efektywny algorytm
  7. wsparcie odczytu podczas zapisu

- **Sekcje programów i programu ładującego w pamięci Flash**

Pamięć Flash jest podzielona na dwie sekcje – sekcję programów i sekcję programu ładującego (rysunek 132). Rozmiar poszczególnych sekcji jest ustawiany za pomocą bezpiecznika BOOTSZ co przedstawia tabela 280 i rysunek 132. Sekcje te mają różny poziom ochrony ponieważ mają oddzielne zestawy bitów blokujących dostęp.

- **Sekcja programów**

Sekcja ta jest wykorzystywana do przechowywania kodu programów. Poziom zabezpieczenia może zostać wybrany za pomocą bitów blokujących ładowanie (bit 0), tabela 271. W trej części pamięci nie może znajdować się program ładujący ponieważ instrukcje SPM są zablokowane przy wykonywaniu programu z tej sekcji.

- **Sekcja programu ładującego BLS**

Podczas gdy sekcja programów służy do składowania ich kodu, to program ładujący musi być składowany w sekcji BLS ponieważ tylko instrukcje SPM (wykonane z tej części pamięci) mogą inicjalizować programowanie. Rozkazy SPM mogą operować na całej pamięci, także część BLS. Poziom ochrony dla tej sekcji może zostać wybrany za pomocą bitów blokujących ładowanie (bit 1), tabela 109 na stronie 272.

- **Sekcje odczytu podczas zapisu i oddzielnego zapisu i odczytu**

Czy CPU wspiera odczyt podczas zapisu lub jest wstrzymane przy aktualizacji oprogramowania ładującego zależy od zaprogramowanego adresu. W dodatku do dwóch sekcji konfigurowanych przez bezpiecznik BOOTSZ, jak opisano powyżej, Flash jest podzielony na dwie stałe sekcje: odczyt podczas zapisu RWW i oddzielny odczyt i zapis NRWW. Granica pomiędzy tymi sekcjami jest podana w uwadze do tabeli na stronie 280 i rysunku 132 na stronie 271. Głównymi różnicami pomiędzy tymi sekcjami są:

  - wymazując lub zapisując stronę w części RWW, część NRWW może być jednocześnie odczytywana
  - wymazując lub zapisując stronę w części NRWW, CPU jest zatrzymane na całą tą operację.

Warto zauważyć, że program użytkownika nie może czytać z sekcji RWW podczas operacji programu ładującego. Składnia sekcja odczyt podczas zapisu odnosi się do sekcji programowanej, która nie jest aktualnie odczytywana podczas aktualizacji oprogramowania ładującego.

- **Sekcja odczyt podczas zapisu RWW**

Jewśli aktualizacja oprogramowania programuje stronę w sekcji RWW można odczytywać kod z Flash, ale tylko taki, który jest umieszczony w NRWW. Podczas programowania oprogramowanie musi zapewnić, że sekcja RWW nigdy nie zostanie odczytana. Jeśli program użytkowy próbuje odczytać kod znajdujący się w tej części pamięci (call/jmp/lpm lub przerwanie) podczas programowania, program może zakończyć się w nieprzewidywalnym stanie. Aby temu zapobiec przerwania powinny być albo zablokowane albo przeniesione do sekcji programu ładującego. Sekcja programu ładującego jest zawsze umieszczona w części NRWW pamięci. Bit zajęcia sekcji RWW (RWWSB) w rejestrze kontroli składowania programu (SPMCR) będzie odczytywany jako logiczna jedynka tak długo jak sekcja RWW jest zablokowana do odczytu. PO zakończeniu programowania, bit RWWSB musi zostać wyzerowany przez oprogramowanie przed odczytem kodu umieszczonego w RWW. Szczegóły w „Rejestr kontroli składowania programu – SPMCR” na stronie 273.

- **Sekcja oddzielnego odczytu i zapisu – NRWW**

Kod znajdujący się w sekcji NRWW może być odczytywany kiedy program ładujący aktualizuje oprogramowanie w sekcji RWW. Kiedy program ładujący aktualizuje sekcje NRWW, CPU jest wstrzymane na czas wymazania lub zapisania całej strony.

Tabela 107

Którą sekcję wskazuje wskaźnik Z podczas programowania?	Która sekcja może być odczytywana podczas programowania?	Czy CPU jest wstrzymane?	Czy odczyt podczas zapisu jest wspierany?
Sekcja RWW	Sekcja NRWW	Nie	Tak
Sekcja NRWW	Nic	Tak	Nie

Figure 131. Read-While-Write vs. No Read-While-Write

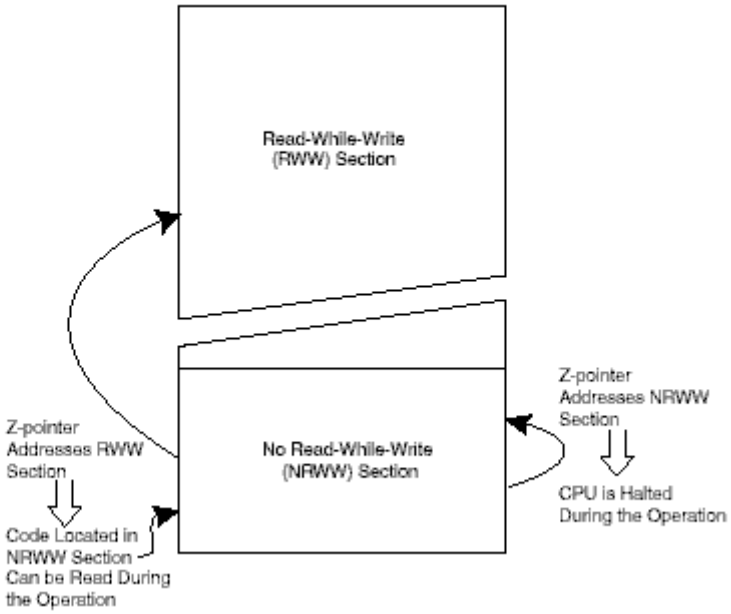
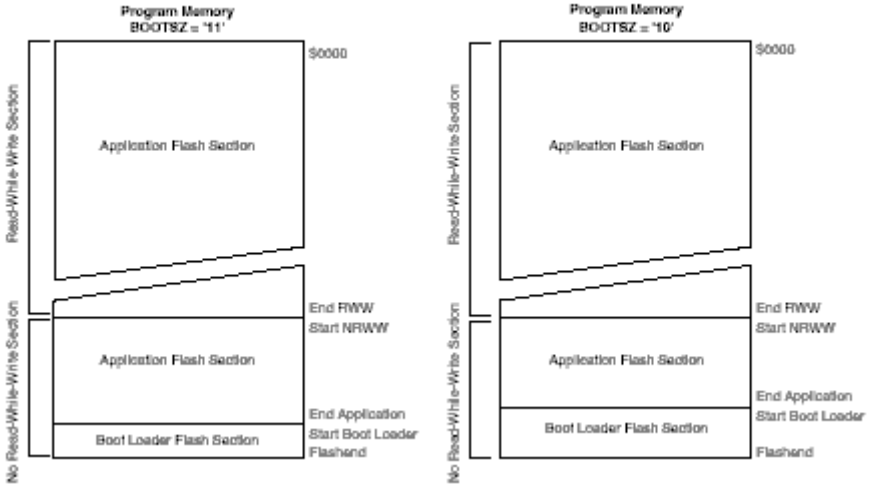
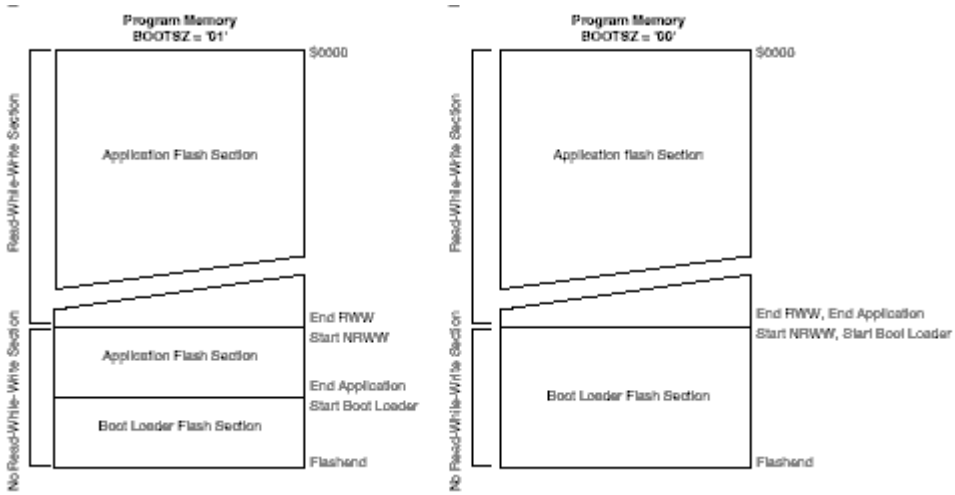


Figure 132. Memory Sections(1)





Uwaga: parametry wykorzystane na rysunku są dane w tabeli 280

- **Bity blokujące program ładujący**

Jeśli nie jest potrzebna żadna ze zdolności programu ładującego to cała pamięć Flash jest przeznaczona dla programów użytkowych. Program ładujący ma dwa zestawy bitów blokujących ładowanie, które mogą być ustawiane niezależnie. To daje użytkownikowi unikalne możliwości wyboru poziomu zabezpieczeń.

Użytkownik może wybrać:

- Ochronę całej pamięci Flash przed aktualizacją oprogramowania przez MCU
- Ochronę tylko części programu ładującego przed aktualizacją oprogramowania przez MCU
- Ochronę części programowej przed aktualizacją oprogramowania przez MCU
- Umożliwia aktualizację całej pamięci

Szczegóły zawarte są w tabelach 108 i 109. Bity blokujące program ładujący mogą zostać ustawione w programie lub w szeregowym lub równoległym trybie programowania, ale mogą być zerowane tylko przez sprzętowy rozkaz. Główna blokada zapisu (bit blokujący – tryb 2) nie kontroluje programowania pamięci Flash przez rozkazy SPM. Tak jak i główna blokada odczytu/zapisu (bit blokady, tryb 1) nie kontroluje odczytu lub zapisu prze LPM/SPM jeśli jest to zamierzone.

Tabela 108

Tryb BLB0	BLB02	BLB01	Ochrona
1	1	1	Nie ma żadnych ograniczeń nałożonych na SPM i LPM przy odwoływaniu się do pamięci programów
2	1	0	SPM nie może zapisywać do pamięci programów
3	0	0	SPM nie może zapisywać do pamięci programów i LPM nie może wykonywać rozkazów z sekcji programu ładującego i czytać z sekcji programów. Jeśli wektory przerwań są umieszczone w części pamięci programu ładującego, przerwania są zablokowane podczas

			wykonywania programu z pamięci programów.
4	0	1	LPM nie może wykonywać rozkazów z sekcji programu ładującego i czytać z sekcji programów. Jeśli wektory przerwań są umieszczone w części pamięci programu ładującego, przerwania są zablokowane podczas wykonywania programu z pamięci programów.

Uwaga: 1 znaczy nie zaprogramowany – 0 zaprogramowany

Tabela 109

Tryb BLB1	BLB12	BLB11	Ochrona
1	1	1	Nie ma żadnych ograniczeń nałożonych na SPM i LPM przy odwoływaniu się do pamięci programu ładującego
2	1	0	SPM nie może zapisywać do pamięci programu ładującego
3	0	0	SPM nie może zapisywać do pamięci programu ładującego i LPM nie może wykonywać rozkazów z sekcji programów i czytać z sekcji programów. Jeśli wektory przerwań są umieszczone w części pamięci programów, przerwania są zablokowane podczas wykonywania programu z pamięci programu ładującego.
4	0	1	LPM nie może wykonywać rozkazów z sekcji programów i czytać z sekcji programu ładującego. Jeśli wektory przerwań są umieszczone w części pamięci programów, przerwania są zablokowane podczas wykonywania programu z pamięci programu ładującego.

- **Wejście do programu ładującego**

Wejście do programu ładującego następuje po wywołaniu rozkazu jump lub call z oprogramowania. Może to zostać zainicjalizowane poprzez komendę dostarczoną poprzez USART lub interfejs SPI. Alternatywnie, bezpiecznik resetu do programu ładującego może zostać zaprogramowany w taki sposób, że wektor resetu wskazuje na program ładujący we Flash po resecie. W takim przypadku, program ładujący jest uruchamiany po resecie. Po załadowaniu kodu oprogramowania, program może rozpocząć wykonywanie programu użytkowego. Bezpieczniki nie mogą być zmieniane przez MCU. Oznacza to, że bezpiecznik resetu programu ładującego raz zaprogramowany będzie wskazywał wektorem resetu na reset programu ładującego i bezpiecznik ten może zostać zmieniony tyko przez szeregowy lub równoległy interfejs.

Tabela 110

BOOTRST	Adres resetu
1	Wektor resetu = reset oprogramowania ($0000)
0	Wektor resetu = reset programu ładującego(tabela 112 na stronie 280)

- **Rejestr kontroli składowania programu w pamięci – SPMCR**

Rejestr kontroli przechowywania programów zawiera bity kontroli potrzebne do kontroli operacji programu ładującego.

Bit	7	6	5	4	3	2	1	0	
	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

a: bit 7 – SPMIE: bit odblokowujący przerwania SPM

Kiedy bit ten ma ustawioną jedynkę i bit I w rejestrze statusu jest ustawiony (1) przerwanie gotowości SPM będzie odblokowane. Przerwanie to będzie wykonywane tak długo jak bit SPMEN w rejestrze SPMCR będzie wyzerowany.

b: bit 6 – RWWSB: odczyt podczas zapisu przy zajęciu sekcji

Podczas operacji samo programowania (wymazanie lub zapis strony) sekcja RWW zostanie zainicjalizowana bit RWWSB zostanie ustawiony przez sprzęt. Kiedy bit ten jest ustawiony, sekcja RWW nie można wejść do tej sekcji. Bit ten jest zerowany jeśli bit RWWSRE jest zapisany jedynką po ukończeniu operacji samo programowania. Lub bit ten jest automatycznie zerowany jeśli zostaie zainicjalizowana operacja ładowania strony.

c: bit 5 – Res: bit zarezerwowany

Bit ten jest zarezerwowany i w ATmega128 zawsze będzie miał wartość zero.

d: bit 4 – RWWSRE: odblokowanie sekcji odczyt podczas zapisu do odczytu

Podczas programowania sekcji RWW, sekcja ta jest zablokowana do odczytu (RWWSB zostaje ustawiony przez sprzęt). Aby zmienić ustawienia sekcji RWW program użytkownika musi czekać, aż programowanie się zakończy (SPMEN zostanie wyzerowane). Potem, jeśli bit RWWSRE jest ustawiony na jeden w tym samym czasie co SPMEN, następna instrukcja SPM w przeciągu czterech cykli zegarowych odblokuje sekcję RWW. Sekcja ta nie może zostać odblokowana kiedy Flash jest zajęty zapisem lub wymazywaniem strony. Jeśli bit RWWSRE jest ustawiony podczas ładowania danych do pamięci Flash, operacja ładowania do pamięci zostanie przerwana, a dane stracone.

e: bit 3 – BLBSET: bit ustawiający blokadę programu ładującego

Jeśli bit ten jest ustawiany na jeden w tym samym czasie co SPMEN, następny rozkaz SPM w przeciągu czterech cykli ustawi bity blokujące program ładujący zgodnie z zawartością rejestru R0. Dane z R1 i adres wskaźnika Z są ignorowane. Bit ten automatycznie zostanie wyzerowany po ukończeniu ustawienia bitu blokującego lub jeśli nie jest wykonywana żadna instrukcja SPM w przeciągu czterech cykli maszynowych.

Instrukcja LPM wciągu trzech cykli maszynowych po ustawieniu BLBSET i SPMEN odczyta albo bity blokujące lub bity bezpieczników (w zależności od Z0 we wskaźniku Z) do rejestru

przeznaczenia. Dalsze informacje w „Odczyt bezpieczników i bitów blokujących z programu” na stronie 277.

f: bit 2 _ PGWRT: zapis strony

Jeśli bit ten jest ustawiony na jeden w tym samym czasie co SPMEN to następna instrukcja SPM zapisze stronę do pamięci zawierającą dane przechowywane w tymczasowym buforze. Adres strony jest wskazywany przez wyższe bity wskaźnika Z. Dane w rejestrach R0 i R1 są ignorowane. Bit PGWRT będzie automatycznie wyzerowany po ukończeniu zapisu strony lub jeśli żadna instrukcja SPM nie zostanie wykonana w przeciągu czterech cykli maszynowych. CPU jest wstrzymane na czas zapisu całej strony jeśli jest zaadresowana sekcja NRWW.

g: bit 1-PGERS: wymazanie strony

Jeśli bit ten jest ustawiony na jeden w tym samym czasie co SPMEN to następna instrukcja SPM wymaże stronę z pamięci. Adres strony jest wskazywany przez wyższe bity wskaźnika Z. Dane w rejestrach R0 i R1 są ignorowane. Bit PGERS będzie automatycznie wyzerowany po ukończeniu zapisu strony lub jeśli żadna instrukcja SPM nie zostanie wykonana w przeciągu czterech cykli maszynowych. CPU jest wstrzymane na czas zapisu całej strony jeśli jest zaadresowana sekcja NRWW.

h: bit 0 – SPMEN: odblokowanie składowania programów do pamięci

Bit ten uaktywnia instrukcje SPM na następne cztery cykle maszynowe. Jeśli jest zapisywany jedynką wraz z RWWSRE, BLBSET, PGWRT lub PGERS następne instrukcja będą miały specjalne znaczenie opisane powyżej. Jeżli tylko bit SPMEN jest zapisane jedynką następne instrukcje przeniosą dane z rejestrów R0 i R1 do tymczasowego bufora zaadresowanego przez wskaźnik Z. LBS wskaźnika Z jest ignorowane. Bit ten zostanie automatycznie wyzerowany po zakończeniu instrukcji SPM lub po czterech cyklach maszynowych jeśli nie została wykonana żadna instrukcja SPM. Podczas zapisu lub wymazywania strony bit SPMEN pozostaje w stanie wysokim aż do ukończenia oeracji.

Wpisanie innych kombinacji w młodsze bity niż: 10001, 01001, 00101, 00011, 00001 nie będzie miała żadnego efektu.

- **Adresowanie pamięci Flash podczas samo zaprogramowywania.**

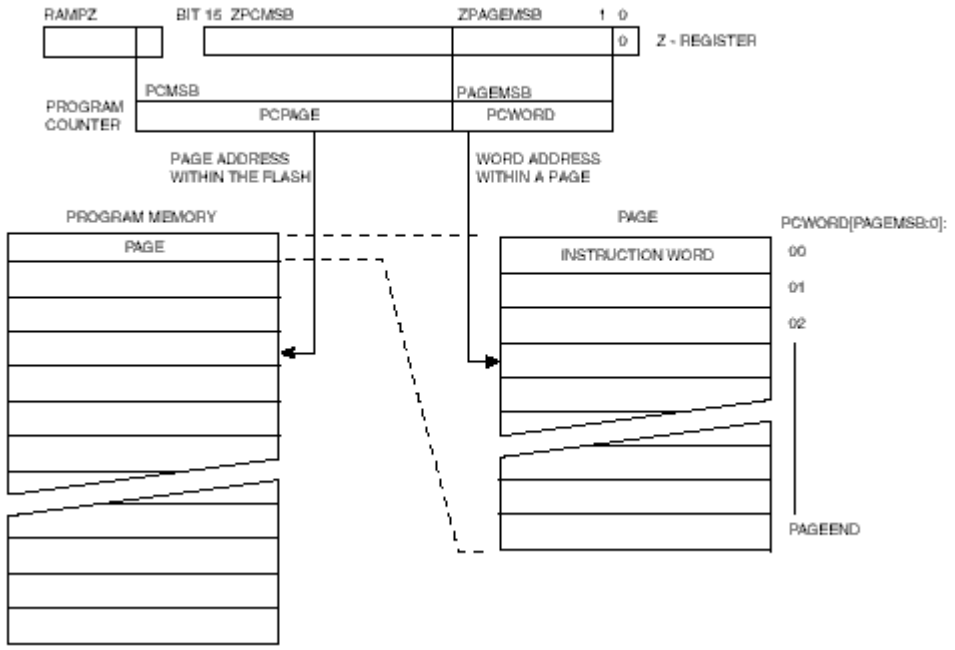
Wskaźnik Z wraz z RAMPZ jest wykorzystywany do adresowania rozkazów SPM. Szczegółowy opis w „rejestr wyboru strony RAM – RAMPZ” na stronie 12.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8
ZL (R30)	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
	7	6	5	4	3	2	1	0

Odkąd Flash jest podzielony na strony (tabela 123 na stronie 287) licznik programu może być traktowany jak by posiadał dwie różne sekcje. Pierwsza sekcja, składająca się z młodszych bitów, adresuje poszczególne słowa na stronie, podczas gdy starsze bity wybierają stronę. Przedstawia to rysunek 133. Warto zauważyć, że operacje zapisu i wymazywania strony są adresowane niezależnie. Dlatego ważnym jest by program ładujący adresował te same strony przy zapisie co przy wymazywaniu. Kiedy operacja została rozpoczęta wartość wskaźnika Z jest zatrzaśnięta i wskaźnik Z wraz z RAMPZ mogą być wykorzystywane w innych operacjach.

Jedyną operacją SPM, która nie korzysta ze wskaźnika Z i RAMPZ jest ustawienie bitów blokujących program ładujący. Zawartość wskaźnika Z i RAMPZ jest ignorowana i nie ma wpływu na tą operację. Rozkazy (E)LPM również wykorzystują wskaźnik Z i RAMPZ do przechowywania adresu. Odkąd instrukcja adresują Flash bajt po bajcie również LSB(bit Z0) wskaźnika Z jest wykorzystywany.

Figure 133. Addressing the Flash During SPM[1]



Uwaga: Zmienne wykorzystane na rysunku są zamieszczone w tabeli 114 na stronie 281.

- **Samo zaprogramowywanie się Flash.**

Pamięć programów jest aktualizowana w trybie strona po stronie. Przed zaprogramowaniem strony, dane są przechowywanie w tymczasowym buforze, strona musi zostać wymazana. Tymczasowy bufor jest wypełniany słowo za słowem przy wykorzystaniu SPM i może zostać zapełniony przed rozkazem wymazania sttrony lub między rozkazem wymazania a zapisania strony.

Możliwość 1, napełnienie bufora przed wymazaniem strony:

- wypełnij tymczasowy bufor strony
- wymarz stronę
- zapisz stronę

Możliwość 2, wypełnienie bufora po wymazaniu strony

- wymaż stronę
- wypełnij bufor
- zapisz stronę

Jeśli tylko część strony ma ulec zmianie, reszta strony musi być składowana (np. w tymczasowym buforze) przed wymazaniem i dopiero wtedy powtórnie zapisana. W przypadku Możliwości 1 program ładujący zapewnia efektywny odczyt podczas zapisu co pozwala aby program użytkowy pierwszy odczytywał strony, robił odpowiednie zmiany i zapisywał z powrotem dane. Wykorzystując Możliwość 2, nie ma możliwość odczytania starych danych podczas ładowania ponieważ strona ta została wymazana. Dostęp do tymczasowego bufora strony może być realizowany w przypadkowej kolejności. Ważne jest tylko by adres strony wymazywanej i na nowo zapisywanej był taki sam. Przejrzyj „Proste przykłady kody w asemblerze...” na stronie 278.

- **Wymazywanie stron przez SPM**

Aby wymazać stronę należy wpisać jej adres do wskaźnika Z i RAMPZ, zapisać X0000011 do SPMCR i wykonać rozkaż SPM w przeciągu czterech cykli maszynowych po zapisie do SPRCR. Dane w rejestrach R0 i R1 są ignorowane. Adres strony musi zostać wpisany do PCPAGE w rejestrze Z. Pozostałe bity we wskaźniku Z będą ignorowane.

- Wymazanie strony z sekcji RWW: sekcja NRWW może być odczytywana w tym czasie.
- Wymazanie strony z sekcji NRWW: CPU jest wstrzymane podczas tej operacji

- **Wypełnianie tymczasowego bufora (ładowanie stron)**

Aby wpisać słowo sterujące należy ustawić adres we wskaźniku Z i dane w rejestrach R0 i R1, zapisać 00000001 do SPMCR i wykonać rozkaż SPM w przeciągu czterech cykli po zapisie SPMCR. Zawartość PCWORD w rejestrze Z jest wykorzystywana do zaadresowania danych w tymczasowym buforze danych. Bufor tymczasowy zostanie wymazany automatycznie po zapisie strony lub przy zapisie RWWSRE w rejestrze SPMCR. Jest on również kasowany po resecie systemu. Należy zauważyć, że niemożliwy jest wielokrotny zapis do tego samego adresu bez wymazania bufora.

- **Zapis strony**

Przy zapisie strony należy stawić adres we wskaźniku Z i RAMPZ, zapisać X0000101 do SPMCR i wykonać rozkaz SPM w przeciągu czterech cykli maszynowych od zapisu do SPMCR. Dane w R0 i R1 są ignorowane. Adres strony musi zostać zapisany do PCPAGE. Pozostałe bit we wskaźniku Z będą ignorowane podczas tej operacji.

- Zapis strony z sekcji RWW: sekcja NRWW może być odczytywana w tym czasie.
- Zapis strony z sekcji NRWW: CPU jest wstrzymane podczas tej operacji

- **Wykorzystywanie przerwań SPM**

Jeśli przerwania SPM sa odblokowane , przerwanie SPM wygeneruje stałe przerwanie przy zerowaniu bitu SPMEN w SPMCR. Oznacza to, że przerwania mogą zostać wykorzystane zamiast odpytywania rejestru SPMCR przez oprogramowanie. Wykorzystując przerwania SPM wektory przerwań powinny zostać przeniesione do sekcji BLS w celu ograniczenia dostępu przerwaniom do sekcji RWW kiedy jest tam blokada odczytu. Jak przenieść wektory przerwań jest opisane w „Przerwania” na stronie 54.

- **Rozwaga przy aktualizowaniu BLS**

Specjalna ostrożność musi zostać zachowana jeśli użytkownik umożliwia aktualizowanie sekcji programu ładującego pozostawiając bit11 blokujący tą sekcję nie zaprogramowany. Przypadkowy zapis do tej sekcji może zniszczyć cały program ładujący i uniemożliwić przyszłą aktualizację oprogramowania. Jeśli nie potrzebujemy zmieniać programu ładującego zaleca się zaprogramowanie tego bitu w celu ochrony programu ładującego przed zmianami.

- **Zabezpieczenie przed odczytem sekcji RWW podczas samo zaprogramowywania się**

Podczas samo zaprogramowyania się sekcja RWW jest zawsze zablokowana do odczytu. Program użytkowy musi zapewnić, że ta sekcja jest adresowana podczas programowania. Bit RWWSB w SPMCR będzie ustawiony tak dług jak sekcja RWW będzie zajęta. Podczas samo programowania się tablica wektorów przerwań powinna zostać przeniesiona do sekcji BLS – opisane na stronie 54. lub przerwania powinny zostać zablokowane. Przed zaadresowaniem RWW po ukończeniu programowania program użytkowy musi wyzerować RWWSB przez zapis do RWWSRE. Przejrzyj „Proste przykłady kody w asemblerze...” na stronie 278.

- **Ustawianie bitów blokujących program ładujący przez SPM**

Aby ustawić bity blokujące program ładujący należy wpisać dane do R0, wpisać X0001001 do SPMCR i wykonać instrukcje SPM w przeciągu czterech cykli po zapisie do SPMCR. Jedynymi dostępnymi bitami blokującymi są bity programu ładującego, które mogą powstrzymać oprogramowanie i sekcje programu ładującego przed jakimikolwiek aktualizacjami MCU.

Bit	7	6	5	4	3	2	1	0
R0	1	1	BLB12	BLB11	BLB02	BLB01	1	1

Obejrzyj tabele 108 i 109 jak różne ustawienia bitów blokujących wpływają na dostęp do Flash.

Jeśli bity 5..2 w R0 są wyzerowane, odpowiadające im bity blokujące program ładujący zosttaną zaprogramowane jeśli instrukcja SPM zostanie wykonana w przeciągu czterech cykli maszynowych po ustawieniu BLBSET i SPMEN w SPMCR. Wskaźnik Z nie wpływa na tą operację, ale w celu zapewnienia kompatybilności z przyszłymi urządzeniami zaleca się zapisanie go $0001. W celu zapewnienia kompatybilności z przyszłymi urządzeniami zaleca się również ustawienie bitów 7, 6 , 1 i 0 w R0 na 1 przy zapisie bitów blokujących. Przy programowaniu tych bitów można odczytywać z całego Flash.

- **Zapis EEPROM zapobiegający zapisowi SPMCR**

Należy zauważyć, że operacja zapisu do EEPROM zablokuje całe oprogramowanie programujące Flash. Jak również uniemożliwi odczyt bezpieczników i bitów blokujących przez oprogramowanie. Zaleca się, aby użytkownik sprawdził bit statusu (EEWE) w EECR i zapewnił wyzerowanie go przed zapisem do rejestru SPMCR.

- **Odczyt bezpiecznika i bitów blokujących z programu**

Jest możliwe równoczesne odczytywanie bezpieczników i bitów blokujących przez oprogramowanie. W celu odczytania bitów blokujących należy załadować wskaźnik Z $0001 i ustawić bit BLBSET w SPMCR. Kiedy rozkaz LPM jest wykonywany w przeciągu trzech cykli CPU po ustawieniu bitu BLBSET i SPMEN wartość bitów blokujących zostanie zapisana w rejestrze przeznaczenia. Bity BLBSET i SPMEN zostaną automatycznie wyzerowane po ukończeniu odczytu lub jeśli żadna instrukcja LPM lub SPM nie została wykonana w tym czasie. Kiedy bity BLBSET i SPMEN są wyzerowane rozkaz LPM będzie pracował jak opisano w instrukcji.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	BLB12	BLB11	BLB02	BLB01	LB2	LB1

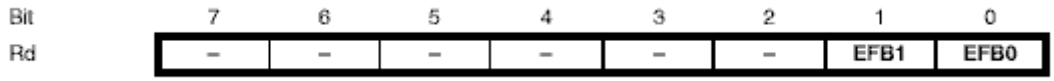
Algorytm odczytu niższych bitów bezpieczników jest identyczny do opisanego powyżej. W celu odczytania niższych bitów bezpieczników należy załadować wskaźnik Z $0000 i ustawić bit BLBSET w SPMCR. Kiedy rozkaz LPM jest wykonywany w przeciągu trzech cykli CPU po ustawieniu bitu BLBSET i SPMEN wartość bitów bezpieczników (FLB) zostanie zapisana w rejestrze przeznaczenia jak pokazano poniżej. Odnieś się do tabeli 119 na stronie 284 .

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Podobnie odczytując wyższe bity bezpieczników należy wpisać $0003 do wskaźnika Z. Kiedy rozkaz LPM jest wykonywany w przeciągu trzech cykli po zapisie bitów BLBSET i SPMEN w SPMCR. Wartość wysokich bitów FBH zostanie umieszczona w rejestrze przeznaczenia jak pokazano poniżej. Odnieś się do tablicy 118 na stronie 284.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

Przy odczycie rozszerzonych bitów bezpieczników należy załadować $0002 do wskaźnika Z. Kiedy rozkaz LPM jest wykonywany w przeciągu trzech cykli po zapisie bitów BLBSET i SPMEN w SPMCR. Wartość rozszerzonych bitów EFB zostanie umieszczona w rejestrze przeznaczenia jak pokazano poniżej. Odnieś się do tablicy 117 na stronie 283.



Bity blokujące i bezpieczniki, które są zaprogramowane będą odczytywane jako zero. Bity blokujące i bezpieczniki, które nie są zaprogramowane będą odczytywane jako jeden.

- **Zabezpieczenie przed uszkodzeniem Flash**

Podczas okresów kiedy V_{cc} jest niskie program w pamięci Flash może ulec uszkodzeniu ponieważ źródło napięcia jest za słabe aby CPU przeprowadzało poprawne operacje. Rezultaty są takie same jak w przypadku płytki systemu wykorzystującej pamięć Flash i takie same rozwiązanie powinny zostać podjęte.

Uszkodzenie programu we Flash może wystąpić w dwóch przypadkach kiedy napięcie jest za niskie .Po pierwsze, rozkazy zapisu do Flash wymagają minimalnego napięcia by móc działać poprawnie. Po drugie, CPU może wykonywać instrukcje niepoprawnie jeżli zasób napięcia jest za niski.

Uszkodzenia Flash mogą zostać łatwo ominięte poprzez przestrzeganie następujących rozwiązań układowych.:

- Jeśli nie przewiduje się potrzeby aktualizacji programu ładującego należy zaprogramować bity blokujące programowanie tej sekcji pamięci.
- Należy utrzymywać wejście AVR RESET aktywne podczas niewystarczającego zasobu napięcia.. Może to zostać wykonane poprzez odblokowanie wewnętrznego detektora obniżonego napięcia sieciowego (BOD) jeśli napięcie pasuje do poziomu detekcji. W przeciwnym wypadku obwód ochronnego napięcia zewnętrznego może zostać wykorzystany. Kiedy zostanie wykonany reset podczas operacji zapisu , operacja ta zostanie ukończona przy zapewnionym odpowiednim źródle zasilania.
- Należy utrzymywać rdzeń AVR w trybie obniżonego poboru mocy w trakcie obniżonego napięcie V_{cc}. Zapobiegnie to przed dekodowaniem i wykonywaniem instrukcji przez CPU efektywnie chroniąc rejestr SPMCR i pamięć przed niezamierzonymi zapisami.

- **Czasy programowania przy użyciu SPM**

Wzorcowy oscylator RC jest wykorzystywany do czasowych dostępów do Flash. Tabela 111 przedstawia typowy czas programowania dostępów CPU do Flash.

Tabela 111

Symbol	Minimalny czas programowania	Maksymalny czas programowania
Zapis Flash (wymazanie, zapis strony i zapis bitów blokujących przez SPM)	3.7ms	4.5ms

- **Proste przykłady kodów w asemblerze dla programu ładującego**

; procedura ta zapisuje jedną stronę danych z RAM do Flash

; pierwsza lokacja danych w RAM jest wskazywana przez wskaźnik Y
; pierwsza lokacja danych we Flash jest wskazywana przez wskaźnik Z
; nie zawiera obsługi błędów
; musi być umieszczone w przestrzeni programu ładującego
; wykorzystuje rejestry R0 i R1 R16 R17 R24 R25 i R20
; składowanie i odnawianie zawartości rejestrów nie jest zawarte w procedurze
; ilość wykorzystywanych rejestrów może zostać zmniejszona przy jednoczesnym
; zwiększeniu długości kodu
; zakłada się że tablica wektorów przerwań albo jest w sekcji programu ładującego albo
; przerwania są zablokowane

```
.equ  PAGESIZEB = PAGESIZE*2        ;PAGESIZEB is page size in BYTES, not
words
.org SMALLBOOTSTART
Write_page:
  ; page erase
  ldi   spmcrval, (1<<PGERS) | (1<<SPMEN)
  call  Do_spm
  ; re-enable the RWW section
  ldi   spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  call  Do_spm

  ; transfer data from RAM to Flash page buffer
  ldi   looplo, low(PAGESIZEB)   ;init loop variable
  ldi   loophi, high(PAGESIZEB)  ;not required for PAGESIZEB<=256
Wrloop:
  ld    r0, Y+
  ld    r1, Y+
  ldi   spmcrval, (1<<SPMEN)
  call  Do_spm
  adiw  ZH:ZL, 2
  sbiw  loophi:looplo, 2         ;use subi for PAGESIZEB<=256
  brne  Wrloop

  ; execute page write
  subi  ZL, low(PAGESIZEB)       ;restore pointer
  sbci  ZH, high(PAGESIZEB)      ;not required for PAGESIZEB<=256
  ldi   spmcrval, (1<<PGWRT) | (1<<SPMEN)
  call  Do_spm

  ; re-enable the RWW section
  ldi   spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  call  Do_spm
```

```
  ; read back and check, optional
  ldi   looplo, low(PAGESIZEB)   ;init loop variable
  ldi   loophi, high(PAGESIZEB)  ;not required for PAGESIZEB<=256
  subi  YL, low(PAGESIZEB)       ;restore pointer
  sbci  YH, high(PAGESIZEB)
Rdloop:
  lpm   r0, Z+
  ld    r1, Y+
  cpse  r0, r1
  jmp   Error
  sbiw  loophi:looplo, 1         ;use subi for PAGESIZEB<=256
  brne  Rdloop

  ; return to RWW section
  ; verify that RWW section is safe to read
Return:
  lds   temp1, SPMCR
  sbrs  temp1, RWWSB           ; If RWWSB is set, the RWW section is not ready
yet
  ret
  ; re-enable the RWW section
  ldi   spmcrval, (1<<RWWSRE) | (1<<SPMEN)
  call  Do_spm
  rjmp  Return
Do_spm:
  ; check for previous SPM complete
Wait_spm:
  lds   temp1, SPMCR
  sbrc  temp1, SPMEN
  rjmp  Wait_spm
  ; input: spmcrval determines SPM action
  ; disable interrupts if enabled, store status
  in    temp2, SREG
  cli
  ; check that no EEPROM write access is present
Wait_ee:
  sbic  EECR, EEWE
  rjmp  Wait_ee

; SPM timed sequence
sts   SPMCR, spmcrval
spm
; restore SREG (to enable interrupts if originally enabled)
out   SREG, temp2
ret
```

- **Parametry programu ładującego ATmega128**

Tabele 112 do 114 zawierają parametry wykorzystywane w opisie samo programowania się.

Tabela 112 konfiguracja rozmiaru sekcji programu ładującego

							Adres resetu programu ładującego

BOOTSZ1	BOOTSZ0	Rozmiar części programu ładującego	strony	Rozmiar sekcji programów	Sekcja programu ładującego we Flash	Koniec sekcji programów	(początek sekcji programu ładującego)
1	1	512 słów	4	$0000 - &FDFF	$FE00 - $FFFF	$FDFF	$FE00
1	0	24 słów	8	$0000 - $FBFF	$FC00 - $FFFF	$FBFF	$FC00
0	1	2048 słów	16	$0000- $F7FF	$F800 - $FFFF	$F7FF	$F800
0	0	4096 słów	32	$0000 $EFFF	$F000 - $FFFF	$EFFF	$F000

Uwaga: różne konfiguracje BOOTSZ są pokazane na rysunku 132

Tabela 113. limit(1) odczytu podczas zapisu

Sekcja	Strony	adres
Sekcja RWW	480	$0000-$EFFF
Sekcja NRWW	32	$F000-$FFFF

Uwaga: szczegóły dotyczące tych dwóch sekcji znajdują się na stronie 270

Tabela 114 opis zmiennych z rysunku 133

Zmienna		Odpowiadająca jej wartość Z	Opis (2)
PCMSR	15		Najstarszy bit w liczniku programu
PAGEMSB	6		Najstarszy bit wykorzystywany do adresowania słów na stronie (bity PC 6..0)
ZPCMSB		Z16(1)	Bit w rejestrze Z, który odwzorowuje PBMSB ponieważ Z0 jest nie używany, ZPCMSB = PCMSB +1
ZPAGEMSB		Z7	Bit w rejestrze Z, który odwzorowuje PBMSB ponieważ Z0 jest nie używany, ZPAGEMSB = PAGEMSB +1
PCPAGE	PC[15:7]	Z16(1):Z7	Adres strony w liczniku programów – wybiera stronę do wymazania i zapisu
PCWORD	PD[6:0]	Z7:Z1	Adres słowa w liczniku programu – wybiera słowo wypełniające bufor tymczasowy(musi być zerem podczas operacji zapisu)

Uwaga: 1: Rejestr Z ma tyko 16 bitów. 16 bit jest umieszczony w RAMPZ rejestrze w odwzorowaniu I/O
2. Z0 powinno być zerem dla wszystkich rozkazów SPM, wybiera bajt dla instrukcji (E)LPM
3. szczegóły w „Adresowaniu Flash w czasie samo programowania się” na stronie 274.

27. Programowanie pamięci

- **Bity blokujące programy i dane**

ATmega128 zapewnia sześć bitów blokujących, które mogą zostać nie zaprogramowane (1) lub można je zaprogramować (0) w celu uzyskania cech opisanych w tabeli 116. Bity blokujące mogą zostać wymazane tylko przez rozkaz ChipErase

Tabele 115

Bit blokujący	Numer bitu	Opis	Domyślna wartość
	7	-	1(nie zaprogramowany)
	6	-	1(nie zaprogramowany)
BLB12	5	Bit blokujący	1(nie zaprogramowany)
BLB11	4	Bit blokujący	1(nie zaprogramowany)
BLB02	3	Bit blokujący	1(nie zaprogramowany)
BLB01	2	Bit blokujący	1(nie zaprogramowany)
LB2	1	Bit blokujący	1(nie zaprogramowany)
LB1	0	Bit blokujący	1(nie zaprogramowany)

.

Tabela 116

Bity blokujące pamięć			Typ ochrony
Tryb LB	LB2	LB1	
1	1	1	Nie są odblokowane żadne cech pamięci
2	1	0	Późniejsze programowanie Flash i EEPROM jest zablokowane w trybie równoległym i szeregowym SPI i JTAG. Bezpieczniki są zablokowane zarówno w trybie szeregowym jak i równoległym.(1)
3	0	0	Późniejsze programowanie i weryfikacja Flash i EEPROM jest zablokowana w trybie równoległym i szeregowym SPI i JTAG. Bezpieczniki są zablokowane zarówno w trybie szeregowym jak i równoległym.(1)
Tryb BLB0	BLB02	BLB01	
1	1	1	Nie ma żadnych ograniczeń dla SPM i (E)LPM na wchodzenie do sekcji programów
2	1	0	SPM nie może zapisywać do sekcji programów
3	0	0	SPM nie może zapisywać do sekcji programów, (E)LPM wykonywane z sekcji programu ładującego nie może czytać w sekcji programów. Jeśli przerwania są umieszczone w sekcji programu ładującego, są one zablokowane przy wykonywaniu programu z sekcji programów
4	0	1	(E)LPM wykonywane z sekcji programu ładującego nie może czytać z sekcji programów. Jeśli przerwania są umieszczone w sekcji programu ładującego, są one zablokowane przy wykonywaniu programu z sekcji programów
Tryb BLB1	BLB12	BLB11	
1	1	1	Nie ma żadnych ograniczeń przy dostępnie SPM i (E)LPM do sekcji programu ładującego
2	1	0	SPM nie może zapisywać do sekcji programu ładującego

3	0	0	SPM nie może zapisywać do sekcji programu ładującego, (E)LPM wykonane z sekcji programów nie może czytać z sekcji programu ładującego. Jeśli wektory przerwań są w sekcji programów, są zablokowane dla programów wykonywanych z sekcji programu ładującego..
4	0	1	(E)LPM wykonane z sekcji programów nie może czytać z sekcji programu ładującego. Jeśli wektory przerwań są w sekcji programów, są zablokowane dla programów wykonywanych z sekcji programu ładującego..

Uwagi: 1. należy zaprogramować bity blokujące przed bezpiecznikami
2. 1 – nie zaprogramowany, 0 – zaprogramowany

- **Bezpieczniki**

ATmega128 ma trzy bezpieczniki . Tabele 117 – 119 opisują ich funkcjonalność i jak są odzwierciedlane w bajtach bezpieczników. Jeśli są zaprogramowane są odczytywane jako 0.

Tabela 117

Rozszerzone bezpieczniki	Numer bitu	opis	Domyślna wartość
-	7		1
-	6		1
-	5		1
-	4		1
-	3		1
-	2		1
M103C(1)	1	Tryb kompatybilności z ATmega103	0 (zaprogramowany)
WDTON(2)	0	Licznik Watchdog'a zawsze włączony	1 (nie zaprogramowany)

Uwaga: 1.tryb kompatybilności z ATmega103 na stronie 4
2. „rejestr kontroli licznika Watchdog'a „ na stronie 51.

Tabela 118

Starszy bajt bezpiecznika	Numer bitu	Opis	Wartość domyślna
OCDEN(4)	7	Odblokowanie OCD	1 (nie zaprogramowany, zablokowane OCD)
JTAGEN	6	Odblokowanie JTAG	0 (zaprogramowany, odblokowany JTAG)
SPIEN(1)	5	odblokowanie szeregowego programowania i ładowania danych	0 (zaprogramowany, odblokowany SPI)
CKOPT(2)	4	Opcje oscylatora	1 (nie zaprogramowany)
EESAVE	3	Pamięć EEPROM jest chroniony poprzez chipErase	1(nie zaprogramowany, EEPROM nie chroniony)

BOOTSZ1	2	Rozmiar sekcji programu ładującego	0 (zaprogramowany) (3)
BOOTSZ0	1	Rozmiar sekcji programu ładującego	0 (zaprogramowany) (3)
BOOTRST	0	Wybór wektora resetu	1 (nie zaprogramowany)

Uwagi: 1. bezpiecznik SPIEN nie jest dostępny w trybie programowania szeregowego SPI

2.Funkcjonalność bezpiecznika CKOPT zależy od ustawienia bitu CKSEL. „Źródła zegara" na stronie 34.

3. Domyślne wartości bezpieczników BOOTSZ1..0 są maksymalnymi wartościami sekcji programu ładującego. Tabela 112 na stronie 280

4. Nigdy nie wysyłać produktu z zaprogramowanym bezpiecznikiem OCDEN bez względu na ustawienia bitów blokujących i bezpiecznika JTAGEN. Zaprogramowany OCDEN odblokowuje niektóre części systemu zegarowego działającego w trybach snu co może spowodować zwiększenie poboru mocy.

Tabela 119

Młodszy bajt bezpiecznika	Numer bitu	Opis	Wartość domyślna
BODLEVEL	7	Poziom wyzwalający detektor obniżonego napięcia sieciowego	1 (nie zaprogramowany)
BODEN	6	Bit odblokowujący det4ektor obniżonego napięcia sieciowego	1 (nie zaprogramowany, BOD zablokowany)
SUT1	5	Wybieralny czas uruchamiania	1 (nie zaprogramowany)(1)
SUT0	4	Wybieralny czas uruchamiania	0 (Zaprogramowany)(1)
CKSEL3	3	Wybieralne źródło zegara	0 (Zaprogramowany)(2)
CKSEL2	2	Wybieralne źródło zegara	0 (Zaprogramowany)(2)
CKSEL1	1	Wybieralne źródło zegara	0 (Zaprogramowany)(2)
CKSEL0	0	Wybieralne źródło zegara	1 (nie zaprogramowany)(2)

Uwagi: 1. Wartości domyślne SUT1..0 ustawiają maksymalny czas uruchamiania. Szczegóły w tabeli 14 na stronie 38.

2. Wartości domyślne CKSEL3..0 ustawiają oscylator RC na 1MHz. Tabela 6 na stronie 34.

Na status bezpieczników nie ma wpływu ChipErase. Warto zauważyć, że bezpieczniki są zablokowane jeśli bit LB1 jest zaprogramowany. Należy zaprogramować bezpieczniki przed programowaniem bitów blokujących.

- **Zatrzaskiwanie bezpieczników**

Wartość bezpieczników są zatrzaskiwane kiedy urządzenie wchodzi w tryb programowania i zmiany tych wartości nie będą miały żadnego efektu do momentu wyjścia z trybu programowania.. Nie odnosi się to bezpiecznika EESAVE, który będzie miał efekt odkąd został zaprogramowany. Bezpieczniki są również zatrzaskiwane w trybie normalnym i przy zwiększonej mocy(power up).

- **Bity charakterystyczne**

Wszystkie mikrokontrolery Atmela mają trzybajtowy charakterystyczny kod, który charakteryzuje urządzenie. Kod ten może być odczytywany w trybie szeregowym i równoległym, także kiedy urządzenie jest zablokowane. Te trzy bity umieszczone są w oddzielnych przestrzeniach adresowych.

Dla ATmega128 bitami charakterystycznymi są:

- $000:$1E (wskazuje, żę urządzenie zostało wyprodukowane przez Atmela)
- $001:$97 (wskazuje na 128KB pamięci Flash)
- $002:$02 (wskazuje, że jest to urządzenie ATmega128 jeśli zgodne są poprzednie bajty)

- **Bajt wzorcowe**

ATmega128 ma bit ze wzorcową wartością dla oscylatora RC. Bajt ten jest umieszczony w starszym bajcie adresu $000 w przestrzeni adresów charakterystycznych. Podczas resetu ten bajt jest automatycznie wpisywany do rejestru OSCCAL aby zapewnić poprawną częstotliwość wzorcowego oscylatora.

- **Parametry programowania równoległego, odwzorowanie wejść i rozkazy**

Sekcja ta opisuje jak programować równolegle pamięć Flash i sprawdzać poprawność pamięci programów, pamięć danych EEPROM, bity blokujące pamięć, bezpieczniki w ATmega128. Zakłada się, że impulsy trwają najkrócej 250 ns w przeciwnym wypadku jest to zaznaczone.

  - **Nazwy sygnałów**

W tej sekcji niektóre wyprowadzenia ATmega128 są opisywane poprzez nazwy sygnałów opisujące ich funkcjonalność podczas programowania równoległego, rysunek 134 i tabela 120. Wyjścia nie opisane w tej tabeli są nazywane normalnymi nazwami.

XA1/XA0 ustalają rodzaj wykonywanej akcji kiedy na wejściu XTAL1 pojawi się impuls. Tabela122.

Po przyłożeniu impulsu do wejść WR lub OE, rodzaj wykonywanej akcji zależy od załadowanego rozkazu. Tabela 123.

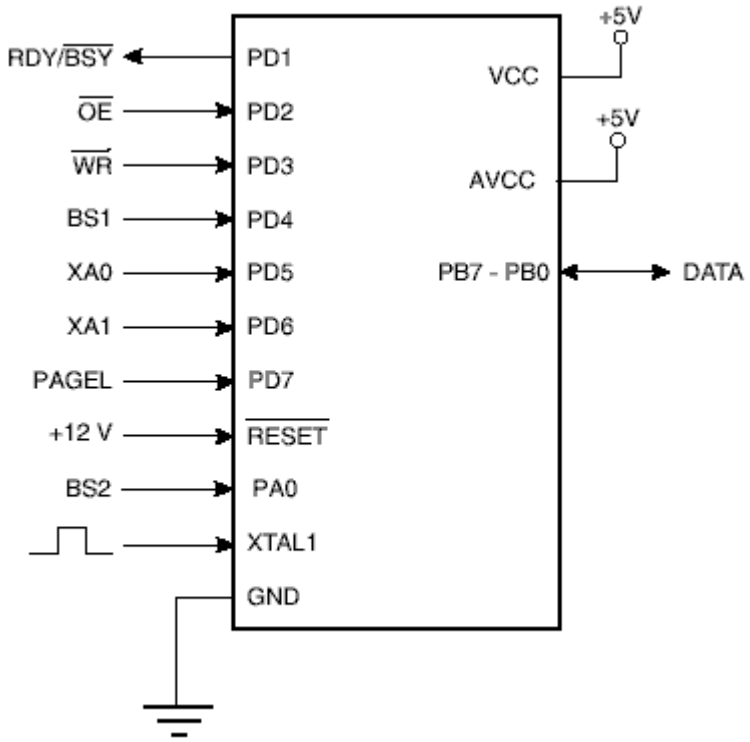


Tabela 120

Nazwa sygnału w trybie programowania	Nazwa wejścia	I/O	Funkcje
RDY/not BSY	PD1	O	): urządzenie jest zajęte programowaniem; 1: urządzenie oczekuje na nowy rozkaz
Not OE	PD2	I	Odblokowanie wyjścia (aktywny stanem niskim)
Not WR	PD3	I	Impuls zapisu (aktywny stanem niskim)
BS1	PD4	I	Wybór bajtu 1(0 – niższy bajt, 1 – wyższy bajt)
XA0	PD5	I	XTAL akcja bitu 0
XA1	PD6	I	XTAL akcja bitu 1
PAGEL	PD8	I	Programowanie pamięci i załadowanie strony danych do EEPROM
BS2	PA0	I	Wybór bajtu 2 (0 – niższy bajt, 1 – drugi wyższy bajt)
DATA	PB7..0	I/O	Dwukierunkowa magistrala danych (wyjście kiedy na OE jest stan niski)

Tabela 121

Wyjście	Symbol	Wartość
PAGEL	Odblokowanie programowania[3]	0
XA1	Odblokowanie programowania[2]	0
XA0	Odblokowanie programowania[1]	0
BS1	Odblokowanie programowania[0]	0

Tabela 122

XA1	XA0	Akcja podęta przy impulsie na XTAL
0	0	Załadowanie adresów Flash lub EEPROM (wysoki bądź niski bajt adresu zależnie od BS1)
0	1	Załadowanie danych (wysoki lub niski bajt dla Flash zależnie od BS1)
1	0	Załadowanie rozkazu
1	1	Nic nie robi

Tabela 123

Słowo sterujące	Wykonany rozkaz
1000 0000	Wymazanie modułu
0100 0000	Zapis bezpieczników
0010 0000	Zapisanie bitów blokujących
0001 0000	Zapis Flash
0001 0001	Zapis EEPROM
0000 1000	Odczyt bajtów charakterystycznych i bitów wzorcowych
0000 0100	Odczyt bezpieczników i bitów blokujących
0000 0010	Odczyt Flash
0000 0011	Odczyt EEPROM

Tabela 124

Rozmiar Flash	Rozmiar strony	PCWORD	Liczba stron	PCPAGE	PCMSB
64K słów (128K bajtów)	128 słów	PC[6:0]	512	PC[15:7]	15

Tabela 125

Rozmiar EEPROM	Rozmiar strony	PCWORD	Liczba stron	PCPAGE	PCMSB
4K bajty	8 bajtów	EEA[2:0]	512	EEA[11:3]	8

- **Równoległe programowanie**

    - **Wejście w tryb programowania**

Następujący algorytm wprowadza urządzenie w tryb programowania:

1. Podać napięcie 4.5 – 5.5 V między wejścia V_{cc} i GRD
2. ustawić wejście RESET na zero (0) przełączyć XTAL najmniej sześć razy
3. ustawić wyjścia prog_enable opisane w tabeli 121 na stronie 287 na 0000 i odczekać najmniej 100ns.
4. podać 11.5 – 12.5V na wejście RESET. Jakakolwiek aktywność na wyjściu prog_enable w przeciągu 100ns po podaniu +12V na RESET spowoduje niepomyślne przejście do trybu programowania

    - **Rozpatrywanie efektywności programowania**

Ładowane rozkazy i adresy pozostają w urządzeniu podczas programowania. W celu zwiększenia efektywności programowania, następujące uwagi powinny zostać uwzględnione:

- Wystarczy tylko raz załadować rozkaz przy wielokrotnym zapisie lub odczycie pamięci
- Ominąć zapis wartości $FF danych, która współzawodniczy z całym EEPROM(chyba że bezpiecznik EESAVE jest zaprogramowany) i Flash po wymazaniu modułu
- Wyższy bajt adresu musi być załadowany tyko raz przy odczycie lub programowaniu nowych 256 słów we Flash i 256 bajtów w EEPROM, Odnosi się to również do odczytu bitów charakterystycznych.

o Wymazanie modułu

Wymazanie modułu wymazuje pamięci Flash, EEPROM i bity blokujące. Bity blokujące nie są kasowane dopóki cała pamięć programu nie zostanie wymazana. Bezpieczniki nie są zmieniane. Wymazanie modułu musi zostać wykonane przed programowaniem Flash.
Uwaga: Pamięć EEPROM jest chroniona przed wymazaniem modułu jeśli bezpiecznik EESAVE jest zaprogramowany.

Ładowanie rozkazu „Wymazania modułu”

1. ustawić XA1, XA0 na 10. To odblokowuje ładowanie rozkazu.
2. ustawić BS1 na 0
3. ustawić DATA na 1000 0000. To jest rozkaz wymazania modułu
4. podać na XTAL dodatni impuls. To ładuje rozkaz.
5. Podać na WR ujemny impuls – 0. To rozpoczyna wymazywanie modułu. RDY/BSY ma wtedy niski poziom
6. Czekać, aż na RDY/BSY pojawi się wysoki poziom przed załadowaniem nowego rozkazu.

o Programowanie Flash

Pamięć Flash jest podzielona na strony, Tabela 123 na stronie 287. Podczas programowania Flash dane są zatrzaskiwane w buforze strony. To pozwala na symultaniczne programowanie całej strony danych. Poniższa procedura opisuje jak programować całą pamięć Flash.

A. Ładowanie rozkazu „Zapis do Flash”

- Ustaw XA1, XA0 na 10. Odblokowanie ładowanie rozkazu.
- Ustaw BS1 na 0
- Ustaw DATA na 0001 0000. To jest rozkaz zapisu Flash.
- Podaj na XTAL dodatni impuls, co załaduje rozkaz.

B. Ładowanie niższego bajtu adresu

- Ustaw XA1, XA0 na 00. Odblokowanie ładowanie adresu.
- Ustaw BS1 na 0. Wybiera niższy adres
- Ustaw DATA = adres niższego bajtu ($00, $FF)
- Podaj na XTAL dodatni impuls, co załaduje adres.

C. Ładowanie niższego bajtu danych

- Ustaw XA1, XA0 na 01. Odblokowanie ładowanie danych.
- Ustaw DATA na niższy adres danych ($00 - $FF)
- Podaj na XTAL dodatni impuls, co załaduje dane.

D. Ładowanie wyższego bajtu danych

- Ustaw BS1 na 1, co wybiera wyższy bajt
- Ustaw XA1, XA0 na 01. Odblokowanie ładowanie rozkazu.
- Ustaw DATA na wyższy bajt ($00 - $FF)
- Podaj na XTAL dodatni impuls, co załaduje bajt

E. Zatrzaśnięcie danych

- Ustawić BS1 na 1 – wybiera wyższy bajt danych
- Podać dodatni impuls na PAGEL. To zatrzaskuje dane. (rysunek 136)

F. Powtarzać kroki od B do E dopóki cały bufor nie jest wypełniony lub zostały załadowane wszystkie dane tej strony

Podczas gdy niższe bity adresów są odwzorowywane do słów na stronie, wyższe bity adresują stronę we Flash. Jest to przedstawione na rysunku 135 na stronie 290. Warto zauważyć, że jeśli jest potrzebne mniej niż 8 bitów adresów (strony < 256) najstarsze bity w młodszym bajcie adresu są wykorzystywane do adresowania strony przy zapisie.

G. Ładowanie wyższego bajtu adresu

- Ustaw XA1, XA0 na 00. Odblokowanie ładowanie adresu.
- Ustaw BS1 na 1, co wybiera wyższy bajt
- Ustaw DATA na wyższy bajt ($00 - $FF)
- Podaj na XTAL dodatni impuls, co załaduje bajt

H. Programowanie strony

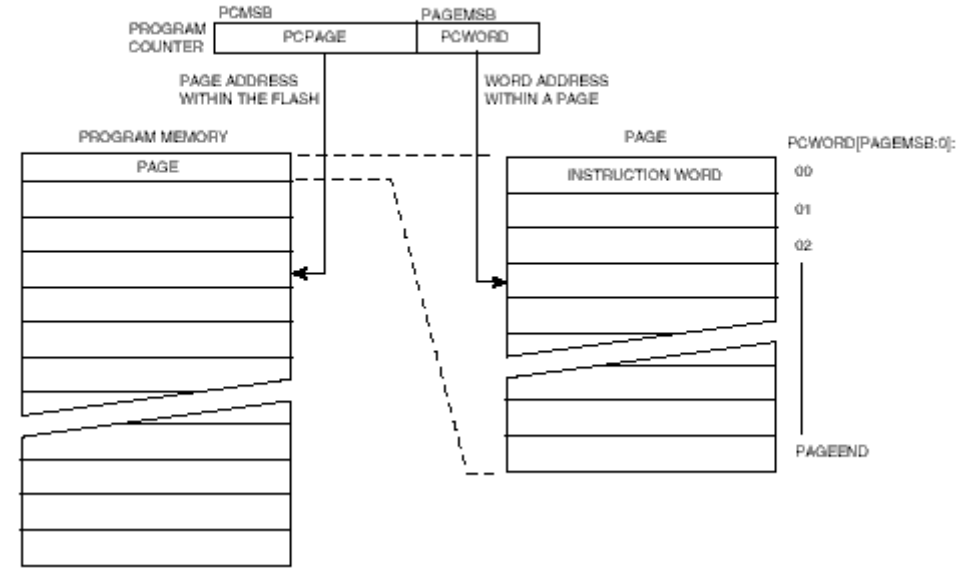
- Ustawić BS1 na 0
- Podać ujemny impuls na WR. To rozpoczyna programowanie całej strony danych. RDY/BSY jest w stanie niskim
- Czekać, aż RDY/BSY będzie w stanie wysokim (rysunek 136)

I. Powtarzać od B do H dopóki cała pamięć Flash nie zostanie zaprogramowana.

J. Zakończenie programowania strony

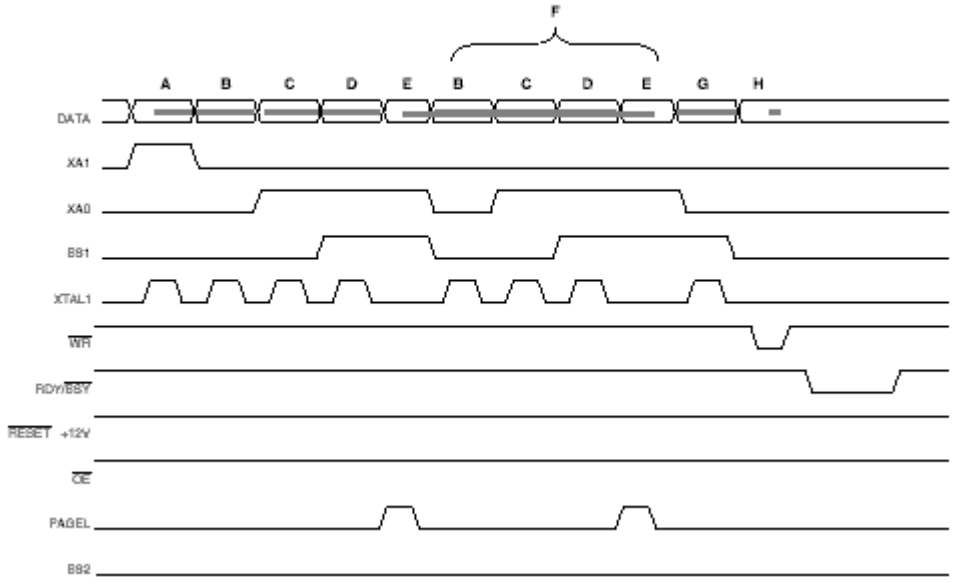
- Ustawić XA1 i XA0 na 10
- Ustawić DATA na 0000 0000. To jest rozkaz -= nie ma żadnej operacji
- Podać dodatniimpuls na XTAL. To załaduje rozkaz jeśli wewnętrzny sygnał zapisu jest zresetowany.

Figure 135. Addressing the Flash which is Organized in Pages



Uwaga: PCPAGE i PCWORD są opisane w tabeli 124 na stronie 287.

Figure 136. Programming the Flash Waveforms



Uwaga: XX nie jest ważne. Litery te odnoszą się do opisu programowania powyżej.

- **Programowanie EEPROM**

EEPROM jest podzielony na strony, tabela 1254 na stronie 287.Podczas programowania EEPROM dane są zatrzaskiwane w buforach strony. To pozwala na symultaniczne programowanie całej strony danych. Poniższa procedura opisuje jak programować całą pamięć EEPROM. (patrz programowanie Flash).

1. A: Ładuj rozkaz 0001 0001
2. G: Ładuj wyższy bajt adresu ($00 - $FF)

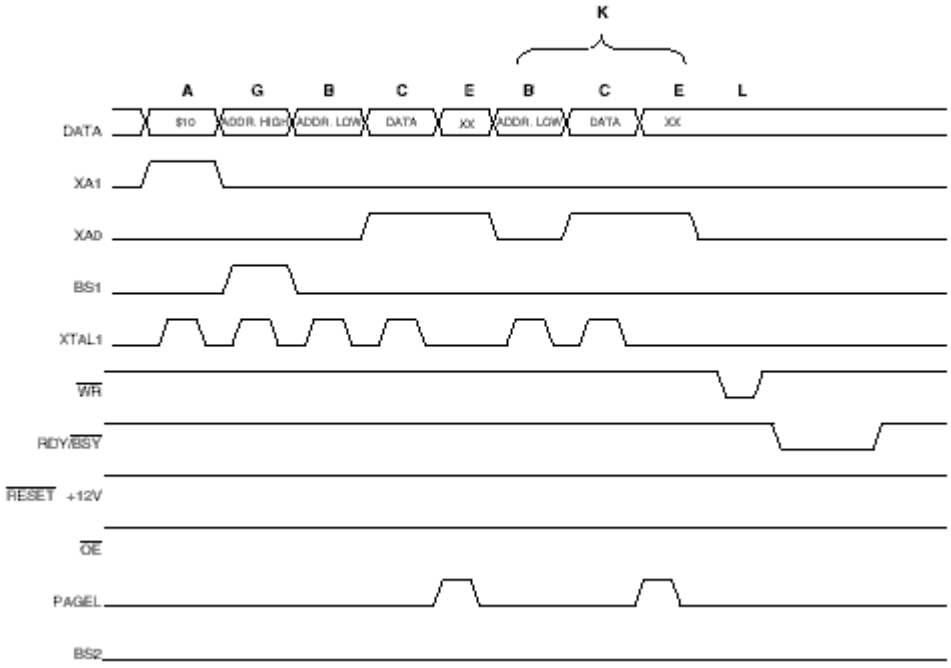
3. B: ładuj niższy bajt adresu ($00-$FF)
4. C: Ładuj dane ($00-$FF)
5. E: Zatrzaśnij dane (podaj na PAGEL dodatni impuls)

K. Powtarzaj od 3 do 5 dopóki cały bufor nie jest wypełniony.

L. Programowanie strony EEPROM

- Ustaw BS1 na 0
- Podaj ujemny impuls na WR. To rozpoczyna programowanie strony. RDY/BSY – poziom niski
- Czekaj na wysoki poziom wyjścia RDY/BSY przed rozpoczęciem programowania następnej strony (rysunek 137)

Figure 137. Programming the EEPROM Waveforms



- **Odczyt Flash**

Algorytm odczytu pamięci Flash :

1. A: Ładuj rozkaz 0000 0010
2. G: Ładuj wyższy bajt adresu ($00-$FF)
3. B: Ładuj niższy bajt adresu ($00-$FF)
4. ustaw OE na 0, BS1 na 0. Niższy bajt słowa danych jest do odczytu na DATA.
5. ustaw BS1 na 1. Na wyjściach DATA jest teraz wyższy bajt danych do odczytu
6. ustaw OE na 1

- **Odczyt EEPROM**

Algorytm odczytu pamięci EEPROM :

1. A: Ładuj rozkaz 0000 0011
2. G: Ładuj wyższy bajt adresu ($00-$FF)
3. B: Ładuj niższy bajt adresu ($00-$FF)
4. ustaw OE na 0, BS1 na 0. Niższy bajt słowa danych jest do odczytu na DATA.
5. ustaw OE na 1

- **Programowanie młodszych bitów bezpieczników**

Algorytm programowania bezpieczników jest następujący

1. A: Ładuj rozkaz 0100 0000
2. C: Ładuj niższy bit danych . Bit n = 0 zaprogramowany, n = 1 nie zaprogramowany
3. ustaw BS1 na 0 i BS2 na 0
4. podaj ujemny impuls na WR i odczekaj aż wyjścia RDY/BSY będą w stanie wysokim

- **Programowanie starszych bitów bezpieczników**

Algorytm programowania bezpieczników jest następujący

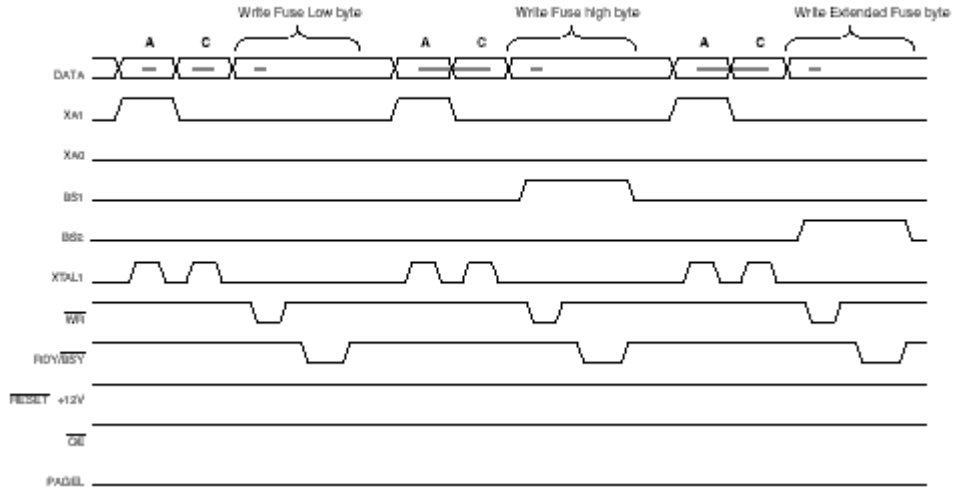
1. A: Ładuj rozkaz 0100 0000
2. C: Ładuj niższy bit danych . Bit n = 0 zaprogramowany, n = 1 nie zaprogramowany
3. ustaw BS1 na 1 i BS2 na 0
4. podaj ujemny impuls na WR i odczekaj aż wyjścia RDY/BSY będą w stanie wysokim
5. ustaw BS1 na 0 aby wybrać niższy bit danych

- **Programowanie bitów rozszerzonych bezpieczników**

Algorytm programowania bitów rozszerzonych bezpieczników jest następujący:

1. A: Ładuj rozkaz 0100 0000
2. C: Ładuj niższy bit danych . Bit n = 0 zaprogramowany, n = 1 nie zaprogramowany
3. ustaw BS1 na 0 i BS2 na 1
4. podaj ujemny impuls na WR i odczekaj aż wyjścia RDY/BSY będą w stanie wysokim
5. ustaw BS2 na 0 aby wybrać niższy bit danych

Figure 138. Programming the Fuses



- **Programowanie bitw blokujących**

Algorytm programowania bitów blokujących jest następujący:

1. A: Ładuj rozkaz 0010 0000
2. C: Ładuj niższy bit danych . Bit n = 0 programuje bit blokujący
3. podaj ujemny impuls na WR i odczekaj aż wyjścia RDY/BSY będą w stanie wysokim

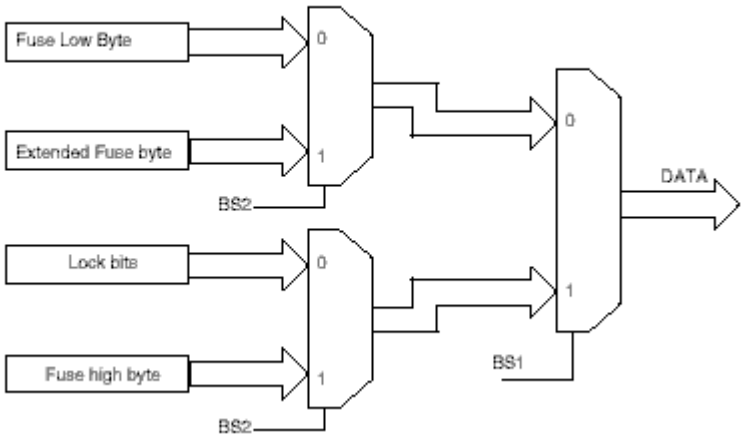
Bity blokujące mogą zostać wyczyszczone tylko przez wymazanie modułu.

- **Odczyt bezpieczników i bitów blokujących**

Algorytm odczytu bitów blokujących i bezpieczników jest następujący:

1. A: Ładuj rozkaz 0000 0100
2. ustaw OE na 0, BS2 na 0 BS1 na 0. Stan niższego bajtu bezpieczników może zostać odczytany z DATA
3. ustaw OE na 0, BS2 na 1 BS1 na 1. Stan wyższego bajtu bezpieczników może zostać odczytany z DATA
4. ustaw OE na 0, BS2 na 1 BS1 na 0. Stan rozszerzonego bajtu bezpieczników może zostać odczytany z DATA
5. ustaw OE na 0, BS2 na 0 BS1 na 1. Stan bitów blokujących może zostać odczytany z DATA
6. ustaw OE na 1

Figure 139. Mapping Between BS1, BS2 and the Fuse- and Lock Bits During Read



- **Odczyt bitów charakterystycznych**

Algorytm odczytu bitów charakterystycznych jest następujący:

1. A: Ładuj rozkaz 0000 1000
2. B: Ładuj młodszy bajt adresu ($00 - $02)
3. ustaw OE na 0, BS1 na 0. Teraz można odczytać wybrane bajty charakterystyczne
4. ustaw OE na 1

- **Odczyt bajtu wzorcowego**

Algorytm odczytu bajtu wzorcowego jest następujący:

1. A: Ładuj rozkaz 0000 1000
2. B: Ładuj niższy bajt adresu
3. ustaw OE na 0, i BS1 na 1. Teraz można odczytać bajty wzorcowe
4. ustaw OE na 1

- **Charakterystyka programowania równoległego**

Figure 140. Parallel Programming Timing, Including some General Timing Requirements

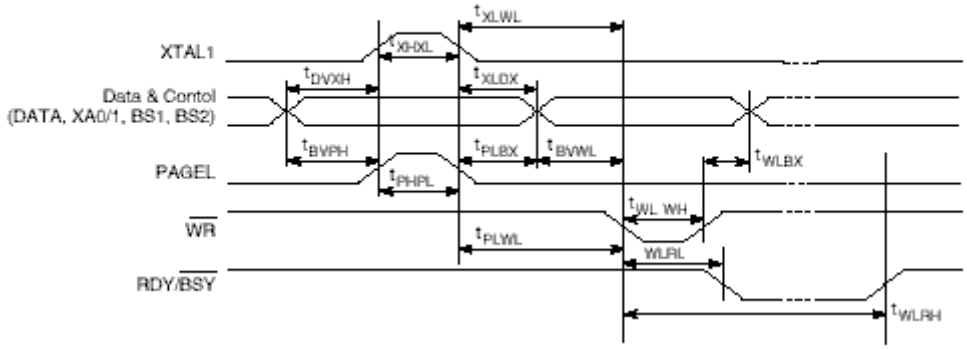
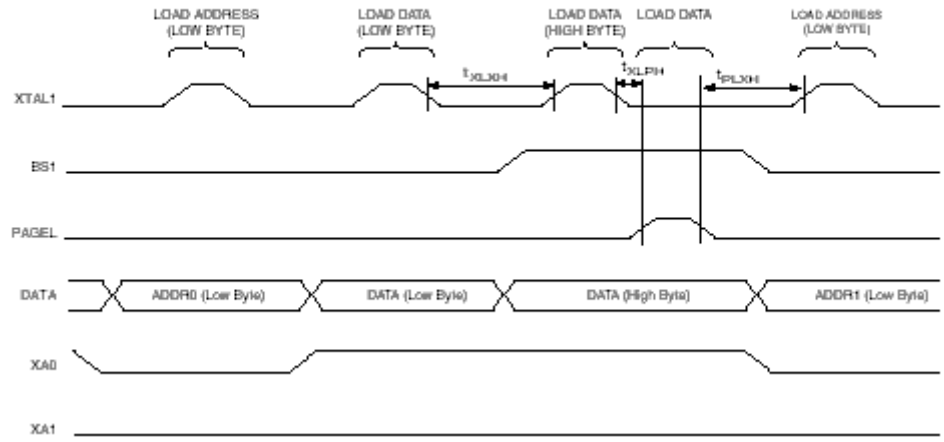
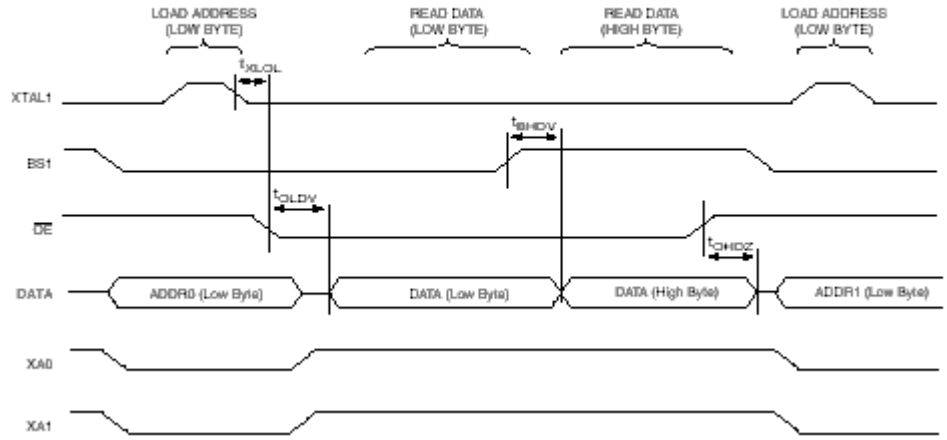


Figure 141. Parallel Programming Timing, Loading Sequence with Timing Requirements



Uwaga: Wymagania czasowe przedstawiona na rysunku 140odnoszą się również do operacji ładowania.

Figure 142. Parallel Programming Timing, Reading Sequence (Within the Same Page) with Timing Requirements



Uwaga: Wymagania czasowe przedstawiona na rysunku 140odnoszą się również do operacji ładowania

Tabela 126

Symbol	Parametr	Minimum	Typ	Maksimum	jednostka
V_{PP}	Odblokowanie programowania napięcia	11.5		12.5	V
I_{PP}	Odblokowanie programowania natężenia			250	Mikro A
t_{DVHX}	Poprawność danych przed podaniem 1 na XTAL1	67			ns
t_{XLXH}	Przejście z niskiego XTAL1 do wysokiego	200			ns
t_{XHXL}	Szerokość impulsu wysokiego XTAL1	150			ns
t_{XLDH}	Wstrzymanie danych i kontroli po ustawieniu XTAl1 na stan niski	67			ns

t_{XLWL}	XTAL1 niski dla WR niskiego	0			ns
t_{XLPH}	XTAL1 niski PAGEL wysoki	0			ns
t_{PLXH}	PAGEN niski do XTAL1 wysoki	150			ns
t_{BVPH}	BS1 wstrzymane po PAGEN wysokie	67			ns
t_{PHPL}	Szerokość wysokiego impulsu PAGEN	150			ns
t_{PLBX}	BS1 wstrzymane po PAGEN niskie	67			ns
t_{WLBX}	BS2/1 wstrzymane po WR niskie	67			ns
t_{PLWL}	PAGEL niskie dla WR niskiego	67			ns
t_{BVWL}	BS1 poprawne do WR niskiego	67			ns
t_{WLWH}	Szerokość impulsu WR w stanie niskim	150			ns
t_{WLRL}	WR niskie dla RDY/BSY niskiego	0		1	mikros
t_{WLRH}	WR niskie dla RDY/BSY wysokiego(1)	3,7		4,5	ms
t_{WLRH_CE}	WR niskie dla RDY/BSY wysokiego dla wymazania strony (2)	7,5		9	ms
t_{XLOL}	XTAL1 niskie dla OE niskiego	0			ns
t_{BVDV}	BS1 poprawne dla poprawnej DATA	0		250	ns
t_{OLDV}	OE niskie dla poprawnej DATA			250	ns
t_{OHDZ}	OE wysokie dla DATA w trzecim stanie			250	Ns

Uwaga: 1. wartość poprawna dla zapisu do Flash, EEPROM , zapisu bitów blokujących i bezpieczników.

2. poprawne dla rozkazu wymazania modułu

- **Szeregowe ładowanie**

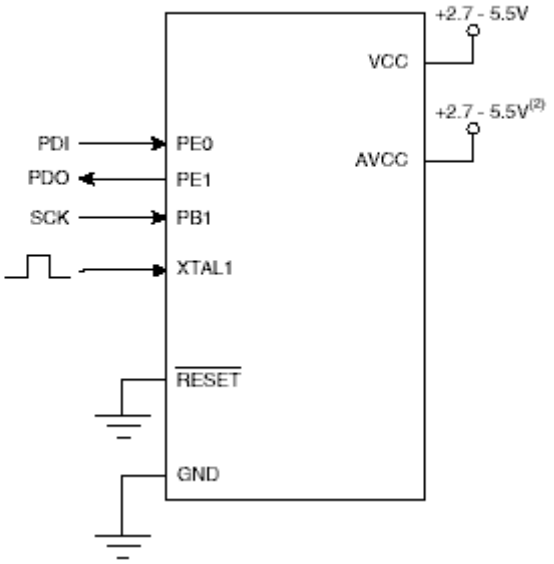
Zarówno tablice pamięci Flash jak i EEPROM mogą być programowane przy wykorzystaniu szeregowej magistrali SPI podczas gdy RESET jest uziemione. Szeregowy interfejs składa się z nóżek SCK, MOSI(wejście) i MISO(wyjście). Po ustawieniu RESET na stan niski instrukcja odblokowująca programowanie musi zostać wykonana przed instrukcjami programowania lub wymazywania. Należy zauważyć, że w tabeli 127 na stronie 296, odwzorowanie wejść dla programowania SPI jest opisane. Nie wszystkie części wykorzystują nóżki przeznaczone dla wewnętrznego interfejsu SPI. Zauważ, że poprzez opis ładowania szeregowego , MOSI i MOSO są wykorzystywane do opisu szeregowych danych wejściowych lub wyjściowych. Dla Atmega128 nóżki te odzwierciedlają PDI i PDO.

- **SPI szeregowe programowanie odwzorowania wejść**

Tabela 127.

Symbol	Nóżki	I/O	Opis
MOSI(PDI)	PE0	I	Szeregowe wejście danych
MOSO(PDO)	PE1	O	Szeregowe wyjście danych
SCK	PB1	I	Szeregowy zegar

Figure 143. SPI Serial Programming and Verify[1]



Uwagi: 1. Jeśli urządzenie jest taktowane przez wewnętrzny oscylator, nie trzeba dołączać źródła zegara do wejść XTAL1

2. V_{CC} – 0.3V < AVCC + 0.3V chociaż AVCC powinno być z przedziału 2.7-5.5V

Podczas programowania EEPROM, automatyczny cykl wymazywania jest wbudowany w operację samo programowania (tylko w trybie szeregowym) i nie ma potrzeby wcześniejszego wykonywania wymazania modułu. Instrukcja wymazania modułu wpisuje zarówno we Flash jak i EEPROM wartość $FF do tablic programów.

Zależnie od bezpiecznika CKSEL, poprawny zegar musi być obecny. Minimalne niskie i wysokie okresy dla wejściowego zegara szeregowego (SCK) są zdefiniowane następująco:

Niski > 2 CPU cykle maszynowe dla f_{ck} < 12 MHz, 3 CPU cykle maszynowe dla f_{ck} >=12MHz
Wysoki: >2 CPU cykle maszynowe dla f_{ck} < 12 MHz, 3 CPU cykle maszynowe dla f_{ck} >=12MHz

- **Algorytm programowania szeregowego SPI**

Wpisując szeregowo dane do ATmega128, dane są taktowane na narastającym zboczu impulsów zegara SCK.

Przy odczycie danych z ATmega128, dane są taktowane na opadającym zboczu impulsów zegara SCK. Obejrzyj rysunek 144 i 145 oraz tabele 145 .

Aby programować i sprawdzać w ATmega128 w trybie programowania szeregowego SPI następująca sekwencja jest zalecana (opis bajtów w tabeli 144):

1. sekwencja zwiększenia mocy

Przyłóż napięcie między V_{cc} i GND podczas gdy RESET i SCK są ustawione na 0. W niektórych systemach programista nie może zagwarantować utrzymywania SCK w stanie niskim podczas

zwiększania mocy. W tym przypadku, na RESET musi zostać podany dodatni impuls o długości minimalnej dwóch cykli maszynowych CPU po tym jak SCK zostało ustawione na 0.
Alternatywą do używania sygnału RESET jest PEN, który powinien być utrzymywany w stanie niskim podczas resetu włączania zasilania kiedy SCK jest ustawiany na 0. W tym przypadku tylko wartość PEN są istotne przy resecie. Jeśli programista nie może zapewnić utrzymania stanu niskiego na SCK, metoda z PEN nie może być wykorzystywana, Urządzenie musi zostać wyłączone w celu rozpoznania normalnych operacji przy użyciu tej metody.

2. Czekaj minimalnie 20 ms i odblokuj SPI szeregowe programowanie przez przesłanie szeregowego rozkazu odblokowującego na wejście MOSI.
3. Rozkaz szeregowego programowania SPI nie będzie działał jeśli komunikacja nie jest zsynchronizowana. Jeśli jest zsynchronizowana drugi bajt ($53) potwierdzi kiedy zostanie sprawdzi trzeci bajt instrukcji odblokowującej programowanie. Czy potwierdzenie jest poprawne czy nie, wszystkie cztery bajty instrukcji muszą zostać przesłane. Jeśli $53 nie jest potwierdzony należ podać na RESET dodatni impuls i rozpocząć nową instrukcję odblokowania programowania.
4. Flash jest programowany strona po stronie. Strona pamięci jest ładowana bajt po bajcie wspierając 7 LSB adresu przez dane warz z rozkazami ładowania programu do pamięci strony. Aby zapewnić poprawne ładowania strony, niższy bajt danych musi być załadowany przed wyższym bajtem pod dany adres. Strona pamięci programu jest składowana poprzez instrukcja zapisu strony pamięci z 9MBS adresu. Jeżeli odpytywanie nie jest wykorzystywane, użytkownik musi odczekać minimum t_{WD_FLASH} przed wydaniem następnej strony. (tabela 128). Wejście do interfejsu szeregowego programowania przed ukończeniem zapisu do Flash może spowodować błąd przy programowaniu.
5. Tablica EEPROM jest programowana bajt po bajcie przez podawanie danych z adresem w odpowiedniej instrukcji zapisu. Pamięć EEPROM jest automatycznie wymazywania nim do danej lokacji zostanie wpisana nowa dana. Jeśli odpytywanie nie jest wykorzystywane, użytkownik musi odczekać minimum t_{WD_EEPROM} przed wydaniem następnego bajtu (tabela 128). W urządzeniu o wymazywalnych modułach, nie trzeba programować żadnych $FF w plikach danych.
6. Każda lokacja pamięci może zostać zweryfikowana poprzez wykorzystanie instrukcji odczytującej, która zwraca zawartość wybranego adresu na wyjściu MISO.
7. Na końcu sesji programowania , RESET może zostać ustawiony w stan wysoki w celu umożliwienia normalnych operacji.
8. Sekwencja wyłączania (jeśli jest potrzebna):
    a. Ustaw RESET na jeden
    b. Wyłącz V_{cc}

- **Odpytywanie danych z Flash**

Kiedy strona we Flash jest programowana, odczytanie adresu z tej strony zwróci wartość $FF. Kiedy urządzenie jest gotowe do zapisu nowej strony, wartości zaprogramowane będą odczytywane poprawnie. Jest to wykorzystywane w celu sprawdzenia kiedy można rozpocząć zapisywanie następnej strony. Zauważ, że cała strona jest zapisywana symultanicznie i jakikolwiek adres z tej strony może zostać wykorzystany w celu odpytania. Odpytywanie danych nie działa przy programowaniu danych na wartości $FF i wtedy użytkownik musi czekać minimalnie t_{WD_FLASH} przed rozpoczęciem programowania następnej strony. Jeśli urządzenie

zawiera we wszystkich lokacjach wartość $FF, programowanie adresów, która mają zawierać $FF może być ominięte. Przejrzyj tabele 128.

- **Odpytywanie danych z EEPROM**

Kiedy nowy bajt jest wpisany i ma zostać zaprogramowany w EEPROM, odczytanie adresu, który jest programowany zwróci wartość $FF. W czasie, kiedy urządzenie jest gotowe do programowania wartość ta zostanie odczytania poprawnie. Wykorzystuje się to do stwierdzenia kiedy można rozpocząć zapis następnego bajtu. Nie będzie to działało dla zapisu wartość $FF, ale użytkownik powinien pamiętać o następujących fakcie: jeśli urządzenie zawiera we wszystkich lokacjach wartość $FF, programowanie adresów, która mają zawierać $FF może zostać ominięte. Nie można tego stosować powtórnym programowania EEPROMU bez wcześniejszego wymazywania modułu. W tym przypadku należy czekać minimum t_{WD_EEPROM} przed rozpoczęciem programowania następnego bajtu. Przejrzyj tabele 128.

Tabela 128

Symbol	Minimalny czas opóźnienia
t_{WD_FLASH}	4,5 ms
t_{WD_EEPROM}	9,0 ms
t_{WD_ERASE}	9,0 ms

Figure 144. .SPI Serial Programming Waveforms

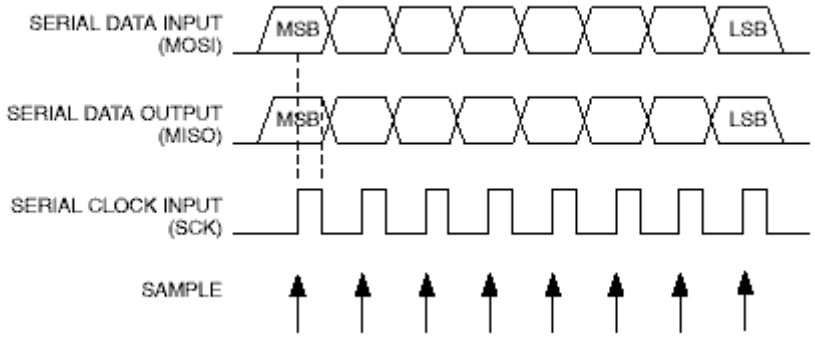


Tabela 129

Table 129. SPI Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable SPI Serial Programming after $\overline{\text{RESET}}$ goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase EEPROM and Flash.
Read Program Memory	0010 **H**000	**aaaa aaaa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from Program memory at word address **a**:**b**.
Load Program Memory Page	0100 **H**000	xxxx xxxx	x**bbb bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to Program Memory page at word address **b**. Data low byte must be loaded before data high byte is applied within the same address.
Write Program Memory Page	0100 1100	**aaaa aaaa**	**b**xxx xxxx	xxxx xxxx	Write Program Memory Page at address **a**:**b**.
Read EEPROM Memory	1010 0000	xxxx **aaaa**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**.
Write EEPROM Memory	1100 0000	xxxx **aaaa**	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a**:**b**.
Read Lock bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read Lock bits. "0" = programmed, "1" = unprogrammed. See Table 115 on page 282 for details.
Write Lock bits	1010 1100	111x xxxx	xxxx xxxx	**11ii iiii**	Write Lock bits. Set bits = "0" to program Lock bits. See Table 115 on page 282 for details.
Read Signature Byte	0011 0000	xxxx xxxx	xxxx xx**bb**	**oooo oooo**	Read Signature Byte **o** at address **b**.
Write Fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = "0" to program, "1" to unprogram. See Table 119 on page 284 for details.
Write Fuse High Bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = "0" to program, "1" to unprogram. See Table 118 on page 284 for details.
Write Extended Fuse bits	1010 1100	1010 0100	xxxx xxxx	xxxx xx**ii**	Set bits = "0" to program, "1" to unprogram. See Table 119 on page 284 for details.
Read Fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse bits. "0" = programmed, "1" = unprogrammed. See Table 119 on page 284 for details.
Read Extendend Fuse bits	0101 0000	0000 1000	xxxx xxxx	**oooo oooo**	Read Extended Fuse bits. "0" = pro-grammed, "1" = unprogrammed. See Table 119 on page 284 for details.
Read Fuse High Bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read Fuse high bits. "0" = pro-grammed, "1" = unprogrammed. See Table 118 on page 284 for details.
Read Calibration Byte	0011 1000	xxxx xxxx	0000 00**bb**	**oooo oooo**	Read Calibration Byte **o** at address **b**.

Uwagi: a = adres starszych bitów
b = adres młodszych bitów
H = 0 – niski bajt = 1 wysoki bajt
O = wyjście danych
I = wejście danych
X = nie należy się tym przejmować

- **Szeregowe programowanie charakterystyk SPI**

Figure 145. SPI Serial Programming Timing

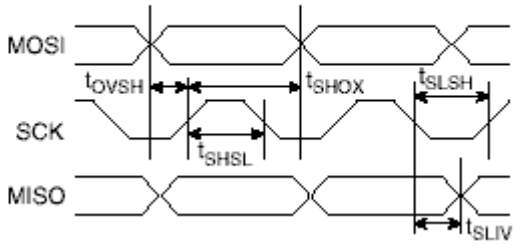


Tabela 130

Symbol	Parametr	Minimum	Typ	Maksimum	Jednostka
$1/t_{CLCL}$	Częstotliwość oscylatora (V_{cc} 2.7 – 5.5V)	0		8	MHz
t_{CLCL}	Okres oscylatora(V_{cc} 2.7 – 5.5V)	125			ns
$1/ t_{CLCL}$	Częstotliwość oscylatora (V_{cc} 4,5 – 5.5V)	0		16	MHz
t_{CLCL}	Okres oscylatora(V_{cc} 4,5– 5.5V)	62,5			ns
t_{SHSL}	SCK szerokość impulsu w stanie wysokim	$2t_{CLCL}$ (1)			ns
t_{SLSH}	SCK szerokość impulsu w stanie niskim	$2t_{CLCL}$ (1)			ns
t_{OVSH}	MOSI uruchamianie dla SCK w stanie wysokim	t_{CLCL}			ns
t_{SHOX}	MOSI zatrzymanie po przejściu SCK na stan niski	$2t_{CLCL}$			ns
t_{SLIV}	SCK w stanie niskim dla MISO poprawnego		15		Ns

Uwagi: 1. $2t_{CLCL}$ dla f_{ck} < 12 MHz, $3t_{CLCL}$ f_{ck} >= 12MHz

- **Programowanie przez interfejs JTAG**

Programowanie poprzez interfejs JTAG wymaga kontroli czterech specyficznych wyjść: TCK, TMS, TDI i TDO. Kontrola resetu i zegara nie jest wymagana.

Aby móc wykorzystywać interfejs JTAG, bezpiecznik JTAGEN musi być zaprogramowany. Urządzenie jest domyślnie ładowane z zaprogramowanym tym bezpiecznikiem. Lub, jeśli bit JDT jest ustawiony, na zewnętrzny reset można podać impuls o stanie niskim. Wtedy, JDT zostanie wyzerowane po dwóch taktach (chip clock) i nóżki JTAG są dostępne dla programowania. Umożliwia to wykorzystanie tych nóżek jako normalnych nóżek portów w trybie normalnym i jako nóżek do programowania przez interfejs JTAG. Technika ta nie może być wykorzystywana kiedy nóżki JTAG są wykorzystywane do skanowania granicy przy debugowniu modułu. W tym przypadku nóżki JTAG muszą wykonywać zamierzoną funkcję.
Zgodnie z definicją w tym arkuszu danych, LBS jest przesuwane do i z pierwsze z wszystkich rejestrów przesuwnych.

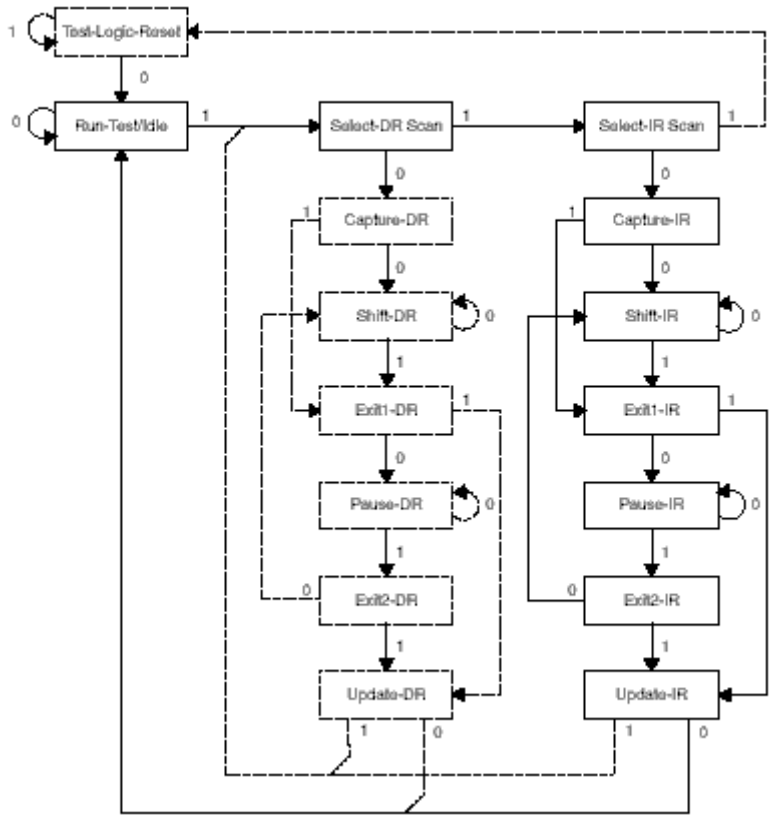
  - **Programowanie specyficznych instrukcji JTAG**

Rejestr instrukcji zajmuje cztery bajty wspierając do 16 instrukcji. Instrukcje JTAG użyteczne przy programowaniu są zamieszczone poniżej.

OPCODE każdej instrukcji jest pokazane za każdą nazwą instrukcji w formacie szesnastkowym. Tekst opisuje, który rejestr danych jest wybrany jako ścieżka między TDI i TDO dla każdej instrukcji.

Stan działania/Idyl kontrolera TAP jest wykorzystywany do generowania wewnętrznego zegara. Może być on również wykorzystywany jako stan idyl między dwoma sekwencjami JTAG. Sekwencja stanów maszyny dla zmiany słowa sterującego jest pokazana na rysunku 146.

Figure 146. State Machine Sequence for Changing the Instruction Word



- **AVR_RESET($C)**

AVR specyfikuje publiczny rozkaz JTAG dla ustawienia urządzenia AVR w tryb resetu lub wyjścia tego urządzenia z trybu resetu. Kontroler TAP nie jest resetowany przez tą instrukcję. Jedno bitowy rejestr resetu jest wybierany jako rejestr danych. Zauważ, że reset będzie aktywny tak długo jak w łańcuch resetu będzie wpisana 1. Wyjście z tego łańcucha nie jest zatrzaskiwane.

Aktywne stany to:
- Shift-DR: Rejestr resetu jest przesuwany przez wejście TCK

- **PROG_ENABLE ($4)**

AVR specyfikuje publiczny rozkaz JTAG odblokowujący programowanie poprzez port JTAG. Szesnastobitowy rejestr odblokowujący programowanie jest wybierany jako rejestr danych. Aktywne stany są następujące:

- Shift-DR: charakterystyka odblokowująca programowanie jest przenoszona do rejestru danych
- Upadte-DR: charakterystyka odblokowująca programowanie jest porównywana z poprawną wartością, i jeśli jest poprawna urządzenie przechodzi do trybu programowania.

- **PROG_COMMANDS($5)**

AVR specyfikuje publiczny rozkaz JTAG służący do wejścia rozkazów programujących przez port JTAG. Piętnastobitowy rejestr rozkazów jest wybrany jako rejestr danych.
Aktywne stany są następujące:

- Capture-DR: rezultat poprzedniej instrukcji jest ładowany do rejestru danych
- Shift-DR: rejestr danych jest przesuwany przez wejście TCK, przesuwając na zewnątrz rezultat poprzedniego rozkazu i wsuwając następny rozkaz
- Update-DR: rozkaz programujący jest podany na wejścia Flash
- Działanie/idyl: generowany jest jeden takt zegara wykonujący podany rozkaz (nie jest zawsze wymagany – patrz tabela 131)

- **PROG_PAGELOAD($6)**

AVR specyfikuje publiczny rozkaz JTAG do bezpośredniego ładowania stron Flash poprzez port JTAG. Rejestr 2048 bitowej wirtualnej strony Flash jest wybrany jako rejestr danych. Jest to wirtualny łańcuch przeszukiwań o długości równej liczbie bitów na stronie Flash. Wewnętrznie rejestr przesuwny jest 8 bitowy. W przeciwieństwie do większości rozkazów JTAG, Update-DR nie jest wykorzystywany do przesyłania danych z rejestru przesuwnego. Dane są automatycznie transmitowane do bufora strony Flash bajt po bajcie w stanie Shift-DR poprzez wewnętrzny stan maszyny. Jedyny aktywny stan:

- Shift-DR: dane strony Flash są przenoszone z TDI przez wejście TCK i automatycznie ładowane w stronie Flash bajt po bajcie.

- **PROG_PAGEREAD($7)**

AVR specyfikuje publiczny rozkaz JTAG odczytujący pełną stronę danyc z Flash poprzez port JTAG. Rejestr 2048 bitowej wirtualnej strony Flash jest wybrany jako rejestr danych. Jest to wirtualny łańcuch przeszukiwań o długości równej liczbie bitów na stronie Flash plus 8. Wewnętrznie rejestr przesuwny jest 8 bitowy. W przeciwieństwie do większości rozkazów JTAG, Update-DR nie jest wykorzystywany do przesyłania danych z rejestru przesuwnego. Dane są automatycznie transmitowane do bufora strony Flash bajt po bajcie w stanie Shift-DR poprzez wewnętrzny stan maszyny. Jedyny aktywny stan:

- Shift-DR: dane strony Flash są automatycznie odczytywane bajt po bajcie i przesuwane na TDO przez TCK. Wejście TDI jest ignorowane.

Uwaga: Instrukcje PROG_PAGELOAD i PROG_PAGEREAD mogą być tylko wykonywane na urządzeniach będących pierwszymi elementami w łańcuch skanowania JTAG. W przeciwnym razie algorytm bajtowy programowania musi zostać wykonany.

- **Rejestr danych**

Rejestr danych jest wybierany przez rozkaz JTAG opisany w sekcji „programowanie specyficznych instrukcji JTAG” na stronie 301. Rejestry danych odpowiadające za operacje programowania to:

- Rejestr resetu
- Rejestr odblokowujący programowanie
- Rejestr rozkazów programowania
- Rejestr ładownia strony Flash
- Rejestr odczytu strony Flash

- **Rejestr resetu**

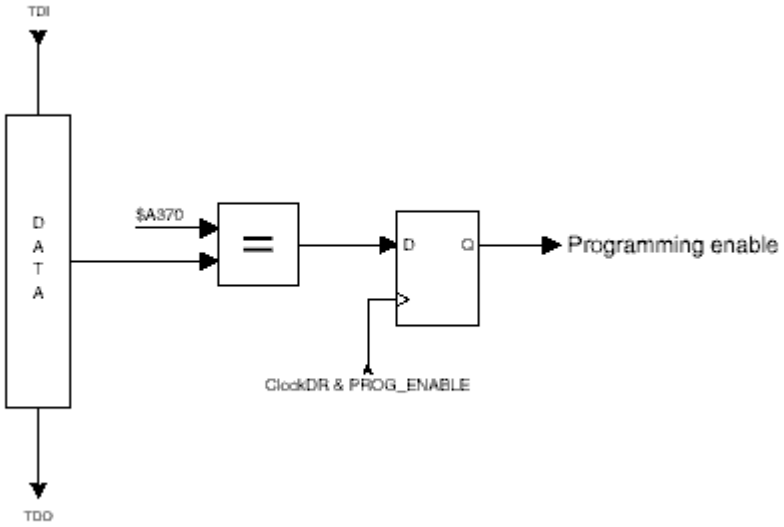
Rejestr resetu jest rejestrem testującym dane wykorzystywanym do resetowania częściowego podczas programowania. Należy zresetować część przed wejściem w tryb programowania.

Wysoka wartość w tym resecie odpowiada ustawieniu resetu zewnętrznego niską wartością. Część ta jest resetowana tak długo jak w rejestrze znajduje się wysoka wartość. W zależności od ustawień bezpiecznika dla opcji zegara, część ta zostanie zretowana poprzez okres resetu (reset timeout period) („Źródła zegarów” na stronie 34)po zwolnieniu rejestru reseetu. Wyjście rejestru danych jest zatrzaskiwane, więc reset będzie następował natychmiastowo, co pokzuje rysunek 122 na stronie 250.

- **Rejestr odblokowujący programowanie**

Rejestr odblokowujący programowanie jest szesnastobitowy. Zawartość tego rejestru jest porównywana z charakterystyką programowania, binarny kod 1010 0011 0111 0000. Kiedy zawartość tego rejestru jest taka sama jak charakterystyka odblokowująca programowanie, programowanie poprzez JTAG jest odblokowane. Rejestr ten jest ustawiany na zero przy resecie uruchamiającym i powinien zawsze być zresetowany przy wyjściu z trybu programowania.

Figure 147. Programming Enable Register



- **Rejestru programowania poleceń**

Rejestr programowania poleceń jest 15 bitowy. Jest on wykorzystywany do szeregowego przesuwania rozkazów i szeregowego wyprowadzania wyników ich wykonania. Zestaw instrukcji JTAG jest przedstawiony w tabeli 131. Sekwencja stanów przy przesuwaniu poleceń programowania jest przedstawiona na rysunku 149.

Figure 148. Programming Command Register

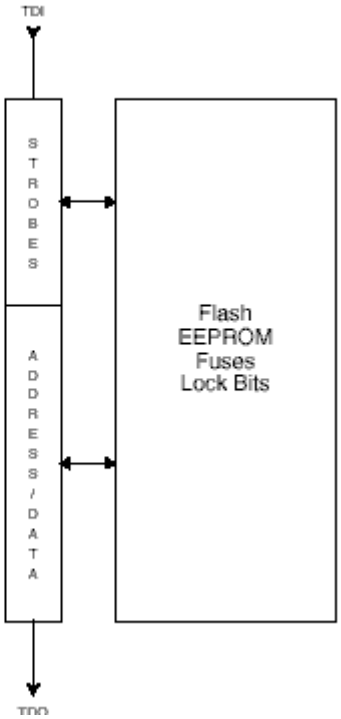


Figure 149. State Machine Sequence for Changing/Reading the Data Word

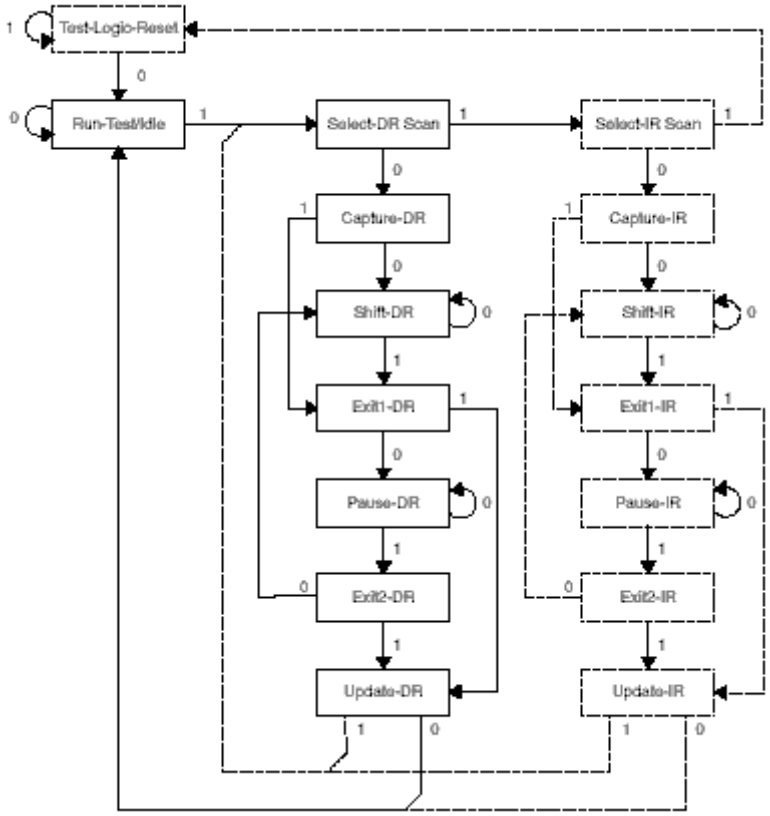


Tabela 131

Table 131. JTAG Programming Instruction Set **a** = address high bits, **b** = address low bits, **H** = 0 - Low byte, 1 - High Byte, **o** = data out, **i** = data in, x = don't care

Instruction	TDI sequence	TDO sequence	Notes
1a. Chip erase	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
1b. Poll for chip erase complete	0110011_10000000	xxxxx**o**x_xxxxxxxx	(2)
2a. Enter Flash Write	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(9)
2c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
2d. Load Data Low Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2e. Load Data High Byte	0010111_**iiiiiiii**	xxxxxxx_xxxxxxxx	
2f. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2g. Write Flash Page	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Poll for Page Write complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
3a. Enter Flash Read	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(9)
3c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
3d. Read Data Low and High Byte	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo**	 low byte high byte
4a. Enter EEPROM Write	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(9)
4c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
4d. Load Data Byte	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	
4e. Latch Data	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Write EEPROM Page	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4g. Poll for Page Write complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)
5a. Enter EEPROM Read	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Load Address High Byte	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	(9)

Instruction	TDI sequence	TDO sequence	Notes
5c. Load Address Low Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
5d. Read Data Byte	0110011_**bbbbbbbb** 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
6a. Enter Fuse Write	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Load Data Low Byte[8]	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	(3)
6c. Write Fuse Extended byte	0111011_00000000 0111001_00000000 0111011_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6d. Poll for Fuse Write complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
6e. Load Data Low Byte[7]	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	(3)
6f. Write Fuse High byte	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6g. Poll for Fuse Write complete	0110111_00000000	xxxxx**o**x_xxxxxxxx	(2)
6h. Load Data Low Byte[7]	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	(3)
6i. Write Fuse Low byte	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6j. Poll for Fuse Write complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)
7a. Enter Lock bit Write	0100011_00100000	xxxxxxx_xxxxxxxx	
7b. Load Data Byte[9]	0010011_11**iiiiii**	xxxxxxx_xxxxxxxx	(4)
7c. Write Lock bits	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
7d. Poll for Lock bit Write complete	0110011_00000000	xxxxx**o**x_xxxxxxxx	(2)
8a. Enter Fuse/Lock bit Read	0100011_00000100	xxxxxxx_xxxxxxxx	
8b. Read Extended Fuse Byte[6]	0111010_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
8c. Read Fuse High Byte[7]	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
8d. Read Fuse Low Byte[8]	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
8e. Read Lock bits[9]	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xx**oooooo**	(5)

Instruction	TDI sequence	TDO sequence	Notes
8f. Read Fuses and Lock bits	0111010_00000000 0111110_00000000 0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_**oooooooo**	(5) fuse ext. byte fuse high byte fuse low byte lock bits
9a. Enter Signature Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
9b. Load Address Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
9c. Read Signature Byte	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
10a. Enter Calibration Byte Read	0100011_00001000	xxxxxxx_xxxxxxxx	
10b. Load Address Byte	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	
10c. Read Calibration Byte	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	
11a. Load No Operation Command	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	

Uwagi: 1. Ta sekwencja rozkazów nie jest wymagana jeśli MSB jest ustawione poprawnie przez poprzednią sekwencję rozkazów

2.powtarzaj, aż o = 1

3. Ustaw bity na 0 aby zaprogramować odpowiadające im bezpieczniki

4.Ustaw bity na 0 aby zaprogramować odpowiadające im bity blokujące

5.0 zaprogramowany – 1 nie zaprogramowany

6. bit odwzorowujący rozszerzony bezpiecznik jest opisany w tabeli 117 na stronie 283

7.bit odwzorowujący wyższy bajt bezpiecznika jest opisany w tabeli 118 na stronie 284

8. bit odwzorowujący niższy bajt bezpiecznika jest opisany w tabeli 118 na stronie 284

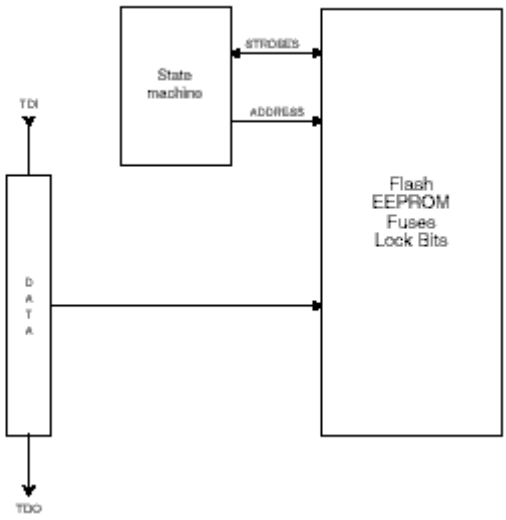
9. bit odwzorowujący bajt bitów blokujących jest opisany w tabeli 115 na stronie 282

10. . adresy przekraczające PSMSB i EEA<SB(tabela 123 i 124) nie są ważne

- **Rejestr ładowania stron wirtualnego Flash**

Rejestr ładowania stron wirtualnego Flash jest wirtualnym łańcuchem o długości odpowiadającej ilość bitów na stronie. Wewnętrznie rejestr przesuwny ma 8 bitów a dane są automatycznie transportowane do bufora stron Flash bajt po bajcie. Przesuń we wszystkich rozkazach na stronie , zaczynając od LSB pierwszej instrukcji i kończąc na MSB. To zapewnia efektywny sposób ładowania całych stron bufora Flash przed zapisaniem strony.

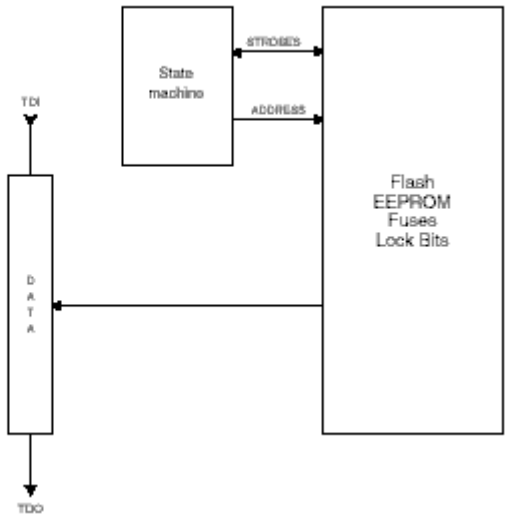
Figure 150. Virtual Flash Page Load Register



- **Rejestr odczytu wirtualnych stron Flash**

Rejestr wirtualnego odczytu stron jest wirtualnym łańcuchem o długości odpowiadającej ilość bitów na stronie zwiększonym o 8. Wewnętrznie rejestr przesuwny ma 8 bitów a dane są automatycznie transportowane do bufora stron Flash bajt po bajcie. Pierwszych osiem cykli jest wykorzystywanych do przetransportowania pierwszego bajtu do rejestru przesuwnego i bity, które pojawią się na wyjściu w tym czasie powinny być ignorowane.Po inicjalizacji dane są przesuwane na zewnątrz zaczynając od LSB pierwszej instrukcji i kończąc na MSB. To zapewnia efektywny sposób odczytu całych stron Flash i sprawdzania poprawności zapisu.

Figure 151. Virtual Flash Page Read Register



- **Algorytm programowania**

Wszystkie odnośniki typów 1a i 1b, i tym podobne, odnoszą się do tabeli 131.

- **Wejście do trybu programowania**

1. Wejdź w instrukcje JTAG VAR_RESET i przesuń 1 w rejestrze resetu
2. Wejdź w rozkaz PROG_ENABLE i przesuń 1010 0011 0111 0000 w rejestrze odblokowującym programowanie

- **Opuszczenie trybu programowania**

1. Wejdź w rozkaz PROG_COMMANDS
2. zablokuj wszystkie instrukcje programujące poprzez wykorzystanie operacji „nie ma żadnej instrukcji" 11a.
3. Wejdź w instrukcje PROG_ENABLE i przesuń 0000 0000 0000 0000 w rejestrze odblokowującym programowanie
4. wejdź w instrukcje AVR_RESET i przesuń 0 w rejestrze resetu

- **Wymazywanie modułów**

1. Wejdź w instrukcje PROG_COMMANDS
2. rozpocznij wymazywanie modułów instrukcją 1a.
3. Sprawdź czy wymazywanie zostało ukończone wykorzystując rozkaz 1b, lub odczekaj t_{WLRH_CE} (tabela na stronie 295)

- **Programowanie Flash**

1. Wejdź w instrukcję PROG_COMMANDS
2. odblokuj zapis Flash wykorzystując rozkaz 2a.
3. Załaduj starszy bajt adresu wykorzystując instrukcję 2b.
4. Załaduj młodszy bajt adresu wykorzystując rozkaz 2c.
5. Załaduj dane przy wykorzystaniu rozkazów 2d, 23 i 2f
6. Powtarzaj kroki 4 i 5 aż zostaną zaprogramowane wszystkie słowa na stronie
7. Zapisz stronę wykorzystując instrukcję 2g.
8. Sprawdź czy zapis został zakończony wykorzystując instrukcję 1h lub odczekaj t_{WLRH}
9. Powtórz kroki 3 do 7 w celu zaprogramowania całej pamięci

Bardziej efektywny transfer danych może być osiągnięty przy wykorzystaniu rozkazu PROG_PAGELOAD:

1. Wejdź w rozkaż JTAG PROG_COMMANDS
2. Odblokuj zapis do Flash wykorzystując instrukcje 2a.
3. Załaduj adres strony wykorzystując instrukcje 2b i 2c. PCWORD (tabela 123 na stronie 287) jest wykorzystywane do adresowania jednej strony i musi zostać zapisane 0.
4. Wejdź q instrukcje JTAG PROG_PAGELOAD
5. Załaduj całą stronę poprzez przesuwanie wszystkich słów instrukcji zaczynając od LSB instrukcji a kończąc na MSB ostatnie instrukcji dla strony.
6. wejdź w rozkaz PROG_COMMANDS
7. zapisz stronę wykorzystując instrukcje zapisu strony 2g
8. Sprawdź czy zapis został zakończony wykorzystując instrukcję 1h lub odczekaj t_{WLRH}

9. Powtórz kroki 3 do 8 w celu zaprogramowania całej pamięci

- **Odczyt Flash**

1. Wejdź w rozkaz PROG_COMMANDS
2. Odblokuj odczyt Flash wykorzystując rozkaz 3a
3. Załaduj adres wykorzystując instrukcje 3b i 3c
4. Odczytaj dane instrukcja 3d
5. Powtarzaj kroki 3 i 4 aż zostanie odczytana cała pamięć

Bardziej efektywny transfer danych może być osiągnięty przy wykorzystaniu rozkazu PROG_PAGEREAD

1. Wejdź w rozkaz PROG_COMMANDS
2. Odblokuj odczyt Flash wykorzystując rozkaz 3a
3. Załaduj adres wykorzystując instrukcje 3b i 3c PCWORD adresuje stronę i musi być zapisane na zero (tabela 123 na stronie 285)
4. Wejdź w rozkaz PROG_PAGEREAD
5. Odczytaj całą stronę przesuwając słowa w stronie zaczynając od LSB a kończąc na instrukcji MSB w stronie. Pamiętaj że 8 pierwszych bitów powinno być zignorowanych.
6. Wejdź do instrukcji PROG_COMMANDS
7. Powtarzaj kroki 3 i 6 aż zostanie odczytana cała pamięć

- **Programowanie EEPROM**

1. Wejdź w instrukcję PROG_COMMANDS
2. odblokuj zapis EEPROM wykorzystując rozkaz 4a.
3. Załaduj starszy bajt adresu wykorzystując instrukcję 4b.
4. Załaduj młodszy bajt adresu wykorzystując rozkaz 4c.
5. Załaduj dane przy wykorzystaniu rozkazów 4d, 4e
6. Powtarzaj kroki 4 i 5 aż zostaną zaprogramowane wszystkie bajty na stronie
7. Zapisz stronę wykorzystując instrukcję 2f.
8. Sprawdź czy zapis został zakończony wykorzystując instrukcję 4g lub odczekaj t_{WLRH}
9. Powtórz kroki 3 do 8 w celu zaprogramowania całej pamięci

Zauważ, że instrukcja PROG_PAGELOAD nie może zostać wykorzystana przy programowaniu EEPROM

- **Odczyt EEPROM**

1. Wejdź w rozkaz PROG_COMMANDS
2. Odblokuj odczyt EEPROM wykorzystując rozkaz 5a
3. Załaduj adres wykorzystując instrukcje 5b i 5c
4. Odczytaj dane instrukcja 5d
5. Powtarzaj kroki 3 i 4 aż zostanie odczytana cała pamięć

Zauważ, że instrukcja PROG_PAGEREAD nie może zostać wykorzystana przy programowaniu EEPROM

- **Programowanie bezpieczników**

1. Wejdź w instrukcję PROG_COMMANDS
2. odblokuj zapis bezpieczników wykorzystując rozkaz 6a.
3. Załaduj bajt danych wykorzystując instrukcję 6b. Wartość bitu 0 odpowiada zaprogramowaniu bezpiecznika.
4. Zapisz rozszerzone bezpieczniki wykorzystując instrukcje 6c
5. Sprawdź czy zapis został zakończony wykorzystując instrukcję 6d lub odczekaj t_{WLRH}
6. Załaduj bajt danych wykorzystując instrukcję 6e. Wartość bitu 0 odpowiada zaprogramowaniu bezpiecznika.
7. Zapisz wyższy bajt bezpiecznika wykorzystując rozkaz 6f
8. Sprawdź czy zapis został zakończony wykorzystując instrukcję 6g lub odczekaj t_{WLRH}
9. Załaduj bajt danych wykorzystując instrukcję 6h. Wartość bitu 0 odpowiada zaprogramowaniu bezpiecznika.
10. Zapisz niższy bajt bezpiecznika wykorzystując rozkaz 6i
11. Sprawdź czy zapis został zakończony wykorzystując instrukcję 6g lub odczekaj t_{WLRH}

- **Programowanie bitów blokujących**

1. Wejdź w instrukcję PROG_COMMANDS
2. odblokuj zapis bitów blokujących wykorzystując rozkaz 7a.
3. Załaduj dane wykorzystując instrukcję 7b. Wartość bitu 0 odpowiada zaprogramowaniu bezpiecznika.
4. Zapisz bit blokujący wykorzystując rozkaz 7c.
5. Sprawdź czy zapis został zakończony wykorzystując instrukcję 7d lub odczekaj t_{WLRH}

- **Odczyt bezpieczników i bitów blokujących**

1. Wejdź w instrukcję PROG_COMMANDS
2. odblokuj odczyt bitów blokujących lub bezpieczników wykorzystując rozkaz 8a.
3. Aby odczytać wszystkie bezpieczniki i bity blokujące wykorzystaj instrukcję 8f.
    a. 8b – rozszerzone bezpieczniki
    b. 8c wysoki bajt bezpieczników
    c. 8d niski bajt bezpieczników
    d. 8e bity blokujące

- **Odczyt bajtów charakterystycznych**

1. Wejdź w instrukcję PROG_COMMANDS
2. odblokuj odczyt bajtów charakterystycznych wykorzystując rozkaz 9a.
3. Załaduj adres $00 wykorzystując instrukcje 9b
4. Odczytaj pierwszy bajt charakterystyczny rozkazem 9c
5. Powtarzaj 3 i 4 z adresami $01 i $02 aby odczytać następne bajty charakterystyczne

- **Odczyt bitu wzorcowego**

1. Wejdź w instrukcję PROG_COMMANDS
2. odblokuj odczyt bajtów wzorcowych wykorzystując rozkaz 10a.
3. Załaduj adres $00 wykorzystując instrukcje 10b
4. Odczytaj bajt wzorcowy rozkazem 10c

CHARAKTERYSTYKA ELEKTRYCZNA

Parametry gwarantowane

Temperatura Pracy ..... -55°C do +125°C
Temperatura Składowania.....-65°C do +150°C

Napięcie na każdej z nóżek z wyjątkiem ~RESET

z uwzględnieniem masy..... od -1.0V do VCC + 0.5V
Maksymalne Napięcie Pracy..... 6.0V
Prąd stały we/wy na styku..... 40.0 mA
Styki masy i napięcia zasilania..... 200.0 mA

PARAMETRY STAŁOPRĄDOWE

T_A = -40°C do 85°C, V_{CC} = 2.7V do 5.5V (jeśli nie jest podana konkretna wartość)

Symbol	*Nazwa*	*Warunki*	**Wart. min**	**Wart. typ**	**Wart. max**	**Jednostka**
V_{IL}	Napięcie wejściowe w stanie niskim	z wyjątkiem styków XTAL1 i ~RESET	-0.5		0.3 V_{CC}	V
V_{IL1}	Napięcie wejściowe w stanie niskim	styk XTAL1, wybrany układ taktujący	-0.5		0.1 V_{CC}	V
V_{IL2}	Napięcie wejściowe w stanie niskim	~RESET	-0.5			V
V_{IH}	Napięcie wejściowe w stanie wysokim	z wyjątkiem styków XTAL1 i ~RESET	0.6 V_{CC}		V_{CC} + 0.5	V
V_{IH1}	Napięcie wejściowe w stanie wysokim	styk XTAL1, wybrany układ taktujący	0.7 V_{CC}		V_{CC} + 0.5	V
V_{IH2}	Napięcie wejściowe w stanie wysokim	~RESET	0.85 V_{CC}		V_{CC} + 0.5	V
V_{OL}	Napięcie wyjściowe w stanie niskim (Porty A,B,C,D, E, F, G)	I_{OL}=20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V		0.7 0.5		V
V_{OH}	Napięcie wyjściowe w stanie wysokim (Porty A,B,C,D)	I_{OH}= -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V		4.0 2.2		V V

I_{IL}	Prąd wejściowy w stanie niskim	Vcc = 5.5V, styk low (wartość bezwzględna)			8.0	µA
I_{IH}	Prąd wejściowy w stanie wysokim	Vcc = 5.5V, styk high (wartość bezwzględna)			8.0	µA
R_{RST}	Reset rezystor podciągający		30		100	~kΩ
R_{PEN}	PEN rezystor podciągający		25		100	~kΩ
R_{PU}	I/O rezystor podciągający		33		122	~kΩ

T_A = -40°C do 85°C, V_{CC} = 2.7V do 5.5V (jeśli nie jest podana konkretna wartość) (ciąg dalszy)

Symbol	*Nazwa*	*Warunki*	**Wart. min**	**Wart. typ**	**Wart. max**	**Jednostka**
I_{CC}	Prąd zasilania	Czynny 4 MHz, V_{CC} = 3V (ATmega128L)			5	mA
		Czynny 8 MHz, V_{CC} = 5V (ATmega128)			20	mA
		Jałowy 4 MHz, V_{CC} = 3V (ATmega128L)			2	mA
		Jałowy 8 MHz, V_{CC} = 5V (ATmega128)			12	mA
	Stan wyłączenia (Power-down)	WDT wł., V_{CC} = 3V		<25	40	µA
		WDT wył., V_{CC} = 3V		<10	25	µA
V_{ACIO}	Wejściowe napięcie niezrównoważenia komparatora analogowego	V_{CC} = 5V Vin = 0.5V_{CC}			40	mV
I_{ACLK}	Wejściowy upływ prądu komparatora analogowego	V_{CC} = 5V Vin = 0.5V_{CC}	-50		50	nA
t_{ACID}	Opóźnienie inicjalizacji komparatora analogowego	V_{CC} = 2.7 V V_{CC} = 5.0 V		750 500		ns
t_{ACID}	Opóźnienie propagacji komparatora analogowego	V_{CC} = 2.7 V V_{CC} = 5.0 V		750 500		ns

UKŁAD TAKTUJĄCY

Rys. 152. **Czasy propagacji sygnału zegarowego**

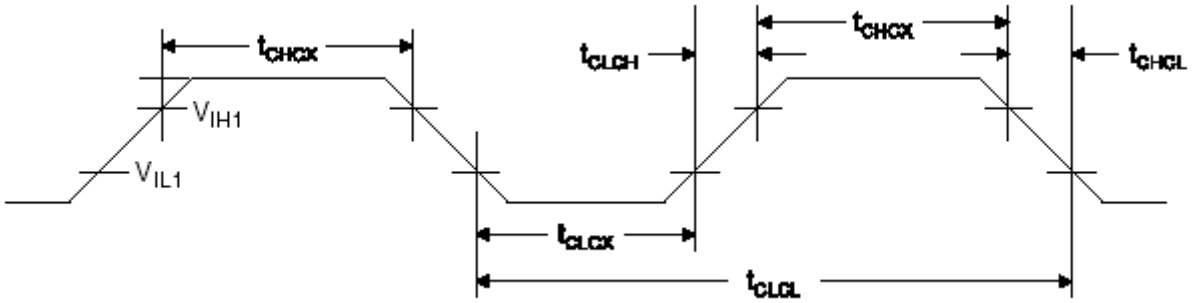


Tabela 132. Układ taktujący

Symbol	Nazwa	VCC = 2.7V do 5.5V		VCC = 4.5V do 5.5V		Jednostka
		Min	**Max**	**Min**	**Max**	
1/tCLCL	Częstotliwość oscylatora	0	8	0	16	MHz
tCLCL	Czas trwania sygnału	125		62.5		ns
tCHCX	Czas trwania Stanu wysokiego	50		25		ns
tCLCX	Czas trwania stanu niskiego	50		25		ns
tCLCH	Czas narastania		1.6		0.5	µs
tCHCL	Czas opadania		1.6		0.5	µs

Tabela 133. Typowe wartości częstotliwości oscylatora RC

R [kΩ]	C [pF]	f
100	70	TBD
31.5	20	TBD
6.5	20	TBD

Uwaga: R powinien być z zakresu 3 k- 100 k, C powinna wynosić przynajmniej 20 pF.

CHARAKTERYSTYKA DWUPRZEWODOWEGO INTERFEJSU SZEREGOWEGO

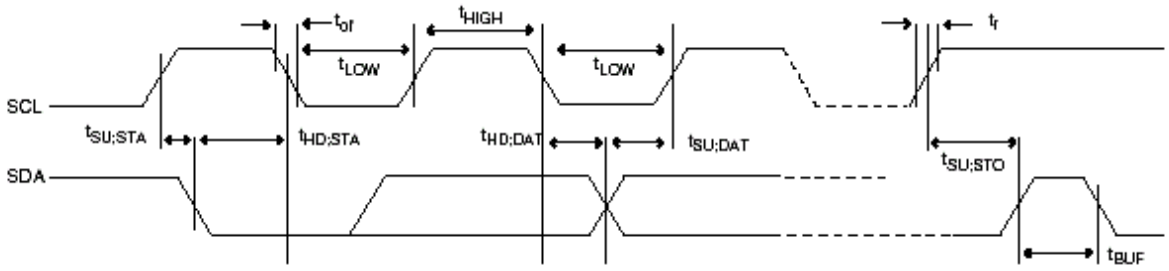
Tabela 134 opisuje wymagania dla układów podłączonych do dwuprzewodowej magistrali szeregowej. ATmega128 dwuprzewodowy interfejs szeregowy spełnia lub przekracza te wymagania.
Oznaczenia czasowe odnoszą się do rys. 153.

Tabela 134. Wymagania dwuprzewodowej magistrali szeregowej

Symbol	Nazwa	Warunki	Min	Max	Jednostka
V_{IL}	Napięcie wejściowe w stanie niskim		-0.5	0.33 V_{CC}	V
V_{IH}	Napięcie wejściowe w stanie wysokim		0.7 V_{CC}	$V_{CC + 0.5}$	V
V_{hys}	Histereza Smitt`a Trigger Inputs		0.05 V_{CC}	-	V
V_{OL}	Napięcie wyjściowe w stanie niskim	3 mA prąd wsteczny	0		V
t_r	Czas narastania dla SDA i SCL		20 + 0.1C_b	300	ns
t_{of}	Czas opadania na wyjściu od V_{IHmin} do V_{ILmax}	10 pF < C_b < 400 pF	20 + 0.1Cb	250	ns
t_{SP}	Spikes Suppressed by Input Filter		0	50	ns
I_i	Prąd we. Na każdym z we/wy styków	0.1 V_{CC} < V_i < 0.9 V_{CC}	-10	10	µA
C_i	Pojemność każdego styku		-	10	pF
f_{SCL}	Częstotliwość zegara SCL	f_{CK} >max(16f_{SCL}, 250kHz)	0	400	kHz
Rp	Wartość rezystora podciągającego	f_{SCL} <= 100 kHz	(V_{CC} – 0.4V) /3mA	1000 ns *1/C_b	Ω
		f_{SCL} > 100 kHz	(V_{CC} – 0.4V) /3mA	300 ns *1/C_b	Ω
$t_{HD;STA}$	Czas przetrzymywania	f_{SCL} <= 100 kHz	4.0	-	µs
		f_{SCL} > 100 kHz	0.6	-	µs
t_{LOW}	Czas trwania stanu niskiego dla zegara SCL	f_{SCL} <= 100 kHz	4.7	-	µs
		f_{SCL} > 100 kHz	1.3	-	µs
t_{HIGH}	Czas trwania stanu wysokiego dla zegara SCL	f_{SCL} <= 100 kHz	4.0	-	µs
		f_{SCL} > 100 kHz	0.6	-	µs
$t_{SU;STA}$	Czas ustalania	f_{SCL} <= 100 kHz	4.7	-	µs
		f_{SCL} > 100 kHz	0.6	-	µs
$t_{HD;DAT}$	Czas przetrzymywania	f_{SCL} <= 100 kHz	0	3.45	µs
		f_{SCL} > 100 kHz	0	0.9	µs
$t_{SU;DAT}$	Czas ustalania danych	f_{SCL} <= 100 kHz	250	-	ns

		f_{SCL} > 100 kHz	100	-	ns
$t_{SU;STO}$	Czas ustalania dla STOP	f_{SCL} <= 100 kHz	4.0	-	µs
		f_{SCL} > 100 kHz	0.6	-	µs
t_{BUF}	Czas zwolnienia magistrali pomiędzy START i STOP	f_{SCL} <= 100 kHz	4.7	-	µs

Rys. 153. Synchronizacja dwuliniowej szeregowej magistrali

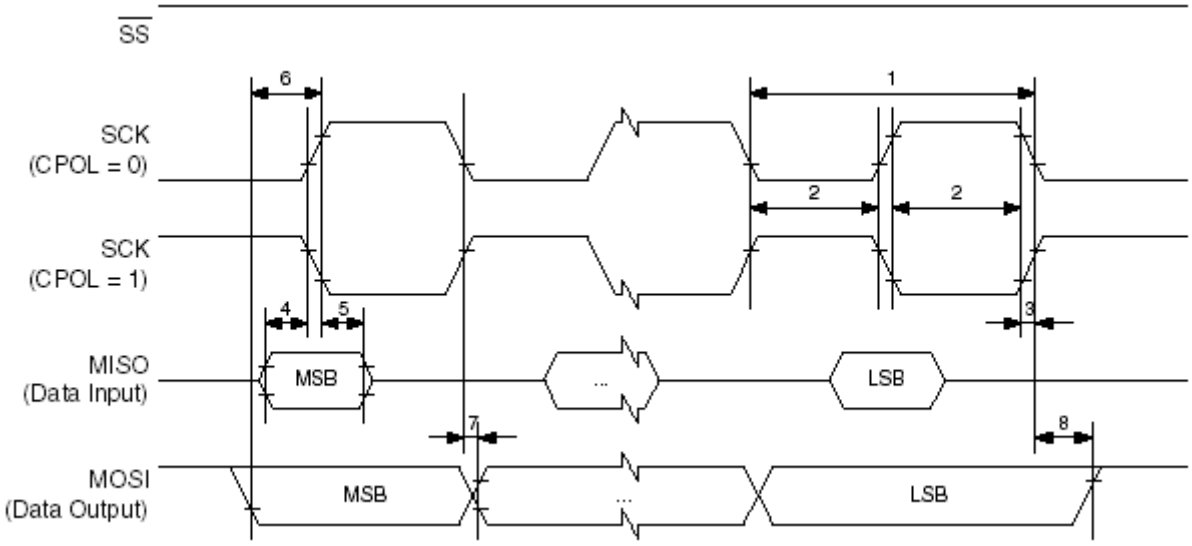


Więcej szczegółów na rysunkach **154** i **155**

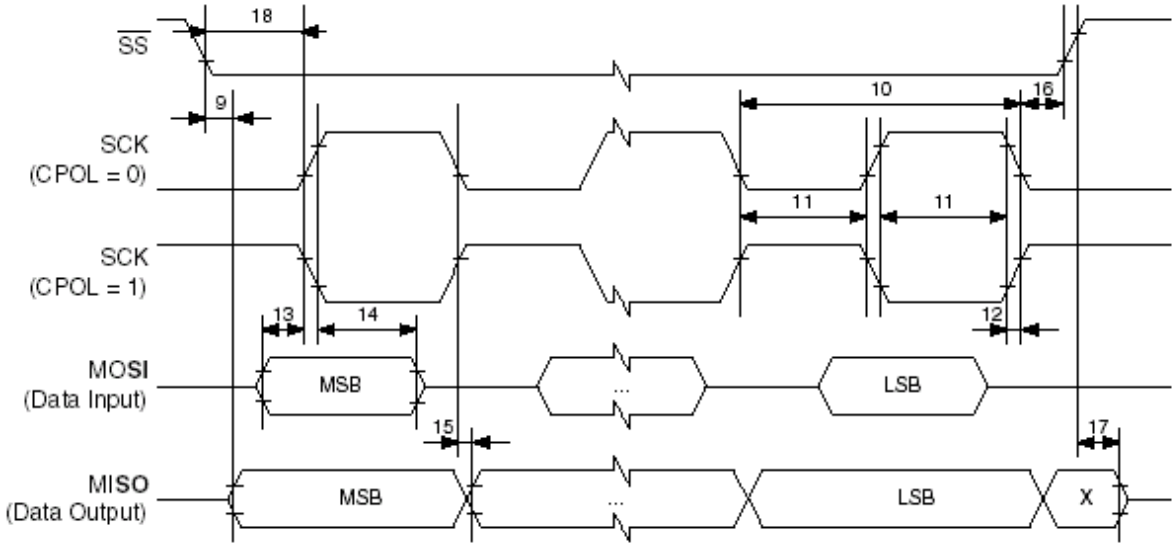
Tabela 135. SPI – parametry synchronizacji

	OPIS	**Tryb**	**Min**	**Typ**	**Max**	
1	SCK okres	Master		patrz tab. 72		ns
2	SCK wysoki/niski	Master		50% cyklu obowiązkowego		
3	Czas narastania/opadania	Master		TBD		
4	ustalanie	Master		10		
5	wstrzymanie	Master		10		
6	Wyjście do SCK	Master		0.5 * t_{sck}		
7	SCK do wyjścia	Master		10		
8	SCK do wyjścia wysoki	Master		10		
9	SS niski do wyjścia	Slave		15		
10	SCK okres	Slave	4*t_{ck}			
11	SCK wysoki/niski	Slave	2*t_{ck}			
12	Czas narastania/opadania	Slave		TBD		
13	ustalanie	Slave	10			
14	wstrzymanie	Slave	10			
15	SCK do wyjścia	Slave		15		
16	SCK do ~SS wysoki	Slave	20			
17	~SS wysoki do trójstanowy	Slave		10		
18	SS niski do SCK	Slave	20			

Rys. 154. SPI Interfejs - wymagania synchronizacji (tryb master)



Rys. 155. SPI Interfejs - wymagania synchronizacji (tryb slave)



CHARAKTERYSTYKA KONWERTERA ANALOGOWO-CYFROWEGO(ADC) – WSTĘPNE DANE

Symbol	Opis	Warunki	Min (1)	Typ (1)	Max (1)	Jednostka
	rozdzielczość	Konwersja niesymetryczna		10		Bity
		Konwersja różnicowa przyrost = 1x lub 20x		8		
		Konwersja różnicowa przyrost = 200x		7		
	Bezwzględna dokładność	Niesymetryczna konwersja V_{REF} = 4V ADC zegar = 200 kHz ADHSM = 0		1	TBD	LSB
		Konwersja niesymetryczna V_{REF} = 4V ADC zegar = 1 MHz ADHSM = 1		TBD	TBD	
	Integralna nieliniowość	V_{REF} = 4V		0.5		
	Różnicowa nieliniowość	V_{REF} = 4V		0.5		
	Błąd Zero (Offset)	V_{REF} = 4V		1		
	Czas konwersji	Konwersja własna ADHSM = 0	65		260	µs
		Konwersja własna ADHSM = 1	65		TBD	
	Częstotliwość zegara	ADHSM=0	50		200	kHz
		ADHSM=1	50		TBD	
AVCC	Analogowe napięcie zasilania		V_{CC} - 0.3 (2)		V_{CC} + 0.3 (3)	V
V_{REF}	Napięcie odniesienia	Konwersja niesymetryczna	2.0		AV_{CC}	
		Konwersja różnicowa	2.0		AV_{CC}-0.2	
	Napięcie wejściowe	Kanały niesymetryczne	GND		V_{REF}	

V_{IN}		Kanały Różnicowe	TBD		TBD	
	Szerokość pasma na wejściu	Kanały niesymetryczne		TBD		kHz
		Kanały Różnicowe		4		
V_{INT}	Wew. napięcie odniesienia		2.3	2.56	2.7	V
R_{REF}	Rezystancja wejścia odniesienia		TBD	TBD	TBD	kΩ
R_{AIN}	Rezystancja analogowego wejścia			TBD		MΩ
I_{HSM}	Zwiększony pobór prądu w trybie szybkiej pracy (ADHSM=1)			TBD		µA

Uwagi:
1. Rzeczywistymi wartościami są TBD.
2. Minimum dla AVCC to 2.7 V.
3. Maximum dla AVCC to 5.5 V.

Synchronizacja pamięci zewnętrznej

Tabela 137. Charakterystyka pamięci zewnętrznej, 4.5 - 5.5 V, brak czasu oczekiwania

	Symbol	**Nazwa**	**8MHZ Oscylator**				*Jednostka*
			Min	**Max**	**Min**	**Max**	
0	$1/t_{CLCL}$				0.0		MHz
1	t_{LHLL}		115		$1.0t_{CLCL}$-10		ns
2	t_{AVLL}		57.5		$0.5t_{CLCL}$-5		ns
3a	t_{LLAX_ST}		5		5		ns
3b	t_{LLAX_LD}		5		5		ns
4	t_{AVLLC}		57.5		$0.5t_{CLCL}$-5		ns
5	t_{AVRL}		115		$1.0t_{CLCL}$-10		ns
6	t_{AVWL}		115		$1.0t_{CLCL}$-10		ns
7	t_{LLWL}		47.5	67.5	$0.5t_{CLCL}$-15	$0.5t_{CLCL}$+5	ns
8	t_{LLRL}		47.5	67.5	$0.5t_{CLCL}$-15	$0.5t_{CLCL}$+5	ns
9	t_{DVRH}		40		40		ns
10	t_{RLDV}			75		$1.0t_{CLCL}$-50	ns
11	t_{RHDX}		0		0		ns
12	t_{RLRH}		115		$1.0t_{CLCL}$-10		ns

13	t_{DVWL}		42.5		$0.5t_{CLCL}$-20		ns
14	t_{WHDX}		115		$1.0t_{CLCL}$-10		ns
15	t_{DVWH}		125		$1.0t_{CLCL}$		ns
16	t_{WLWH}		115		$1.0t_{CLCL}$-10		ns

Tabela 138. Charakterystyka pamięci zewnętrznej, 4.5 - 5.5 V, 1 cykl czasu oczekiwania

	Symbol	**Nazwa**	**8MHZ Oscylator**				*Jednostka*
			Min	**Max**	**Min**	**Max**	
0	$1/t_{CLCL}$	Częstotliwość oscylatora			0.0	16	MHz
10	t_{RLDV}	Read Low do Data Valid		200		$2.0t_{CLCL}$-50	ns
12	t_{RLRH}	Szerokość impulsu RD	240		$2.0t_{CLCL}$-10		ns
15	t_{DVWH}	Data Valid do WR High	240		$2.0t_{CLCL}$		ns
16	t_{WLWH}	Szerokość impulsu WR	240		$2.0t_{CLCL}$-10		ns

Tabela 139. Charakterystyka pamięci zewnętrznej, 4.5 - 5.5 V, SRWn1 = 1, SRWn0 = 0

	Symbol	**Nazwa**	**8MHZ Oscylator**				*Jednostka*
			Min	**Max**	**Min**	**Max**	
0	$1/t_{CLCL}$	Częstotliwość oscylatora			0.0	16	MHz
10	t_{RLDV}	Read Low do Data Valid		325		$3.0t_{CLCL}$-50	ns
12	t_{RLRH}	Szerokość impulsu RD	365		$3.0t_{CLCL}$-10		ns
15	t_{DVWH}	Data Valid to WR High	375		$3.0t_{CLCL}$		ns
16	t_{WLWH}	Szerokość impulsu WR	365		$3.0t_{CLCL}$-10		ns

Tabela 140. Charakterystyka pamięci zewnętrznej, 4.5 - 5.5 V, SRWn1 = 1, SRWn0 = 1

	Symbol	**Nazwa**	**8MHZ Oscylator**				*Jednostka*
			Min	**Max**	**Min**	**Max**	
0	$1/t_{CLCL}$	Częstotliwość oscylatora			0.0	16	MHz
10	t_{RLDV}	Read Low to Data Valid		325		$3.0t_{CLCL}$-50	ns

12	t_{RLRH}	Szerokość impulsu RD	365		$3.0t_{CLCL}$-10		ns
14	t_{WHDX}	Data Hold po WR High	240		$2.0t_{CLCL}$-10		ns
15	t_{DVWH}	Data Valid to WR High	375		$3.0t_{CLCL}$		ns
16	t_{WLWH}	Szerokość impulsu WR	365		$3.0t_{CLCL}$-10		ns

Tabela 141. Charakterystyka pamięci zewnętrznej, 2.7 - 5.5 V, brak czasu oczekiwania

	Symbol	**Nazwa**	**8MHZ Oscylator**				*Jednostka*
			Min	**Max**	**Min**	**Max**	
0	$1/t_{CLCL}$	Częstotliwość oscylatora			0.0		MHz
1	t_{LHLL}	Szerokość impulsu ALE	235		t_{CLCL}-15		ns
2	t_{AVLL}	Address Valid A do ALE low	115		$0.5t_{CLCL}$-10		ns
3a	$t_{LLAX\_}$	Address Hold po ALE Low, dostęp do zapisu	5		5		ns
3b	$t_{LLAX\_}$	Address Hold po ALE Low, Dostęp do odczytu	5		5		ns
4	t_{AVLLC}	Address Valid C do ALE Low	115		$0.5t_{CLCL}$-10		ns
5	t_{AVRL}	Address Valid do RD Low	235		$1.0t_{CLCL}$-15		ns
6	t_{AVWL}	Address Valid do WR Low	235		$1.0t_{CLCL}$-15		ns
7	t_{LLWL}	ALE Low do WR Low	115	130	$0.5t_{CLCL}$-10	$0.5t_{CLCL}$+	s
8	t_{LLRL}	ALE Low do RD Low	115	130	$0.5t_{CLCL}$-10	$0.5t_{CLCL}$+	ns
9	t_{DVRH}	Data Setup do RD High	45		45		ns
10	t_{RLDV}	Read Low do Data Valid		190		$1.0t_{CLCL}$-60	ns
11	t_{RHDX}	Data Hold po	0		0		ns

		RD High					
12	t_{RLRH}	Szerokość impulsu RD	235		$1.0t_{CLCL}$-15		ns
13	t_{DVWL}	Data Setup do WR Low	105		$0.5t_{CLCL}$-20		ns
14	t_{WHDX}	Data Hold do WR High	235		$1.0t_{CLCL}$-15		ns
15	t_{DVWH}	Data Valid do WR High	250		$1.0t_{CLCL}$		ns
16	t_{WLWH}	Szerokość impulsu WR	235		$1.0t_{CLCL}$-15		ns

Tabela 142. Charakterystyka pamięci zewnętrznej, 2.7 - 5.5 Volts, SRWn1 = 0, SRWn0 = 1

	Symbol	**Nazwa**	**8MHZ Oscylator**		**ZMIENNY OSCYLATOR**		*Jednostka*
			Min	**Max**	**Min**	**Max**	
0	$1/t_{CLCL}$	Częstotliwość oscylatora			0.0	8	MHz
10	t_{RLDV}	Read Low do Data Valid		440		$2.0t_{CLCL}$-60	ns
12	t_{RLRH}	Szerokość impulsu RD	485		$2.0t_{CLCL}$-15		ns
15	t_{DVWH}	Data Valid do WR High	500		$2.0t_{CLCL}$		ns
16	t_{WLWH}	Szerokość impulsu WR	485		$2.0t_{CLCL}$-15		ns

Tabela 143. Charakterystyka pamięci zewnętrznej, 2.7 - 5.5 V, SRWn1 = 1, SRWn0 = 0

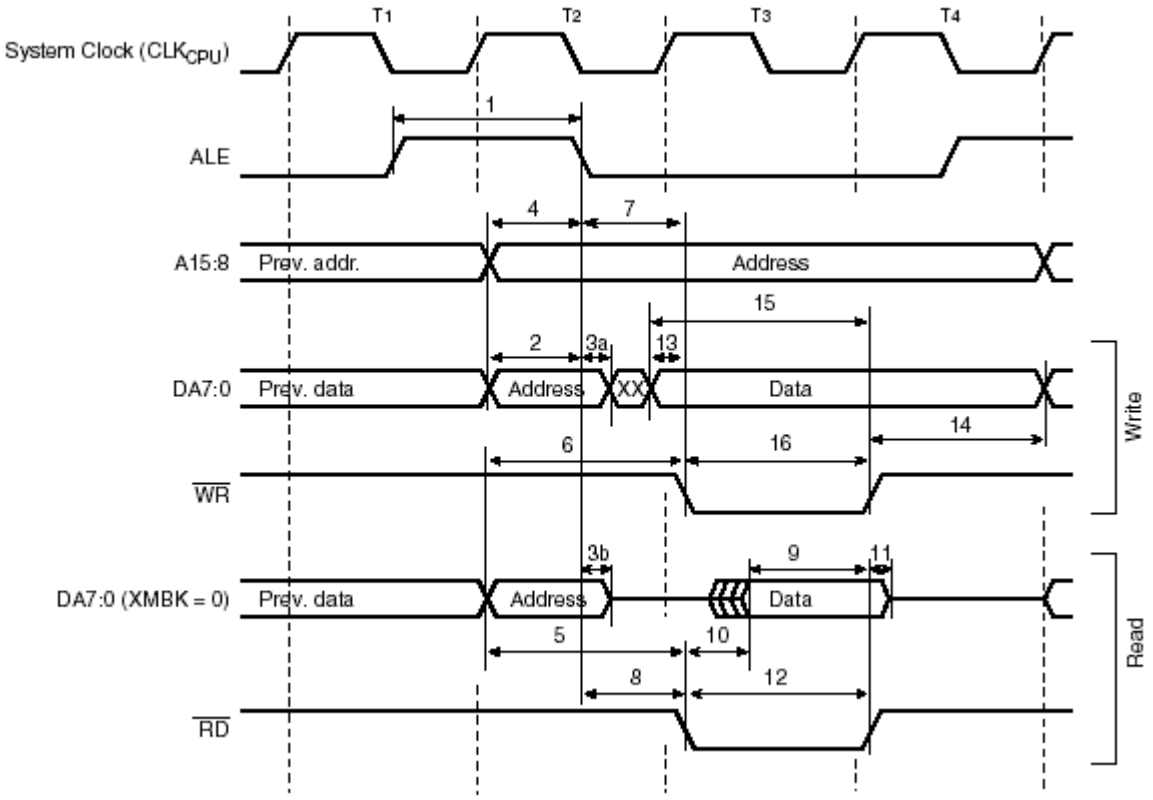
	Symbol	**Nazwa**	**8MHZ Oscylator**		**Zmienny oscylator**		*Jednostka*
			Min	**Max**	**Min**	**Max**	
0	$1/t_{CLCL}$	Częstotliwość oscylatora			0.0	8	MHz
10	t_{RLDV}	Read Low do Data Valid		690		$3.0t_{CLCL}$-60	ns
12	t_{RLRH}	Szerokość	735		$3.0t_{CLCL}$-15		ns

		impulsu RD					
15	t_{DVWH}	Data Valid do WR High	750		$3.0t_{CLCL}$		ns
16	t_{WLWH}	Szerokość impulsu WR	735		$3.0t_{CLCL}$-15		ns

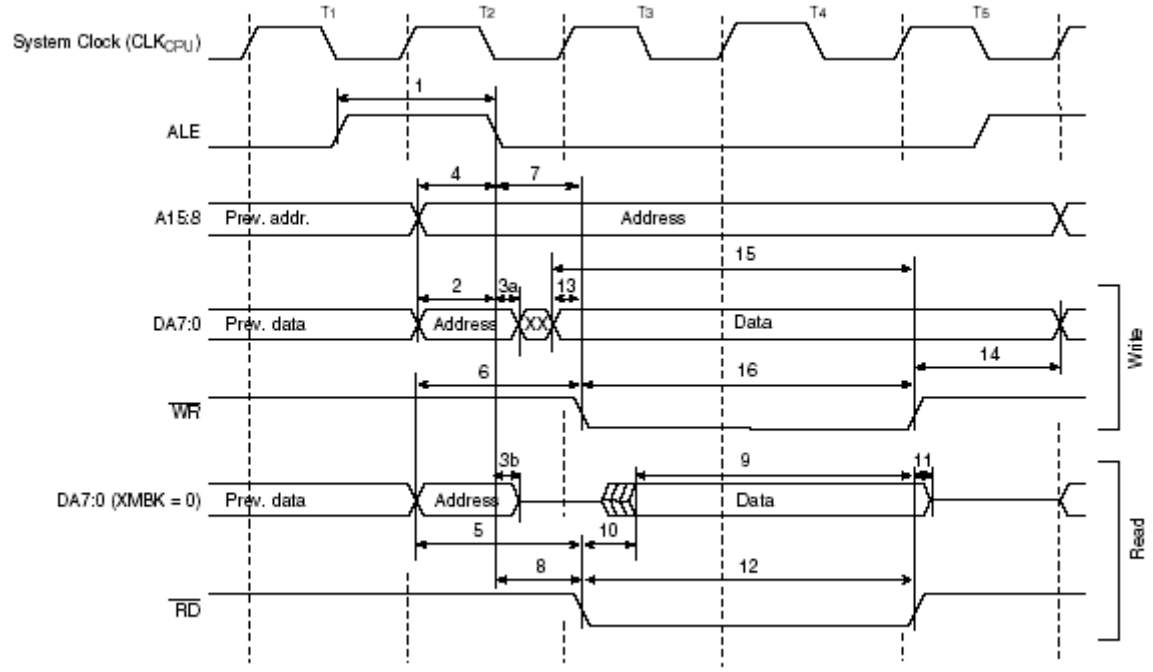
Tabela 144. Charakterystyka pamięci zewnętrznej, 2.7 - 5.5 Volts, SRWn1 = 1, SRWn0 = 1

	Symbol	**Nazwa**	**8MHZ Oscylator**		**Zmienny oscylator**		*Jednostka*
			Min	**Max**	**Min**	**Max**	
0	$1/t_{CLCL}$	Częstotliwość oscylatora			0.0	8	MHz
10	t_{RLDV}	Read Low do Data Valid		690		$3.0t_{CLCL}$-60	ns
12	t_{RLRH}	Szerokość impulsu RD	735		$3.0t_{CLCL}$-15		ns
14	t_{WHDX}	Data Hold po WR High	485		$2.0t_{CLCL}$-15		ns
15	t_{DVWH}	Valid do WR High	750		$3.0t_{CLCL}$		ns
16	t_{WLWH}	Szerokość impulsu WR	735		$3.0t_{CLCL}$-15		ns

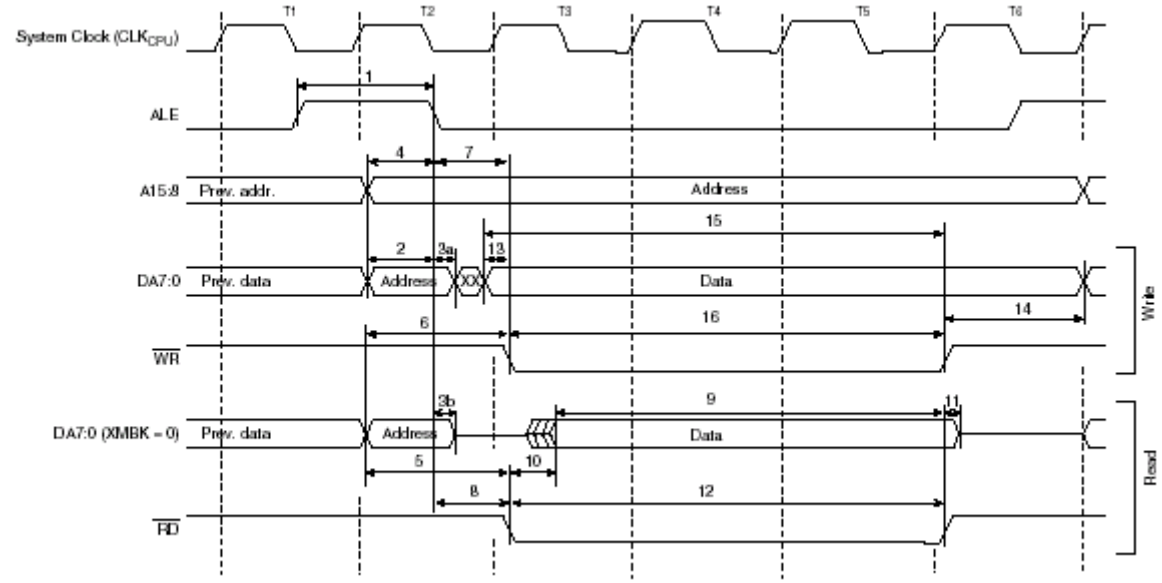
Rys. 156. Synchronizacja pamięci zewnętrznej(SRWn1 = 0, SRWn0 = 0



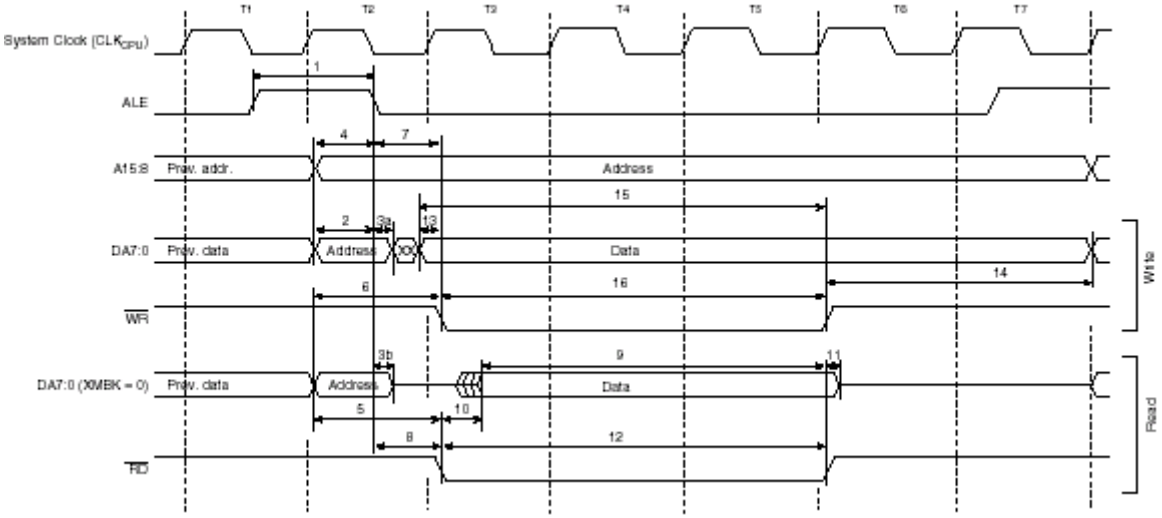
Rys. 157. Synchronizacja pamięci zewnętrznej (SRWn1 = 0, SRWn0 = 1)



Rys. 158. Synchronizacja pamięci zewnętrznej (SRWn1 = 1, SRWn0 = 0)



Rys. 159. Synchronizacja pamięci zewnętrznej (SRWn1 = 1, SRWn0 = 1)(1)

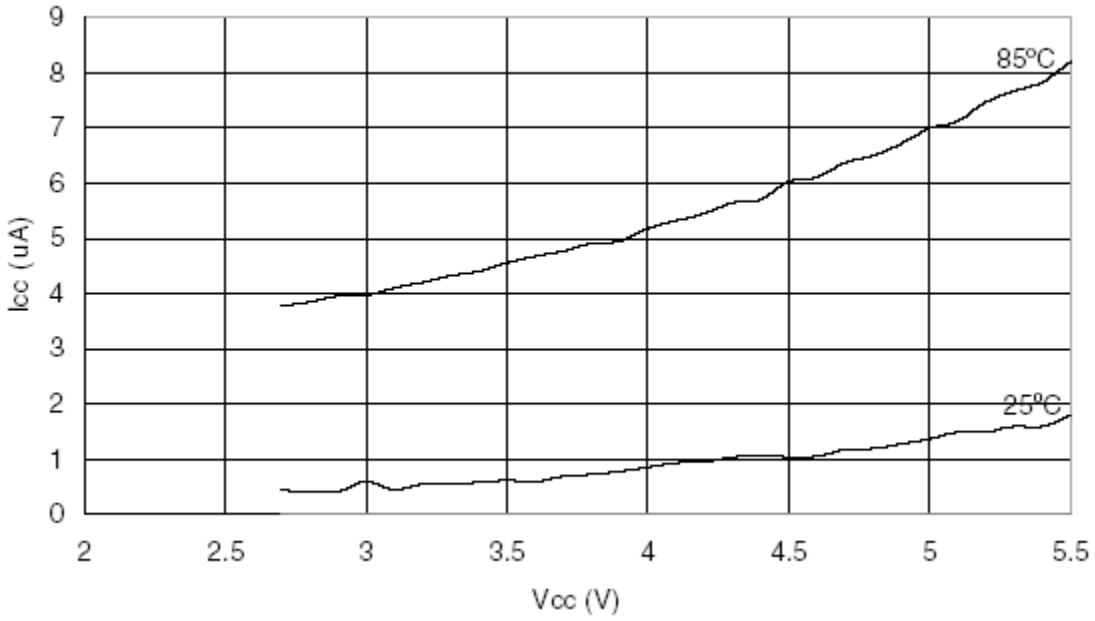


Uwaga: 1. Impuls ALE w ostatnim okresie(T4-T7) istnieje tylko jeśli następna instrukcja ma dostęp do pamięci RAM (wewnętrznej lub zewnętrznej).

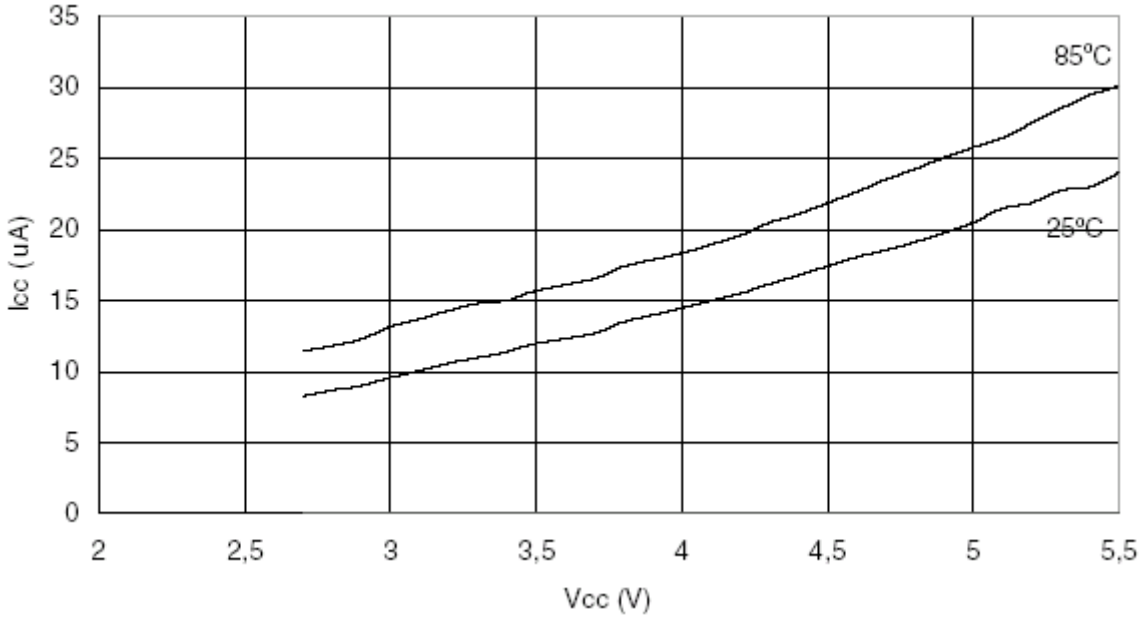
WŁAŚCIWOŚCI ATMEGA128 T – DANE WSTĘPNE

Poniższe wykresy ilustrują typowe zachowanie. Wszystkie pomiary poboru prądu zostały wykonane dla wszystkich styków we/wy skonfigurowanych jako wejścia z włączonym wewnętrznym rezystorem podciągającym. Jako zegar został użyty generator przebiegów sinusoidalnych z wyjściem rail-to-rail.
Pobór prądu w stanie wyłączenia nie zależy wyboru zegara. Pobór prądu jest funkcją kilku czynników: napięcia roboczego, częstotliwości roboczej, obciążenia styków we/wy, tempa przełączania styków we/wy oraz temperatury otoczenia. Przeważającymi czynnikami są robocze napięcie oraz częstotliwość.
Prąd pobrany z naładowanych pojemnością styków może być obliczony (dla jednego styku) jako $C_L*V_{CC}*f$ gdzie C_L = naładowana pojemność, V_{CC} = napięcie robocze oraz f = średnia częstotliwość przełączania dla styków we/wy.

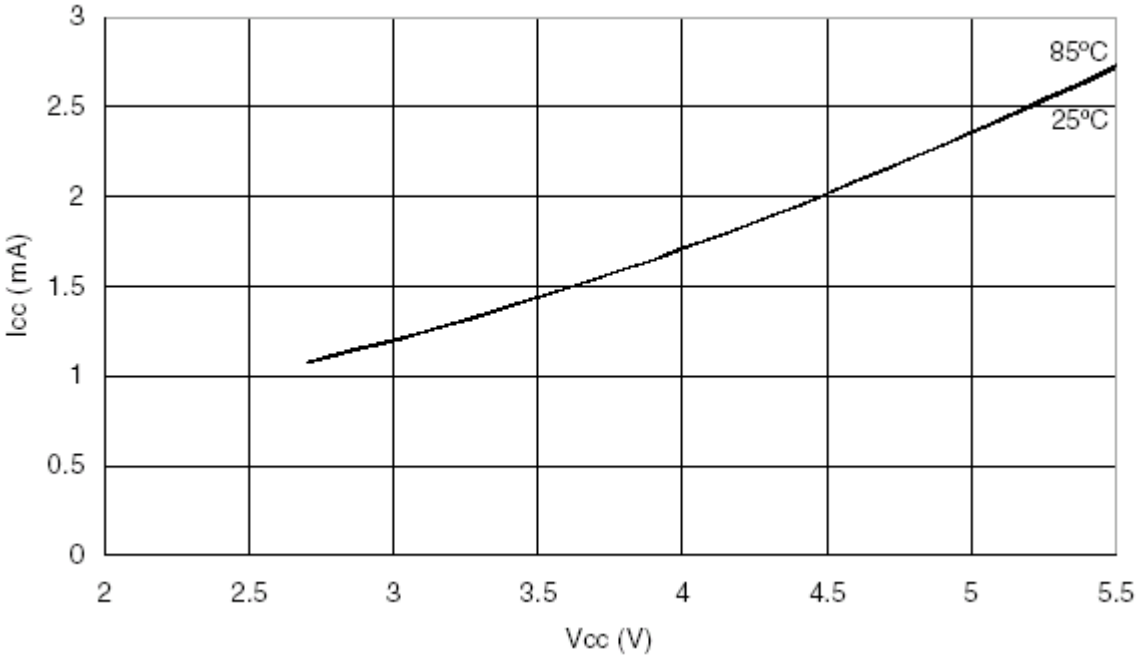
Rys. 160. Prąd zasilania w stanie wyłączenia, napięcie zasilaniaV_{CC} (sygnalizator Watchdog wyłączony)



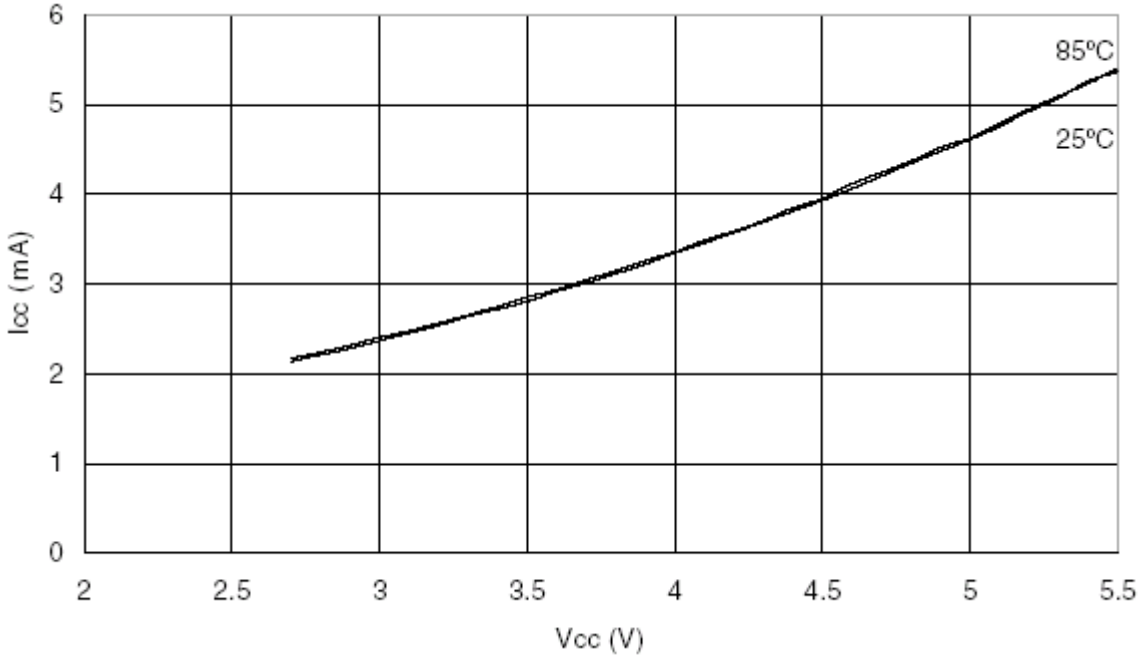
Rys. 161. Prąd zasilania w stanie wyłączenia, napięcie zasilania V_{CC} (sygnalizator Watchdog włączony)



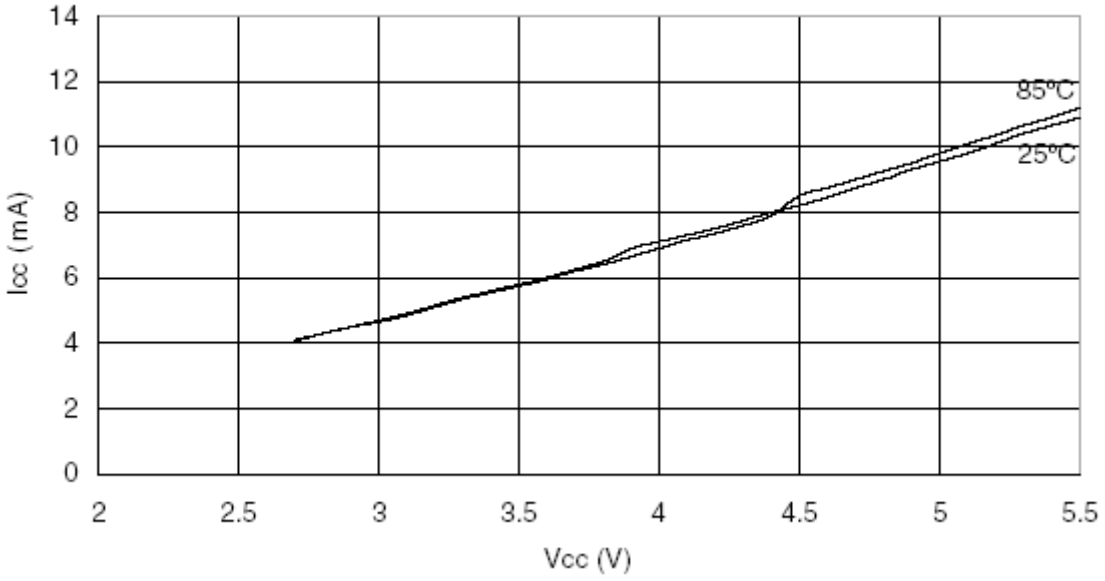
Rys. 162. Aktywny prąd zasilania, napięcie zasilania V_{CC}, wewnętrzny RC 1 MHz



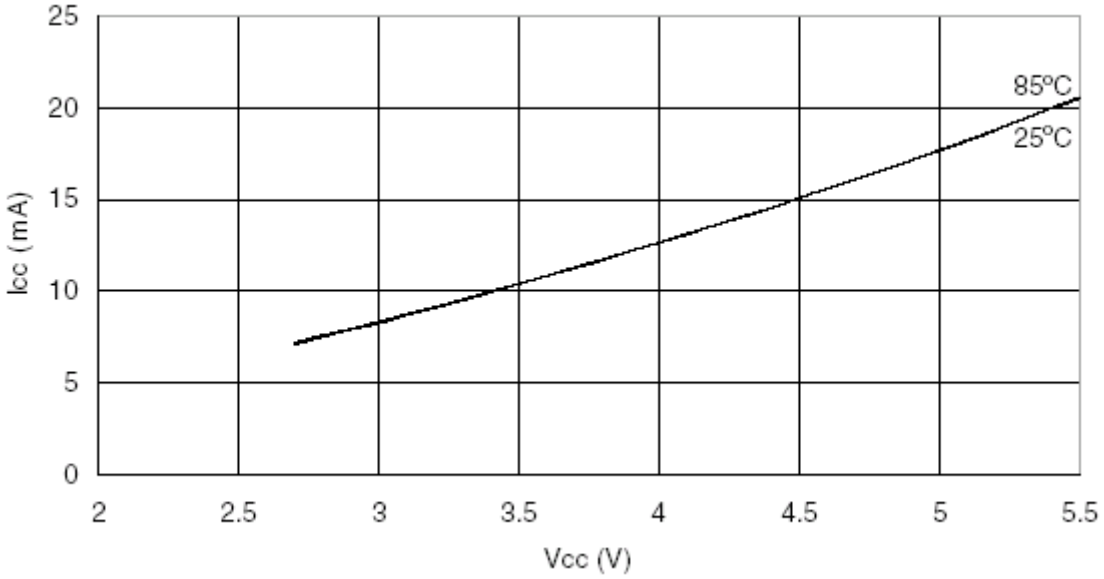
Rys. 163 Aktywny prąd zasilania, napięcie zasilania VCC, wewnętrzny RC 2 MHz



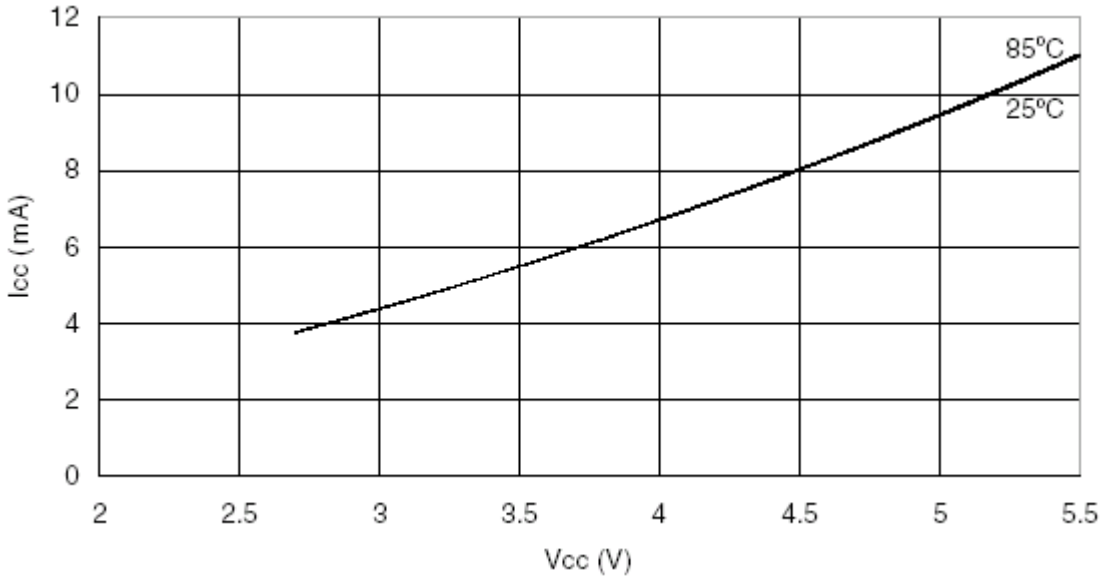
Rys. 164. Aktywny prąd zasilania, napięcie zasilania VCC, wewnętrzny RC 4 MHz



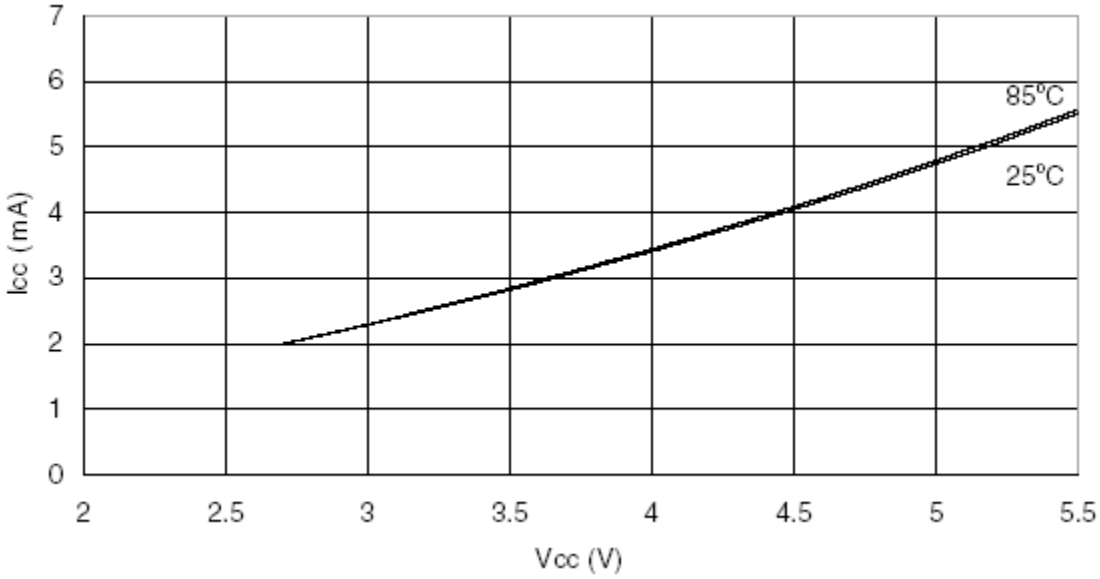
Rys. 165. Aktywny prąd zasilania, napięcie zasilania VCC, wewnętrzny RC 8 MHz



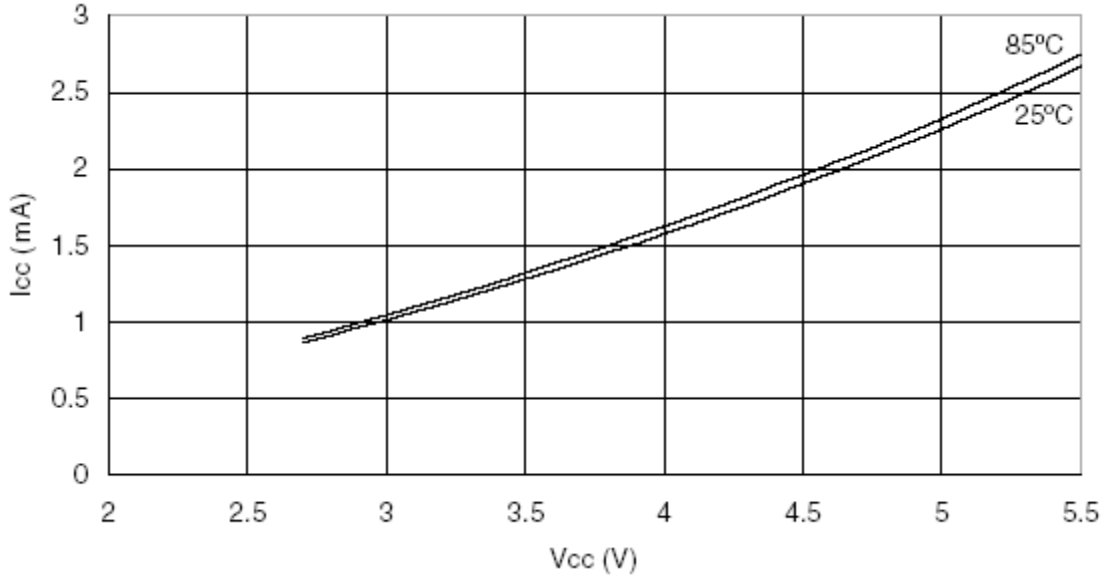
Rys. 166. Bierny prąd zasilania , napięcia zasilania V_{CC}, wewnętrzny RC 8 MHz



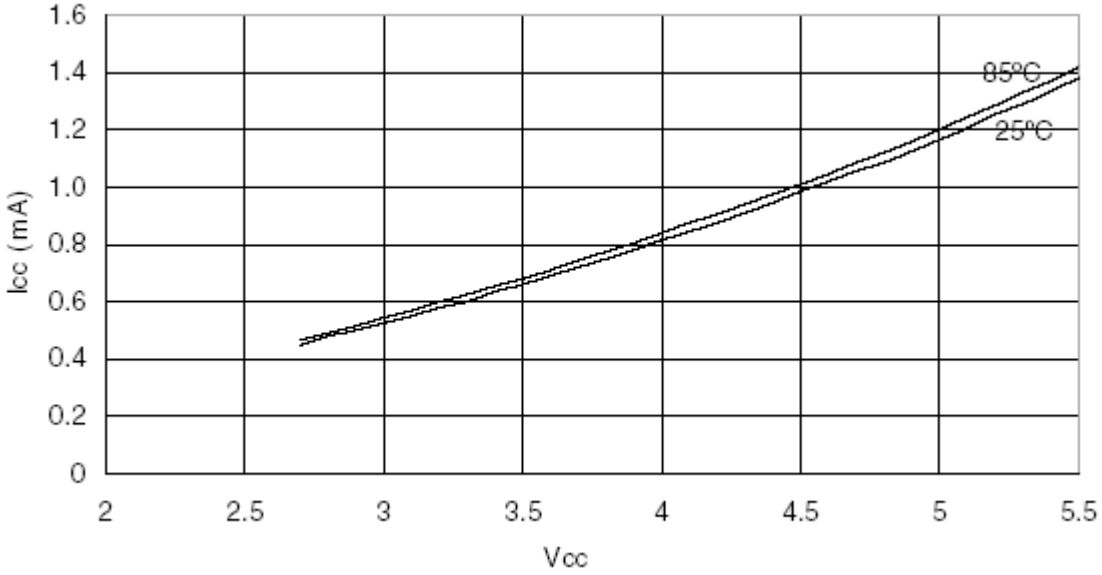
Rys. 167. Bierny prąd zasilania , napięcia zasilania V_{CC}, wewnętrzny RC 4 MHz



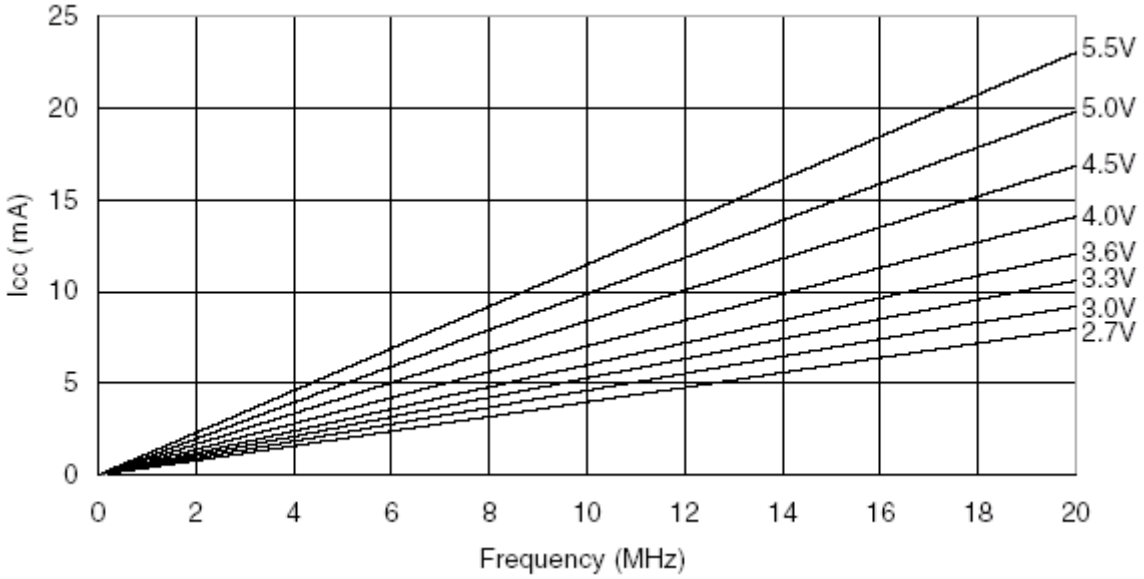
Rys. 168. Bierny prąd zasilania , napięcia zasilania V_{CC}, wewnętrzny RC 2 MHz



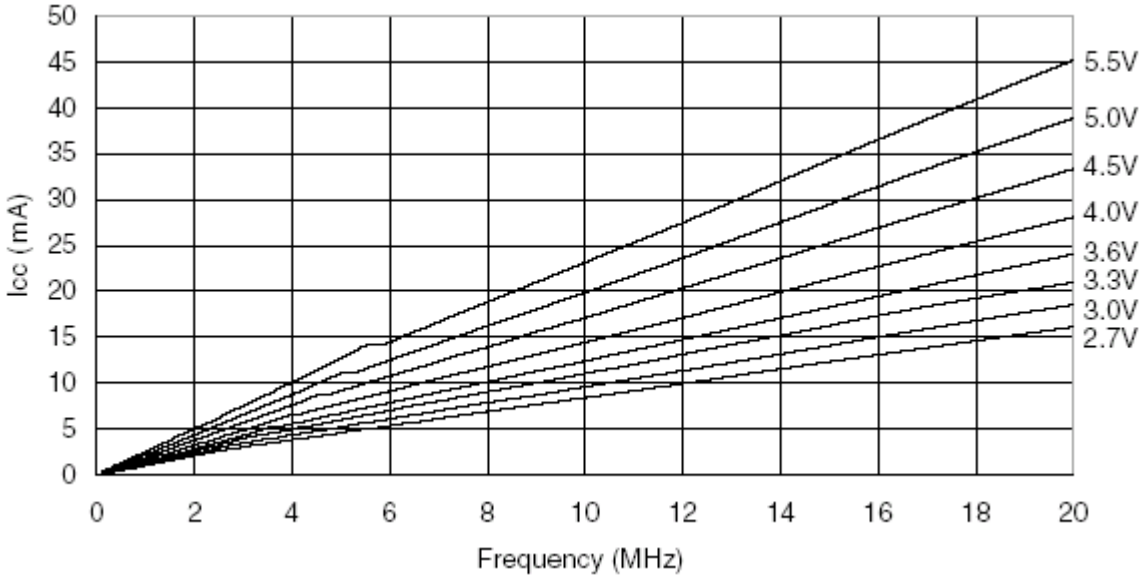
Rys. 169. Bierny prąd zasilania , napięcia zasilania V_{CC}, wewnętrzny RC 1 MHz



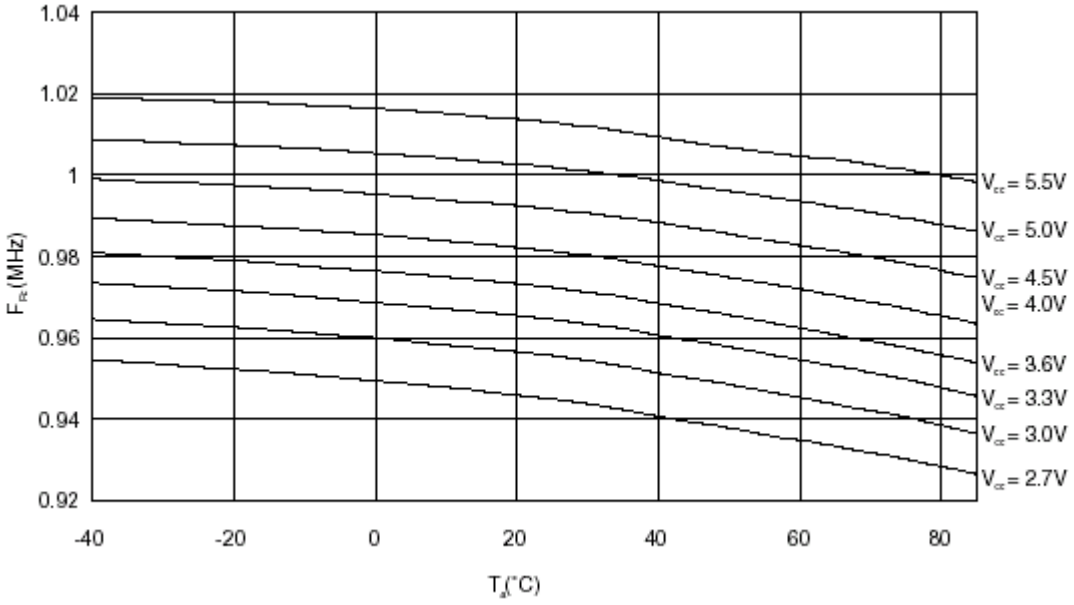
Rys. 170. Bierny prąd zasilania , częstotliwość



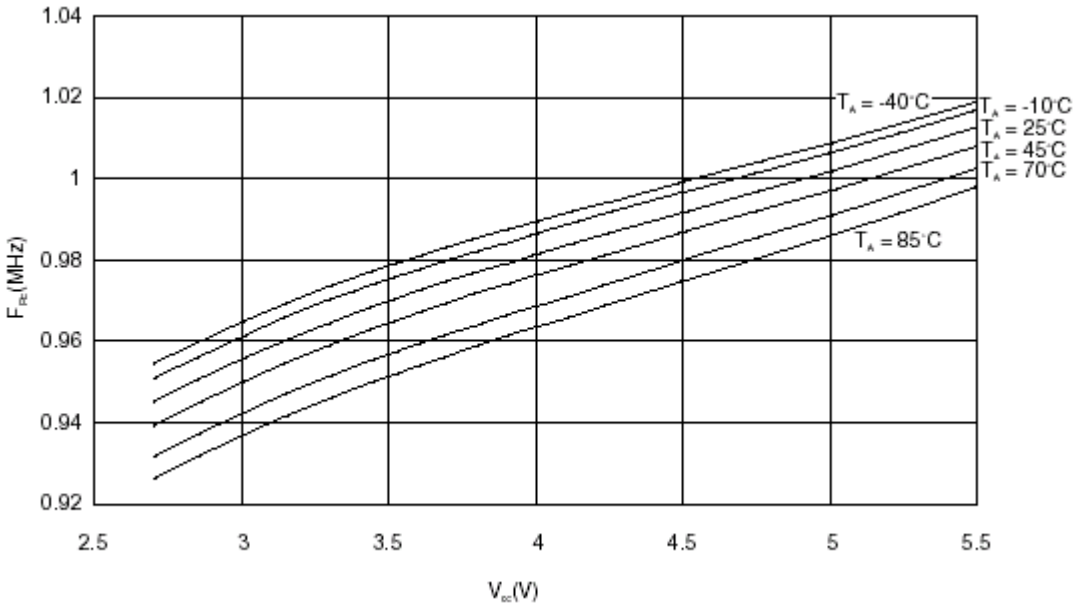
Rys. 171. Aktywny prąd zasilania, częstotliwość



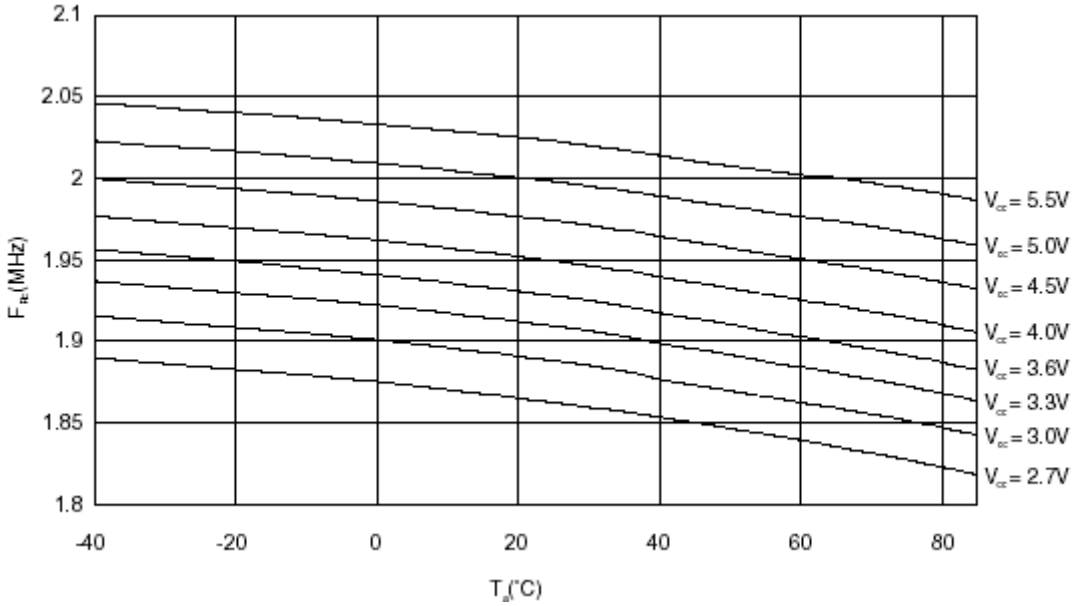
Rys. 172. Częstotliwość oscylatora RC, temperatura(urządzenia są wyskalowane do 1 MHz przy Vcc = 5V, T=25c)



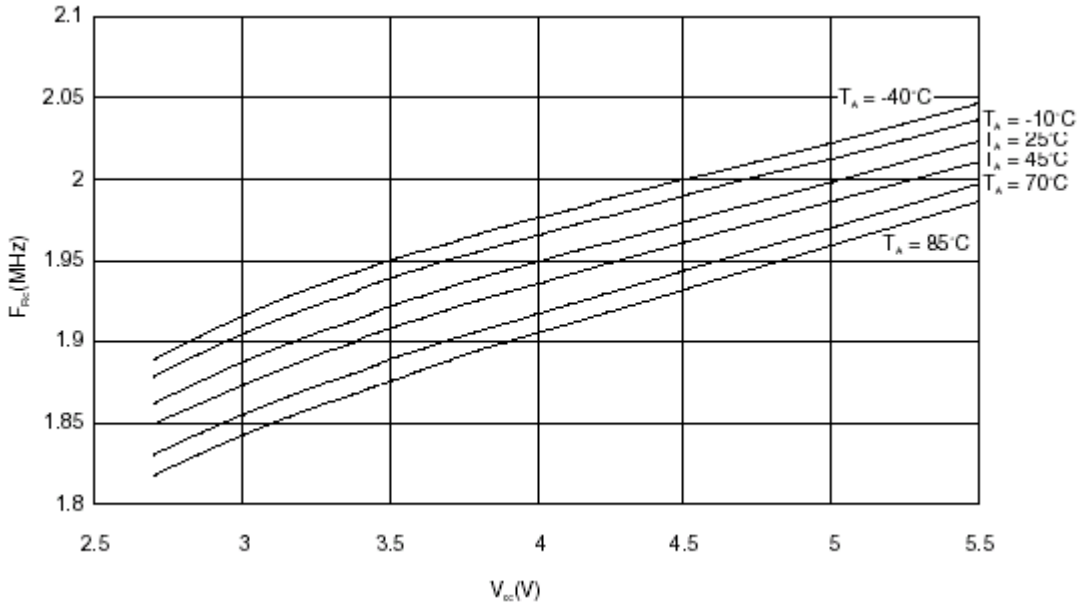
Rys. 173. Częstotliwość oscylatora RC , napięcie robocze(urządzenia są wyskalowane do 1 MHz przy Vcc = 5V, T=25c)



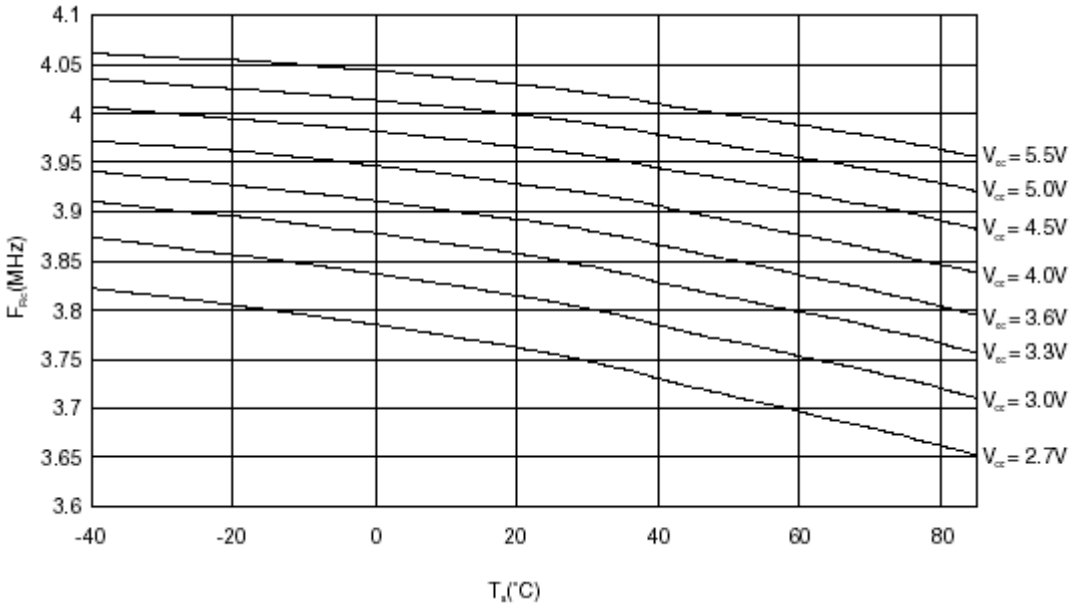
Rys. 174. Częstotliwość oscylatora RC, temperatura(urządzenia są wyskalowane do 2 MHz przy Vcc = 5V, T=25c)



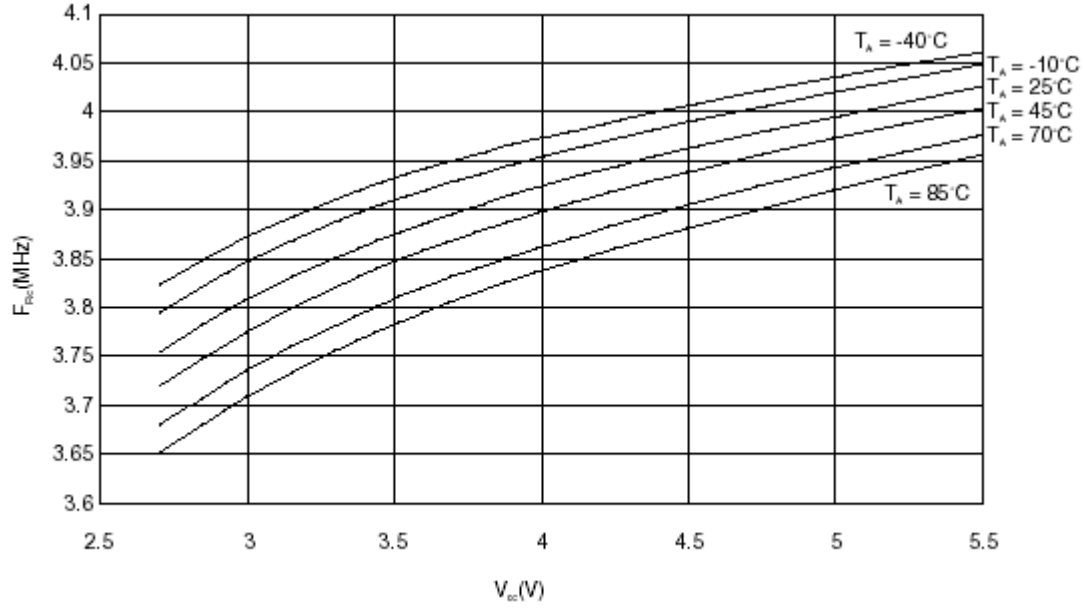
Rys. 175. Częstotliwość oscylatora RC , napięcie robocze(urządzenia są wyskalowane do 2 MHz przy Vcc = 5V, T=25c)



Rys. 176. Częstotliwość oscylatora RC, temperatura(urządzenia są wyskalowane do 4 MHz przy Vcc = 5V, T=25c)

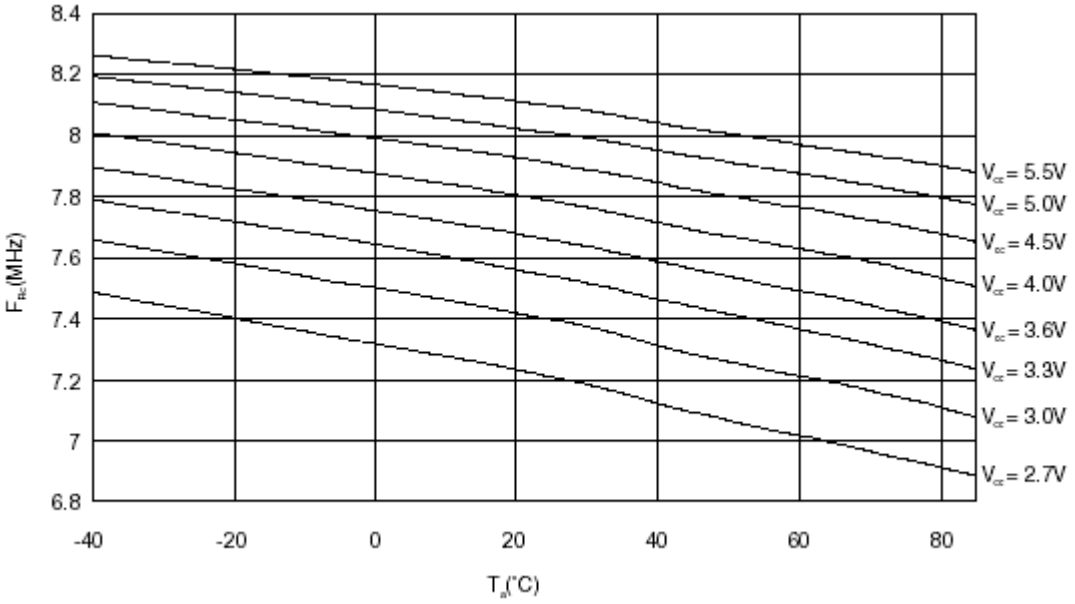


Rys. 177. Częstotliwość oscylatora RC , napięcie robocze(urządzenia są wyskalowane do 4 MHz przy Vcc = 5V, T=25c)

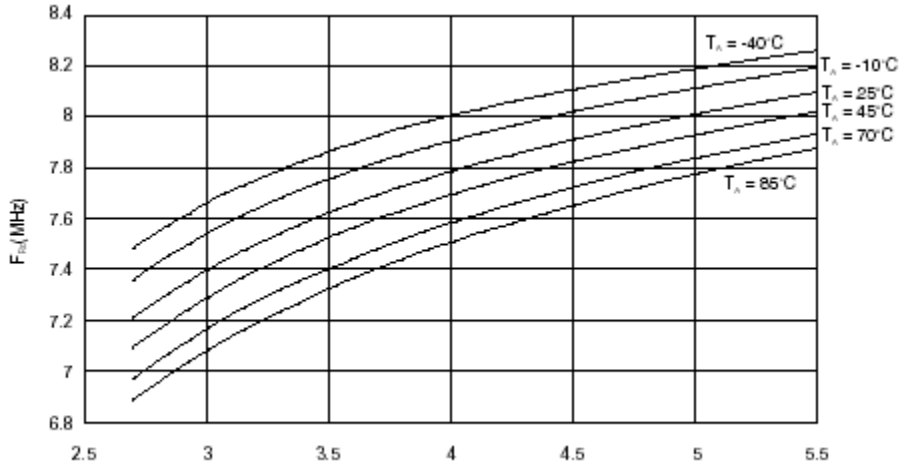


Rys. 178. Częstotliwość oscylatora RC, temperatura(urządzenia są wyskalowane do

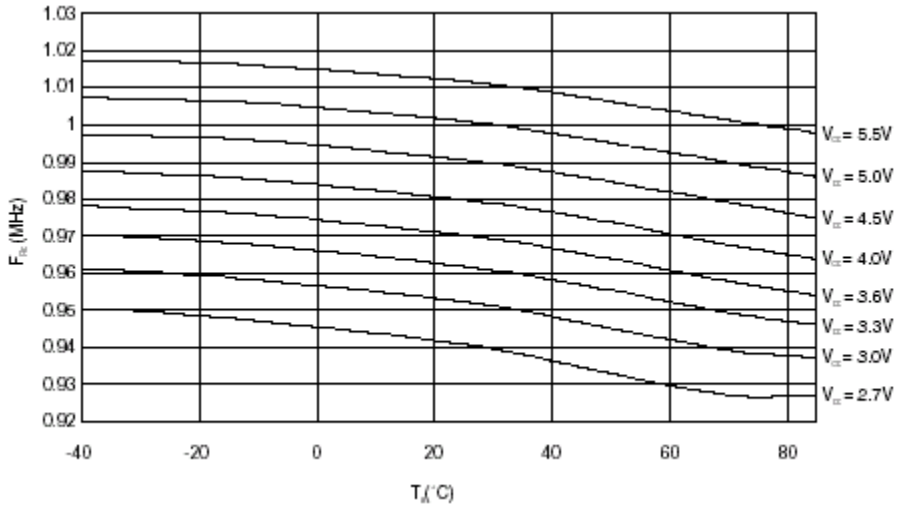
8 MHz przy Vcc = 5V, T=25c)



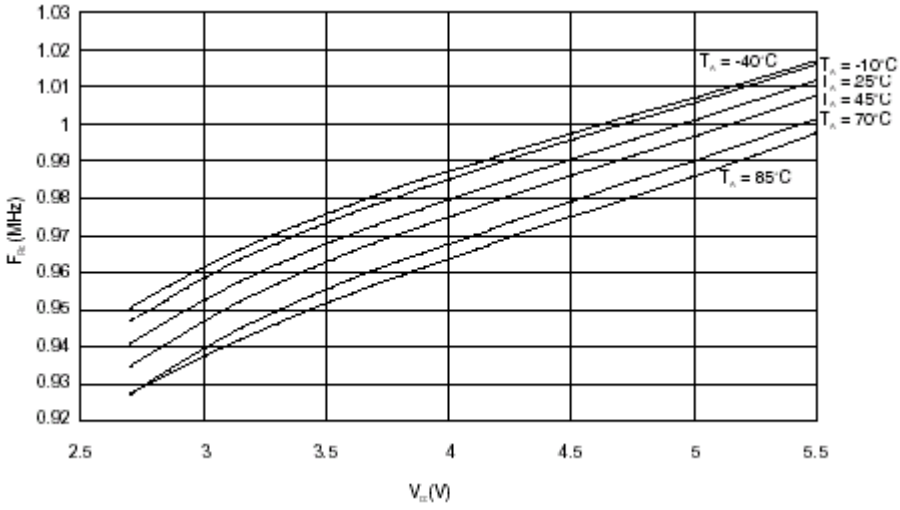
Rys. 179. Częstotliwość oscylatora RC , napięcie robocze(urządzenia są wyskalowane do 8 MHz przy Vcc = 5V, T=25c)



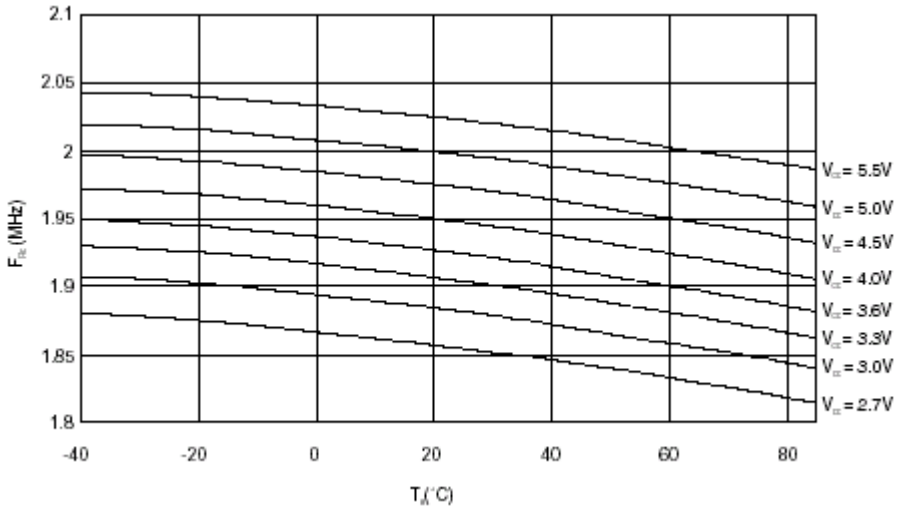
Rys. 180. Częstotliwość oscylatora RC, temperatura(urządzenia są wyskalowane do 1 MHz przy Vcc = 5V, T=25c)



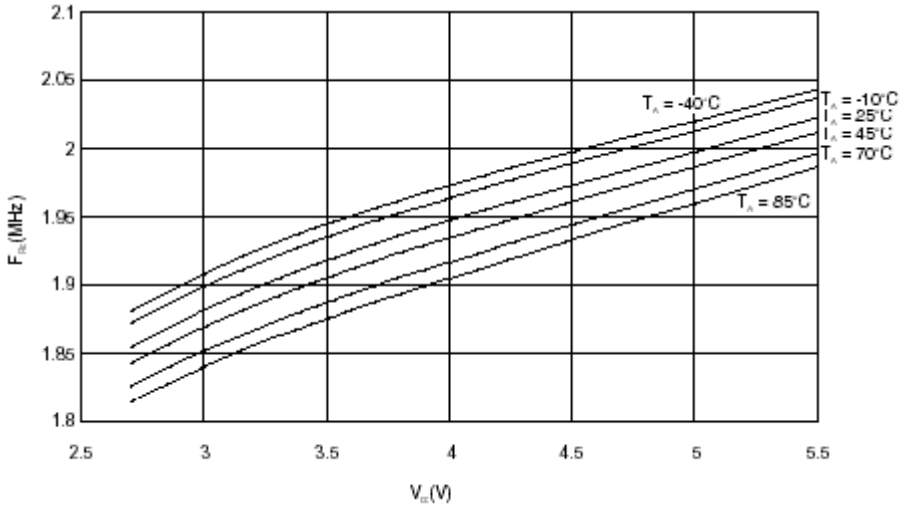
Rys. 181. Częstotliwość oscylatora RC , napięcie robocze(urządzenia są wyskalowane do 1 MHz przy Vcc = 5V, T=25c)



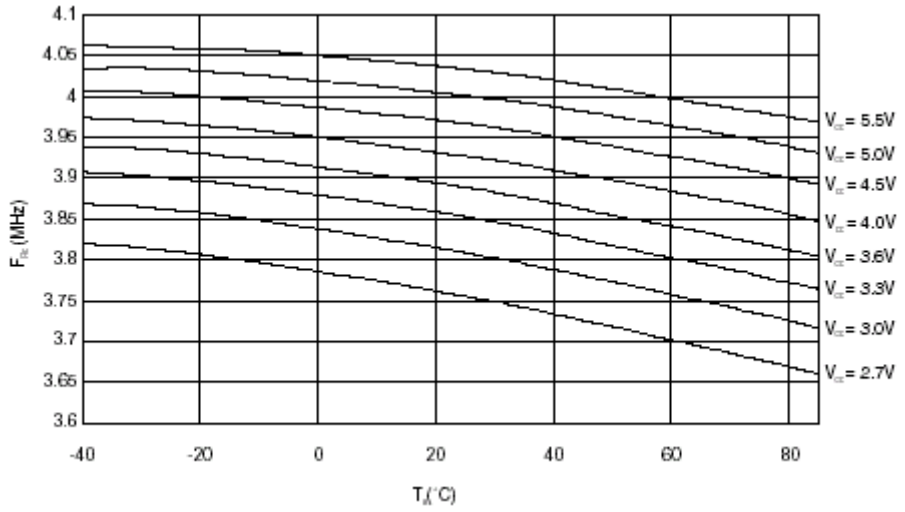
Rys. 182. Częstotliwość oscylatora RC, temperatura(urządzenia są wyskalowane do 2 MHz przy Vcc = 5V, T=25c)



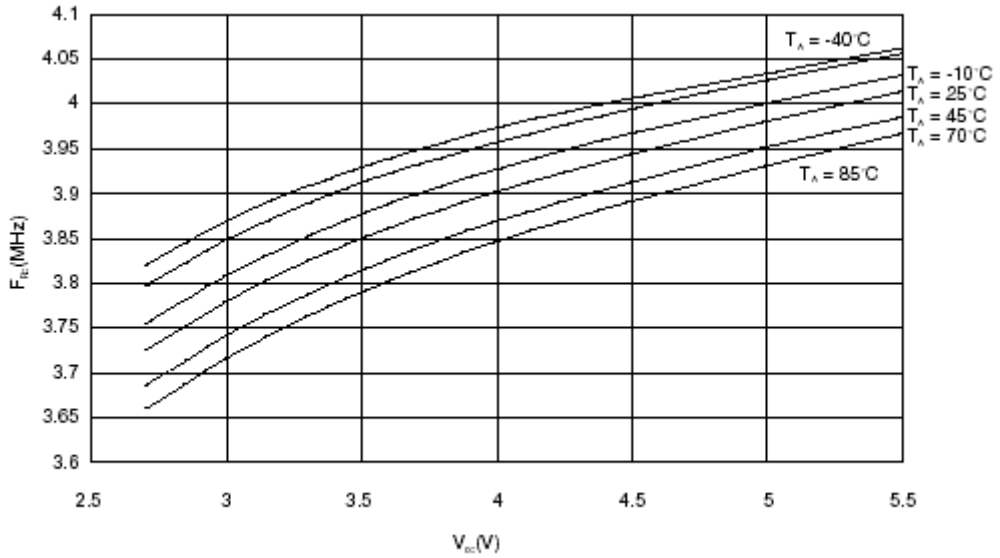
Rys. 183. Częstotliwość oscylatora RC , napięcie robocze(urządzenia są wyskalowane do 2 MHz przy Vcc = 5V, T=25c)



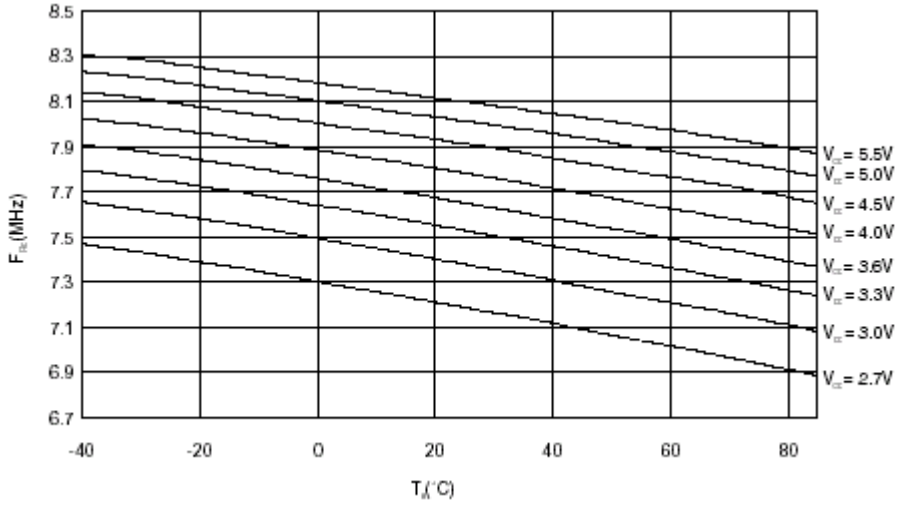
Rys. 184. Częstotliwość oscylatora RC, temperatura(urządzenia są wyskalowane do 4 MHz przy Vcc = 5V, T=25c)



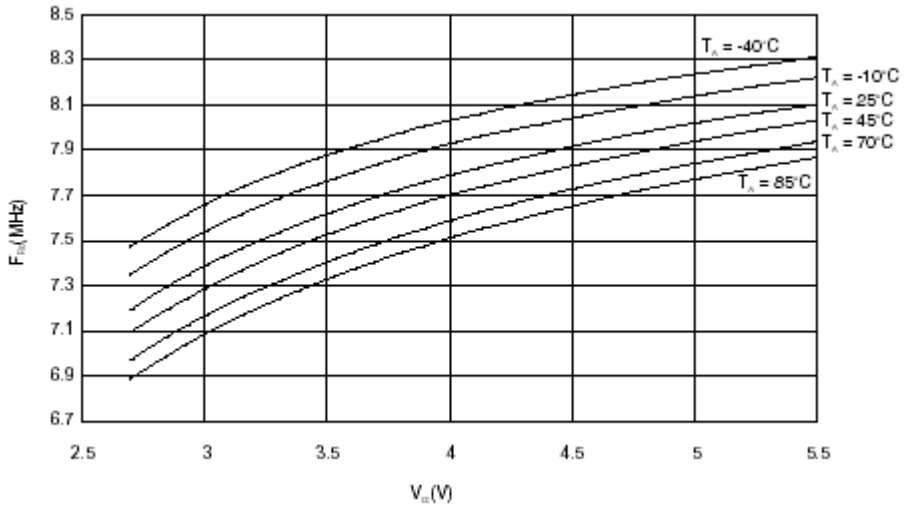
Rys. 185. Częstotliwość oscylatora RC , napięcie robocze(urządzenia są wyskalowane do 4 MHz przy Vcc = 5V, T=25c)



Rys. 186. Częstotliwość oscylatora RC, temperatura(urządzenia są wyskalowane do 8 MHz przy Vcc = 5V, T=25c)



Rys. 187. Częstotliwość oscylatora RC , napięcie robocze(urządzenia są wyskalowane do 8 MHz przy Vcc = 5V, T=25c)



SPIS REJESTRÓW

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
($FF)	Reserved	–	–	–	–	–	–	–	–	
..	Reserved	–	–	–	–	–	–	–	–	
($9E)	Reserved	–	–	–	–	–	–	–	–	
($9D)	UCSR1C	–	UMSEL1	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	186
($9C)	UDR1	USART1 I/O Data Register								183
($9B)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	184
($9A)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	185
($99)	UBRR1L	USART1 Baud Rate Register Low								188
($98)	UBRR1H	–	–	–	–	USART1 Baud Rate Register High				188
($97)	Reserved	–	–	–	–	–	–	–	–	
($96)	Reserved	–	–	–	–	–	–	–	–	
($95)	UCSR0C	–	UMSEL0	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	186
($94)	Reserved	–	–	–	–	–	–	–	–	
($93)	Reserved	–	–	–	–	–	–	–	–	
($92)	Reserved	–	–	–	–	–	–	–	–	
($91)	Reserved	–	–	–	–	–	–	–	–	
($90)	UBRR0H	–	–	–	–	USART0 Baud Rate Register High				188
($8F)	Reserved	–	–	–	–	–	–	–	–	
($8E)	Reserved	–	–	–	–	–	–	–	–	
($8D)	Reserved	–	–	–	–	–	–	–	–	
($8C)	TCCR3C	FOC3A	FOC3B	FOC3C	–	–	–	–	–	132
($8B)	TCCR3A	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	127
($8A)	TCCR3B	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	130
($89)	TCNT3H	Timer/Counter3 – Counter Register High Byte								132
($88)	TCNT3L	Timer/Counter3 – Counter Register Low Byte								132
($87)	OCR3AH	Timer/Counter3 – Output Compare Register A High Byte								133
($86)	OCR3AL	Timer/Counter3 – Output Compare Register A Low Byte								133
($85)	OCR3BH	Timer/Counter3 – Output Compare Register B High Byte								133
($84)	OCR3BL	Timer/Counter3 – Output Compare Register B Low Byte								133
($83)	OCR3CH	Timer/Counter3 – Output Compare Register C High Byte								133
($83)	OCR3CH	Timer/Counter3 – Output Compare Register C High Byte								133
($82)	OCR3CL	Timer/Counter3 – Output Compare Register C Low Byte								133
($81)	ICR3H	Timer/Counter3 – Input Capture Register High Byte								134
($80)	ICR3L	Timer/Counter3 – Input Capture Register Low Byte								134
($7F)	Reserved	–	–	–	–	–	–	–	–	
($7E)	Reserved	–	–	–	–	–	–	–	–	
($7D)	ETIMSK	–	–	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C	135
($7C)	ETIFR	–	–	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C	136
($7B)	Reserved	–	–	–	–	–	–	–	–	
($7A)	TCCR1C	FOC1A	FOC1B	FOC1C	–	–	–	–	–	131
($79)	OCR1CH	Timer/Counter1 – Output Compare Register C High Byte								133
($78)	OCR1CL	Timer/Counter1 – Output Compare Register C Low Byte								133
($77)	Reserved	–	–	–	–	–	–	–	–	
($76)	Reserved	–	–	–	–	–	–	–	–	
($75)	Reserved	–	–	–	–	–	–	–	–	
($74)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	201
($73)	TWDR	Two-wire Serial Interface Data Register								203
($72)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	203
($971	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	202
($70)	TWBR	Two-wire Serial Interface Bit Rate Register								201
($6F)	OSCCAL	Oscillator Calibration Register								38
($6E)	Reserved	–	–	–	–	–	–	–	–	
($6D)	XMCRA	–	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11		29
($6C)	XMCRB	XMBK	–	–	–	–	XMM2	XMM1	XMM0	31
($6B)	Reserved	–	–	–	–	–	–	–	–	
($6A)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	84
($69)	Reserved	–	–	–	–	–	–	–	–	
($68)	SPMCSR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	273
($67)	Reserved	–	–	–	–	–	–	–	–	
($66)	Reserved	–	–	–	–	–	–	–	–	
($65)	PORTG	–	–	–	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	83
($64)	DDRG	–	–	–	DDG4	DDG3	DDG2	DDG1	DDG0	83
($63)	PING	–	–	–	PING4	PING3	PING2	PING1	PING0	83
($62)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	82

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
($61)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	83
($60)	Reserved	–	–	–	–	–	–	–	–	
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	9
$3E ($5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	12
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
$3C ($5C)	XDIV	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0	40
$3B ($5B)	RAMPZ	–	–	–	–	–	–	–	RAMPZ0	12
$3A ($5A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	85
$39 ($59)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	86
$38 ($58)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF	INTF1	INTF0	86
$37 ($57)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	103, 134, 154
$36 ($56)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	103, 136, 155
$35 ($55)	MCUCR	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	29, 41, 58
$34 ($54)	MCUCSR	JTD	–	–	JTRF	WDRF	BORF	EXTRF	PORF	49, 250
$33 ($53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	98
$32 ($52)	TCNT0	Timer/Counter0 (8 Bit)								100
$31 ($51)	OCR0	Timer/Counter0 Output Compare Register								100
$30 ($50)	ASSR	–	–	–	–	AS0	TCN0UB	OCR0UB	TCR0UB	101
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	127
$2E ($4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	130
$2D ($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								132
$2C ($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								132
$2B ($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								133
$2A ($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								133
$29 ($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								133
$28 ($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								133
$27 ($47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								134
$26 ($46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								134
$25 ($45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	152
$24 ($44)	TCNT2	Timer/Counter2 (8 Bit)								154
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register								154
$22 ($42)	OCDR	IDRD/ OCDR7	OCDR6	OCDR5	OCDR4	OCDR3	OCDR2	OCDR1	OCDR0	247
$21 ($41)	WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	51
$20 ($40)	SFIOR	TSM	–	–	ADHSM	ACME	PUD	PSR0	PSR321	66, 104, 140, 241
$20 ($40)	SFIOR	TSM	–	–	ADHSM	ACME	PUD	PSR0	PSR321	66, 104, 140, 241
$1F ($3F)	EEARH	–	–	–	–	EEPROM Address Register High				19
$1E ($3E)	EEARL	EEPROM Address Register Low Byte								19
$1D ($3D)	EEDR	EEPROM Data Register								20
$1C ($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	20
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	81
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	81
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	81
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	81
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	81
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	81
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	81
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	81
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	82
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	82
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	82
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	82
$0F ($2F)	SPDR	SPI Data Register								164
$0E ($2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	164
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	162
$0C ($2C)	UDR0	USART0 I/O Data Register								183
$0B ($2B)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	184
$0A ($2A)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	185
$09 ($29)	UBRR0L	USART0 Baud Rate Register Low								188
$08 ($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	222
$07 ($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	238
$06 ($26)	ADCSRA	ADEN	ADSC	ADRF	ADIF	ADIE	ADPS2	ADPS1	ADPS0	239
$05 ($25)	ADCH	ADC Data Register High Byte								241
$04 ($24)	ADCL	ADC Data Register Low byte								241
$03 ($23)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	82
$02 ($22)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	82

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$01 ($21)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	82
$00 ($20)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	83

Uwagi: 1. W celu zapewnienia kompatybilności z przyszłymi urządzeniami, bity zarezerwowane powinny być ustawione na zero jeżeli jest do nich dostęp. Zarezerwowane adresy pamięci I/O nigdy nie powinny być zarezerwowane.

2. Niektóre flagi stanu mogą być wyzerowane poprzez wpisanie logicznej jedynki do nich.. Warto zauważyć, że rozkazy CBI i SBI będą operować na wszystkich bitach rejestrów I/O zapisując jedynkę z powrotem do którejkolwiek z odczytanych flag, w związku z tym czyszcząc flagę. Rozkazy SBI i CBI pracują tylko z rejestrami $00-$1F.

LISTA ROZKAZÓW MIKROKONTROLERA ATMEGA128.

Rozkazy przesłań.

Mnemonik	Opis	Działanie	Znaczniki	Liczba cykli
MOV R*d*, R*s*	przesłanie bajtu między rejestrami	R*d* ← R*s*	–	1
MOVW R*d*, R*s*	przesłanie słowa między parami rejestrów	R*d*+1:R*d* ← R*s*+1:R*s*	–	1
LDI R*d*, *n*	przesłanie wartości natychmiastowej do rejestru	R*d* ← *n*	–	1
LD R*d*, *ri*	przesłanie do rejestru z adresowaniem pośrednim	R*d* ← (*ri*)	–	2
LD R*d*, *ri*+	przesłanie pośrednie z postinkrementacją	R*d* ← (X), X ← X + 1	–	2
LD R*d*, –*ri*	przesłanie pośrednie z predekrementacją	X ← X – 1, R*d* ← (*ri*)	–	2
LDD R*d*, *ri*+*q*	przesłanie pośrednie z przemieszczeniem	R*d* ← (*ri* + *q*), przy czym *ri* nie może być X	–	2
LDS R*d*, *k*	przesłanie pośrednie z pamięci SRAM	R*d* ← (*k*)	–	2
ST *ri*, R*s*	przesłanie pośrednie z rejestru	(*ri*) ← R*s*	–	2
ST *ri*+, R*s*	przesłanie pośrednie z rejestru z postinkrementacją	(*ri*) ← R*s*, *ri* ← *ri* + 1	–	2
ST –*ri*, R*s*	przesłanie pośrednie z rejestru z predekrementacją	*ri* ← *ri* – 1, (*ri*) ← R*s*	–	2
STD *ri*+*q*, R*s*	przesłanie pośrednie z rejestru z przemieszczeniem	(*ri* + *q*) ← R*s*, przy czym *ri* nie może być X	–	2
STS *k*, R*s*	przesłanie pośrednie z rejestru do pamięci SRAM	(*k*) ← R*s*	–	2
LPM	przesłanie z pamięci programu	R0 ← (Z)	–	3
LPM R*d*, Z	przesłanie z pamięci programu	R*d* ← (Z)	–	3
LPM R*d*, Z+	przesłanie z pamięci programu z postinkrementacją	R*d* ← (Z), Z ← Z + 1	–	3
ELPM	rozszerzone przesłanie z pamięci programu	R0 ← (RAMPZ0:Z)	–	3
ELPM R*d*, Z	rozszerzone przesłanie z pamięci programu	R*d* ← (RAMPZ0:Z)	–	3
ELPM R*d*, Z+	rozszerzone przesłanie z pamięci programu z postinkrementacją	Rd ← (RAMPZ0:Z), RAMPZ0:Z ← RAMPZ0:Z + 1	–	3
SPM	przesłanie do pamięci programu	(Z) ← R1:R0	–	–

IN R*d*, *p*	przesłanie z portu do rejestru	R*d* ← *p*	–	1
OUT *p*, R*s*	przesłanie z rejestru do portu	*p* ← R*s*	–	1
PUSH R*s*	położenie zawartości rejestru na stos	SP ← SP – 1, (SP) ← R*r*	–	2
POP R*d*	zdjęcie wartości ze stosu do rejestru	R*d* ← (SP), SP ←SP + 1	–	2

Rozkazy arytmetyczne i logiczne.

Mnemonik	Opis	Działanie	Znaczniki	Liczba cykli
ADD R*d*, R*s*	dodanie zawartości dwóch rejestrów	R*d* ← R*d* + R*s*	Z, C, N, V, H	1
ADC R*d*, R*s*	dodanie zawartości dwóch rejestrów z uwzględnieniem przeniesienia	R*d* ← R*d* + R*s* + C	Z, C, N, V, H	1
ADIW R*d*, *w*	dodanie wartości natychmiastowej do zawartości pary rejestrów	R*d*+1:R*d* ← R*d*+1:R*d* + *w*	Z, C, N, V, S	2
SUB R*d*, R*s*	odjęcie zawartości dwóch rejestrów	R*d* ← R*d* – R*s*	Z, C, N, V, H	1
SUBI R*d*, *n*	odjęcie wartości natychmiastowej od zawartości rejestru	R*d* ← R*d* – *n*	Z, C, N, V, H	1
SBC R*d*, R*s*	odjęcie zawartości dwóch rejestrów z uwzględnieniem przeniesienia (pożyczki)	R*d* ← R*d* – R*s* – C	Z, C, N, V, H	1
SBCI R*d*, *n*	odjęcie wartości natychmiastowej od zawartości rejestru z uwzględnieniem przeniesienia (pożyczki)	R*d* ← R*d* – *n* – C	Z, C, N, V, H	1
SBIW R*d*, *w*	odjęcie wartości natychmiastowej od zawartości pary rejestrów	R*d*+1:R*d* ← R*d*+1:R*d* – *w*	Z, C, N, V, S	2
AND R*d*, R*s*	iloczyn logiczny zawartości dwóch rejestrów	R*d* ← R*d* $\wedge$ R*s*	Z, N, V	1
ANDI R*d*, *n*	iloczyn logiczny zawartości rejestru i wartości natychmiastowej	R*d* ← R*d* $\wedge$ *n*	Z, N, V	1
OR R*d*, R*s*	suma logiczna zawartości dwóch rejestrów	R*d* ← R*d* $\vee$ R*s*	Z, N, V	1
ORI R*d*, *n*	suma logiczna zawartości rejestru i wartości natychmiastowej	R*d* ← R*d* $\vee$ *n*	Z, N, V	1
EOR R*d*, R*s*	suma modulo 2 zawartości dwóch rejestrów	R*d* ← R*d* $\oplus$ R*s*	Z, N, V	1
COM R*d*	uzupełnienie jedynkowe zawartości rejestru	R*d* ← $FF – R*d*	Z, C, N, V	1
NEG R*d*	uzupełnienie dwójkowe (negacja) zawartości rejestru	R*d* ← –R*d*	Z, C, N, V, H	1
SBR R*d*, *n*	ustawienie bitu(ów) w rejestrze	R*d* ← R*d* $\vee$ *n*	Z, N, V	1
CBR R*d*, *n*	wyzerowanie bitu(ów) w rejestrze	R*d* ← R*d* $\wedge$ ($FF – *n*)	Z, N, V	1
INC R*d*	inkrementacja zawartości rejestru	R*d* ← R*d* + 1	Z, N, V	1
DEC R*d*	dekrementacja zawartości rejestru	R*d* ← R*d* – 1	Z, N, V	1
TST R*d*	testowanie zera lub ujemnej zawartości rejestru	R*d* ← R*d* $\wedge$ R*d*	Z, N, V	1
CLR R*d*	wyzerowanie zawartości rejestru	R*d* ← R*d* $\oplus$ R*d*	Z, N, V	1

Mnemonik	Opis	Działanie	Znaczniki	Liczba cykli
SER R*d*	ustawienie wszystkich bitów w rejestrze na 1	R*d* ← $FF	–	1
MUL R*d*, R*s*	mnożenie zawartości rejestrów bez znaku	R1:R0 ← R*d* × R*s*	Z, C	2
MULS R*d*, R*s*	mnożenie zawartości rejestrów ze znakiem	R1:R0 ← R*d* × R*s*	Z, C	2
MULSU R*d*, R*s*	mnożenie zawartości rejestru ze znakiem przez zawartość rejestru bez znaku	R1:R0 ← R*d* × R*s*	Z, C	2
FMUL R*d*, R*s*	ułamkowe mnożenie zawartości rejestrów bez znaku	R1:R0 ← (R*d* × R*s*) << 1	Z, C	2
FMULS R*d*, R*s*	ułamkowe mnożenie zawartości rejestrów ze znakiem	R1:R0 ← (R*d* × R*s*) << 1	Z, C	2
FMULSU R*d*, R*s*	ułamkowe mnożenie zawartości rejestru ze znakiem przez zawartość rejestru bez znaku	R1:R0 ← (R*d* × R*s*) << 1	Z, C	2

Rozkazy skoków i rozgałęzień.

Mnemonik	Opis	Działanie	Znaczniki	Liczba cykli
RJMP *k*	skok względny	PC ← PC + *k* + 1	–	2
IJMP	skok pośredni przez rejestr Z	PC ← Z	–	2
JMP *k*	skok bezpośredni	PC ← *k*	–	3
RCALL *k*	względny skok do podprogramu	SP ← SP – 2, (SP) ← PC, PC ← PC + *k* + 1	–	3
ICALL	pośredni skok do podprogramu przez rejestr Z	SP ← SP – 2, (SP) ← PC, PC ← Z	–	3
CALL *k*	bezpośredni skok do podprogramu	SP ← SP – 2, (SP) ← PC, PC ← *k*	–	4
RET	powrót z podprogramu	PC ← (SP), SP ← SP + 2	–	4
RETI	powrót z podprogramu obsługi przerwania	PC ← (SP), SP ← SP + 2	I	4
CPSE R*d*, R*s*	porównanie zawartości rejestrów i pominięcie następnego rozkazu, gdy równe	if (R*d* = R*s*) PC ← PC + 2 lub 3	–	1 / 2 / 3
CP R*d*, R*s*	porównanie zawartości rejestrów	R*d* – R*s*	Z, N, V, C, H	1
CPC R*d*, R*s*	porównanie zawartości rejestrów z uwzględnieniem przeniesienia	Rd – Rs – C	Z, N, V, C, H	1
CPI R*d*, *n*	porównanie	R*d* – *n*	Z, N, V, C, H	1

Mnemonik	Opis	Działanie	Znaczniki	Liczba cykli
	zawartości rejestru z wartością natychmiastową			
SBRC R*d*, *b*	pominięcie następnego rozkazu, gdy bit w rejestrze wyczyszczony	if (R*d*[*b*] = 0) PC ← PC + 2 lub 3	–	1 / 2 / 3
SBRS R*d*, *b*	pominięcie następnego rozkazu, gdy bit w rejestrze ustawiony	if (R*d*[*b*] = 1) PC ← PC + 2 lub 3	–	1 / 2 / 3
SBIC *p*, *b*	pominięcie następnego rozkazu, gdy bit w porcie wyczyszczony	if (*p*[*b*] = 0) PC ← PC + 2 lub 3	–	1 / 2 / 3
SBIS *p*, *b*	pominięcie następnego rozkazu, gdy bit w porcie ustawiony	if (*p*[b] = 1) PC ← PC + 2 lub 3	–	1 / 2 / 3
BRBS *s*, *k*	skok, gdy bit w rejestrze stanu ustawiony	if (SREG[*s*] = 1) PC ← PC + *k* + 1	–	1 / 2
BRBC *s*, *k*	skok, gdy bit w rejestrze stanu wyczyszczony	if (SREG[*s*] = 0) PC ← PC + *k* + 1	–	1 / 2
BREQ *k*	skok, gdy równe	if (Z = 1) PC ← PC + *k* + 1	–	1 / 2
BRNE *k*	skok, gdy nierówne	if (Z = 0) PC ← PC + *k* + 1	–	1 / 2
BRCS *k*	skok, gdy jest przeniesienie	if (C = 1) PC ← PC + *k* + 1	–	1 / 2
BRCC *k*	skok, gdy brak przeniesienia	if (C = 0) PC ← PC + *k* + 1	–	1 / 2
BRSH *k*	skok, gdy równe lub wyżej	if (C = 0) PC ← PC + *k* + 1	–	1 / 2
BRLO k	skok, gdy niżej	if (C = 1) PC ← PC + *k* + 1	–	1 / 2
BRMI *k*	skok, gdy minus	if (N = 1) PC ← PC + *k* + 1	–	1 / 2
BRPL *k*	skok, gdy plus	if (N = 0) PC ← PC + *k* + 1	–	1 / 2
BRGE *k*	skok, gdy większe lub równe	if (N $\oplus$ V = 0) PC ← PC + *k* + 1	–	1 / 2
BRLT *k*	skok, gdy mniejsze od zera	if (N $\oplus$ V = 1) PC ← PC + k + 1	–	1 / 2
BRHS *k*	skok, gdy jest przeniesienie połówkowe	if (H = 1) PC ← PC + *k* + 1	–	1 / 2
BRHC *k*	skok, gdy brak przeniesienia połówkowego	if (H = 0) PC ← PC + *k* + 1	–	1 / 2
BRTS *k*	skok, gdy ustawiona flaga T	if (T = 1) PC ← PC + *k* + 1	–	1 / 2
BRTC *k*	skok, gdy wyczyszczona flaga T	if (T = 0) PC ← PC + *k* + 1	–	1 / 2
BRVS *k*	skok, gdy jest nadmiar	if (V = 1) PC ← PC + *k* + 1	–	1 / 2

Mnemonik	Opis	Działanie	Znaczniki	Liczba cykli
BRVC *k*	skok, gdy nie ma nadmiaru	if (V = 0) PC ← PC + *k* + 1	–	1 / 2
BRIE *k*	skok, gdy przerwania włączone	if (I = 1) PC ← PC + *k* + 1	–	1 / 2
BRID *k*	skok, gdy przerwania wyłączone	if (I = 0) PC ← PC + *k* + 1	–	1 / 2

Rozkazy bitowe.

Mnemonik	Opis	Działanie	Znaczniki	Liczba cykli
SBI *p*, *b*	ustawienie bitu w porcie we/wy	*p*[*b*] ← 1	–	2
CBI *p*, *b*	wyczyszczenie bitu w porcie we/wy	*p*[*b*] ← 0	–	2
LSL R*d*	przesunięcie logiczne w lewo	R*d*[n + 1] ← R*d*[n], R*d*[0] ← 0, n = 0..6	Z, C, N, V	1
LSR R*d*	przesunięcie logiczne w prawo	R*d*[n] ← R*d*[n + 1], R*d*[7] ← 0, n = 0..6	Z, C, N, V	1
ROL R*d*	obrót w lewo przez bit przeniesienia	R*d*[0] ← C, R*d*[n + 1] ← R*d*[n], C ← R*d*[7], n = 0..6	Z, C, N, V	1
ROR R*d*	obrót w prawo przez bit przeniesienia	R*d*[7] ← C, R*d*[n] ←R*d*[n + 1], C ← Rd[0], n = 0..6	Z, C, N, V	1
ASR R*d*	przesunięcie arytmetyczne w prawo	R*d*[n] ← R*d*[n + 1], n = 0..6	Z, C, N, V	1
SWAP R*d*	zamiana tetrad w rejestrze	R*d*[3..0] ↔ R*d*[7..4]	–	1
BSET *s*	ustawienie flagi w rejestrze stanu	SREG[*s*] ← 1	SREG[*s*]	1
BCLR *s*	wyczyszczenie flagi w rejestrze stanu	SREG[*s*] ← 0	SREG[*s*]	1
BST R*d*, *b*	przesłanie bitu z rejestru do flagi T	T ← R*d*[*b*]	T	1
BLD R*d*, *b*	przesłanie wartości flagi T do bitu rejestru	R*d*[*b*] ← T	–	1
SEC	ustawienie flagi przeniesienia	C ← 1	C	1
CLC	wyzerowanie flagi przeniesienia	C ← 0	C	1
SEN	ustawienie flagi N	N ← 1	N	1
CLN	wyzerowanie flagi N	N ← 0	N	1
SEZ	ustawienie flagi zera	Z ← 1	Z	1
CLZ	wyczyszczenie flagi zera	Z ← 0	Z	1
SIĘ	włączenie przerwań	I ← 1	I	1
CLI	wyłączenie przerwań	I ← 0	I	1
SES	ustawienie flagi znaku	S ← 1	S	1
CLS	wyczyszczenie flagi znaku	S ← 0	S	1
SEV	ustawienie flagi nadmiaru	V ← 1	V	1
CLV	wyczyszczenie flagi nadmiaru	V ← 0	V	1
SET	ustawienie flagi T	T ← 1	T	1
CLT	wyczyszczenie flagi T	T ← 0	T	1

Mnemonik	Opis	Działanie	Znaczniki	Liczba cykli
SEH	ustawienie flagi przeniesienia połówkowego	H ← 1	H	1
CLH	wyczyszczenie flagi przeniesienia połówkowego	H ← 0	H	1

Rozkazy sterujące MCU.

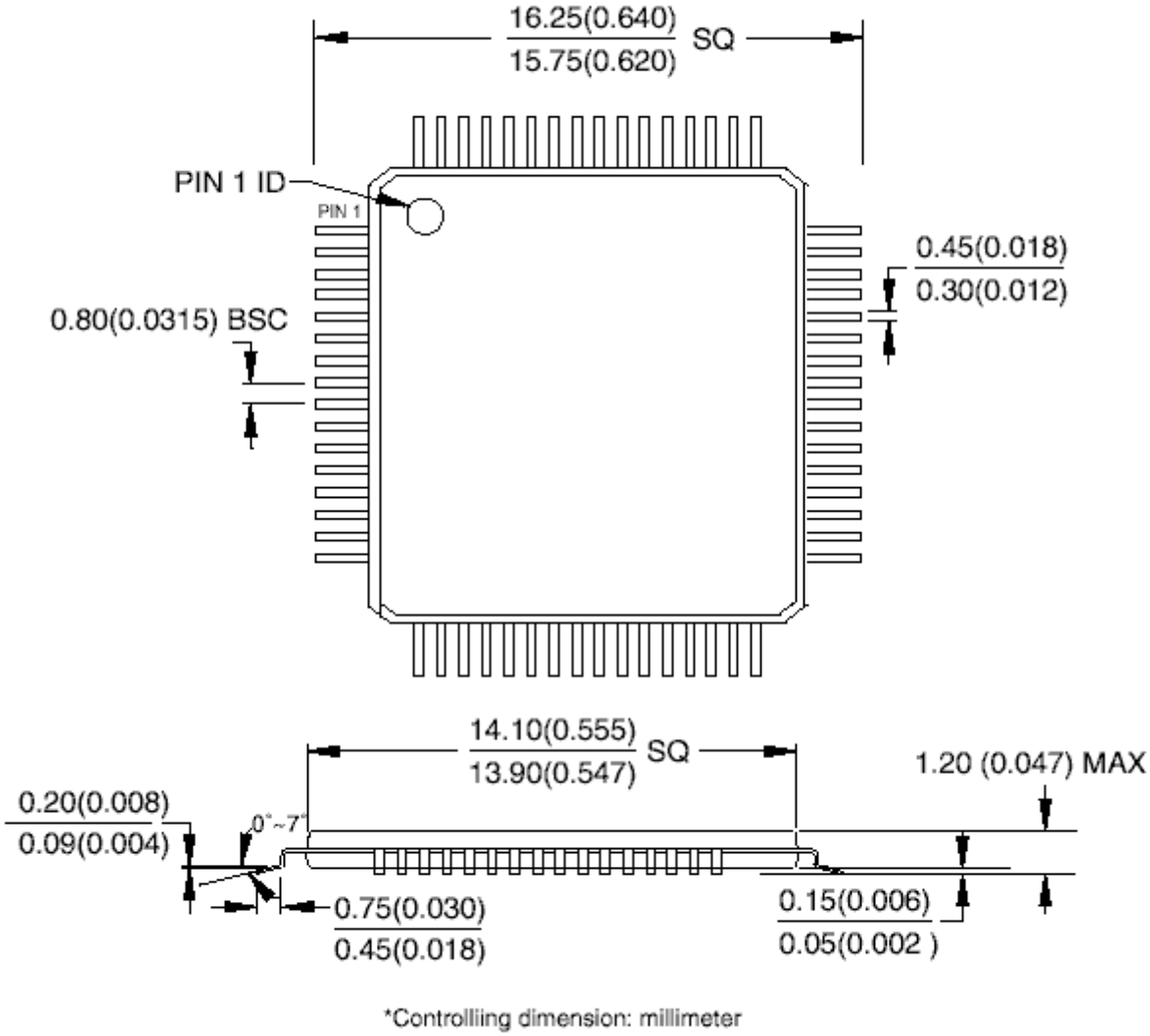
Mnemonik	Opis	Działanie	Znaczniki	Liczba Cykli
NOP	brak operacji		–	1
SLEEP	przejście w tryb uśpienia	patrz opis funkcji Sleep	–	1
WDR	wyzerowanie zegara układu watchdog	patrz opis układu WDR	–	1
BREAK	pułapka	tylko w trybie debugowania (On-Chip Debug)	–	–

UPORZĄDKOWANIE INFORMACJI

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
8	2.7 - 5.5V	ATmega128L-8AC	64A	Commercial (0°C to 70°C)
		ATmega128L-8AI	64A	Industrial (-40°C to 85°C)
16	4.5 - 5.5V	ATmega128-16AC	64A	Commercial (0°C to 70°C)
		ATmega128-16AI	64A	Industrial (-40°C to 85°C)

PAKIETY INFORMACJI

64 wyjścia, wąskie (1.0 mm) plastykowe poczwórne płaskie pakunki (TQFP)
wymiary całości 14*14mm, 2.0 mm odciski wiązek, 0.8mm podziałka
Wymiary w milimetrach i (calach)*
Standard JEDEC MS-026 AEB



ZMIANA ARKUSZA DANYCH DLA ATMEGA128

Proszę zauważyć, że numery stron, do których odnosi się ta sekcja należą do tego dokumentu.

1. *Poprawiony opis alternatywnych funkcji portu G*

Opis TOSC1 i TOSC2 na stronie 79 w „Opis alternatywnych funkcji portu G”

2. *dodane numery wersji JTAG dla rev F i G*

tabela 99na stronie 249

3. *dodane niektóre limity testów i danych wzorcowych*

usunięte niektóre z TBD w tabelach na stronach:
tabela 19 na stronie 46, tabela 20 na stronie 50, tabela 130 na stronie 301, „charakterystyki DC” na stronie 314, tabela 132 na stronie 316, tabela 135 na stronie 318, tabela 136 na stronie 320

4. *poprawione ”porządkowanie informacji” na stronie 347*
5. *dodane niektóre dane wzorcowe w sekcji „ATmega128 typowe wzorce...” na stronie 326*
6. *usunięty algorytm zostawienia JTAG w trybie programowania.*

Zobacz „pozostanie w trybie programowania” na stronie 311.

7. *dodany opis o tym jak wchodzić do rozszerzonych bezpieczników przez tryb programowania JTAG*

przejrzyj „Programowanie bezpieczników” na stronie 312 i „Odczyt bezpieczników i bitów blokujących” na stronie 313.