SerwoKontroler MAS-SC16A

dr inż. Maciej Sławiński

3 grudnia 2009

Pełna dokumentacja i projekt SerwoKontrolera (Wersja: 1.7beta)

Spis treści

1 Wstęp

1.1 Opis

Serwokontroler MAS-SC16A jest urządzeniem przeznaczonym do inteligentnego sterowania maszynami. Dokument zawiera szczegółowy opis oraz projekt serwokontrolera i jego oprogramowania. Oprogramowanie serwokontrolera to:

- oprogramowanie mikrokontrolera w urządzeniu
- aplikacja powalająca na pisanie, kompilację i zarządzanie pracą urządzenia
- funkcje realizujące komendy w programie MATLAB

Niniejszy dokument został podzielony na części. Poszczególne części dokumentu to dokumentacja dotycząca określonej tematyki powiązanej z budową, użytkowaniem i programowaniem urządzenia.
Wykaz i przeznaczenie poszczególnych części dokumentu:

- **Instrukcje**– instrukcje przeznaczone dla konkretnych użytkowników:
  - **Użytkownika**– standardowy użytkownik urządzenia
  - **Aplikacji MAS-SC16A**– użytkownik aplikacji zarządzającej pracą urządzenia i pozwalającą na jego programowanie
  - **Pakietu MATLAB**– użytkownik urządzenia w programie MATLAB
  - **Do zastosowań autorskich**– dla użytkowników realizujących we własnym zakresie program zarządzający pracą urządzenia
  - **Dla twórców SYSTEMU**– dla użytkowników realizujących autorską wersję SYSTEMU
- **Projekt układu**– dokumentacja projektu urządzenia
- **Oprogramowanie**– dokumentacja projektów oprogramowania
  - **Mikrokontroler**– znajdującego się w mikrokontrolerze urządzenia
  - **Aplikacja MAS-SC16A**– przeznaczona do pracy na komputerze PC
  - **MATLAB**– funkcje do zastosowań w programie MATLAB

1.2 Wersje

Wersje oprogramowania i układu:

- 1,0...1,6 wersje dotyczące eliminacji poszczególnych błędów urządzenia
- 1,7 dopisanie komendy sterującej grupą ośmiu serwomechanizmów i realizacja biblioteki do programu MATLAB

Część I
Instrukcje obsługi

W tej części dokumentacji znajdują się instrukcje obsługi dla poszczególnych zastosowań układu:

- Instrukcja użytkownika systemu
- Instrukcja obsługi aplikacji zarządzającej MAS-SC16A
- Instrukcja obsługi pakietu funkcji programu MATLAB
- Instrukcja do zastosowań autorskich
- Instrukcja dla twórców Systemu

2 Instrukcja obsługi urządzenia

2.1 Wstęp

MAS-SC16A to serwokontroler czyli urządzenie pozwalające na sterowanie pracą serwomechanizmów (PWM) jak również pracą układów o sterowaniu typu włącz/wyłącz. Układ posiada również wejścia analogowe, analogowo-cyfrowe (konfigurowalne) oraz wejścia przerwań systemowych. Zasilanie urządzenia może odbywać się poprzez wykorzystanie napięcia zasilającego serwomechanizmy bądź może być zrealizowane poprzez wewnętrzny stabilizator. Układ wyposażony jest w port RS232 realizujący komunikację z komputerem bądź innym urządzeniem.

2.2 Parametry urządzenia

Lp	Parametr	Symbol	MIN	NOMINAL	MAX	Jednostki
1	Napięcie zasilania	USR	4	5	6	V
2	Napięcie zasilania	UC	6	-	15	V
3	Prędkość złącza RS	$BaudRate$	4,8	115,2	230,4	kbps
4	Czas impulsów PWM	T_{PWM}	0	1,5	2,5	ms
5	Częstotliwość zegara	f_{osc}	-	11,0592	-	MHz
6	Wyjścia PWM	S	-	-	16	-
7	Wyjścia cyfrowe	DO	-	-	8	-
8	Wejścia analogowe	AI, ADI	-	-	8	-
9	Wejścia cyfrowe	ADI	-	-	6	-
10	Wejścia przerwań	INT	-	-	2	-

Tabela 1: Lista parametrów urządzenia

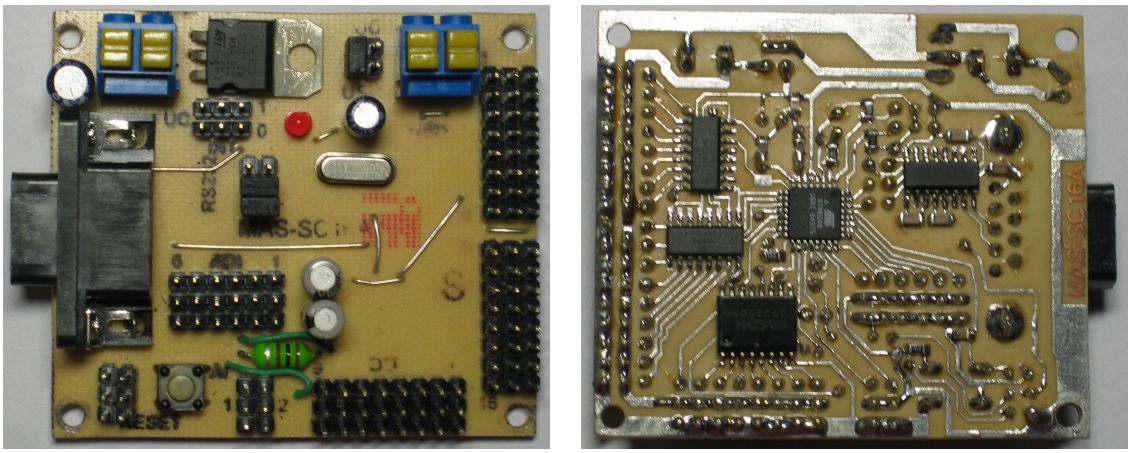
Dostępne wartości prędkości komunikacji RS232:
4800 bps, 9600 bps, 14,4 kbps, 19,2 kbps, 28,8 kbps, 38,4 kbps, 57,6 kbps, 115,2 kbps, 230,4 kbps.

2.3 Opis urządzenia

Urządzenie zmontowane jest na płytce drukowanej o wymiarach: 65mm x 55mm. Umieszczone są tam złącza 3-pin wszystkich wyjść i wejść układu, złącze RS232, złącza zasilania (UC, USR), złącza programowania sprzętowego oraz konfigurujące napięcie zasilania i port RS.

W Systemie zaimplementowano automatyczne przywracanie ustawień domyślnych portu RS. Realizowane jest to w momencie gdy układ zostanie uruchomiony przy sprzętowym anulowaniu identyfikacji Systemu (pinB.4(16)=0).

Sprzętowe pominięcie identyfikacji Systemu zostało również zaimplementowane w BootLoaderze. Identyfikacja Systemu jest pomijana w momencie podłączenia pin 16 (pinB.4) mikrokontrolera do napięcia zasilania.



Rysunek 1: Wygląd układu

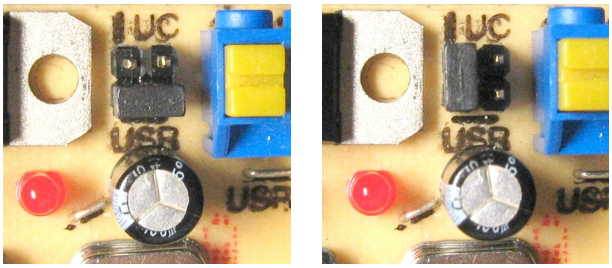
Na rysunku 1 przedstawiono wygląd płytki urządzenia gotowego do użycia. Po lewej stronie znajduje się widok złączy i zworek konfigurujących parametry sprzętowe układu, natomiast po stronie prawej umieszczone jest zdjęcie dolnej warstwy płytki z przylutowanymi elementami elektronicznymi.

2.4 Konfiguracja i podłączenie

2.4.1 Zasilanie

Zależnie od tego jak układ ma być zasilany podłączamy napięcie UC lub USR odpowiednio konfigurując złącze zasilania (rys. 2).

Jeżeli mikrokontroler zasilany jest napięciem UC wtedy zarówno wyjścia S1-S16 jak i DO1-DO8 mogą być zasilane niezależnie napięciem USR.



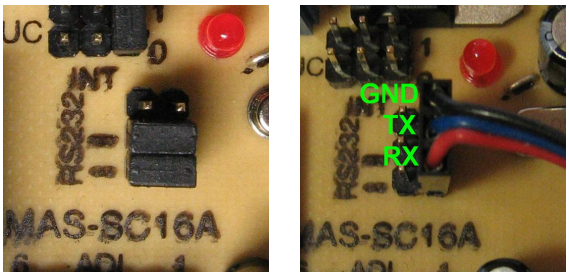
Napięcie USR Napięcie UC

Rysunek 2: Konfiguracja zasilania

2.4.2 Port RS232

Złącze pin-9 portu RS232 połączone jest z mikrokontrolerem poprzez układ MAX232. Układ ten zapewnia realizację połączenia z komputerem (poziomy napięć +12V/-12V). Jest również możliwe podłączenie dowolnego innego urządzenia wyposażonego w złącze RS232 o poziomach napięć (0V/5V).

Podłączenie portu RS232 realizowane jest poprzez przewód przedłużacza RSa. (NIE NULLMODEM)



Pin-9 Złącze RS (5V)

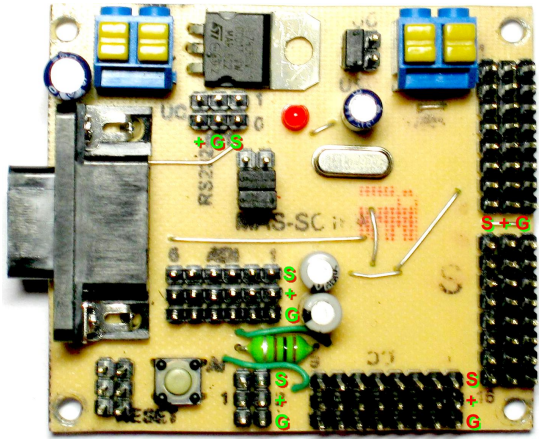
Rysunek 3: Port RS232

Na rysunku 3 przedstawiona została fizyczna realizacja połączenia portu RS232. Prócz fizycznej konfiguracji portu RS istotne jest również jakie prędkości są zaimplementowane w układzie.

Dopuszczalne wartości prędkości transmisji za pomocą portu RS to: 4800bps, 9600bps, 14,4kbps, 19,2kbps, 28,8kbps, 38,4kbps, 57,6kbps, 115,2kbps, 230,4kbps.

2.4.3 Wejścia i wyjścia układu

Rysunek 4 przedstawia konfigurację wyprowadzeń złączy wejść i wyjść urządzenia.



Rysunek 4: Konfiguracja złączy wejść i wyjść urządzenia

Złącza wejściowe opisane są kolorem zielonym natomiast wyjścia to kolor czerwony. Znaczenie symboli użytych w opisie:

- S - wyjście/wejście sygnałowe
- G - masa
- + - napięcie zasilania wyjścia-USR, wejścia-UC

2.5 Uruchomienie urządzenia

Po skonfigurowaniu zasilania i innych parametrów sprzętowych można podłączyć napięcie zasilające i tym samym uruchomić układ. Pierwszym trybem pracy jest BootLoader.

W trybie BootLoadera sprawdzana jest obecność Systemu w pamięci programu mikrokontrolera. Jeśli w układzie System został zapisany to jest on uruchamiany. Od strony użytkownika nie zostanie rozpoznane że mikrokontroler uruchomił się od BootLoadera ponieważ pierwszym całkowicie uruchomionym i zgłaszającym się trybem pracy będzie System. W przypadku, gdy w pamięci programu mikrokontrolera znajduje się również Program użytkownika wtedy System po uruchomieniu się rozpoczyna pracę programu użytkownika (tryb Program).

Jeśli układ jest podłączony do komputera to aplikacja MAS-SC16A otrzymuje informacje o trybie pracy w którym uruchomiony został układ i wyświetla stosowne informacje na ekranie. Aplikacja MAS-SC16A umożliwia również realizację wszystkich operacji programowych urządzenia. Szczegółowy opis aplikacji jest zamieszczony w rozdziale 2.6.

2.6 Program zarządzający

Aplikacja MAS-SC16A jest programem zarządzającym pracą urządzenia od strony programowej. Pierwsze to możliwość konfigurowania ustawień parametrów urządzenia takich jak:

- prędkość złącza RS
- parametr automatycznego uruchamiania programu użytkownika

Zmiana prędkości RSa odbywa się poprzez zmianę ustawień w oknie dialogowym "Ustawienia portu i komunikacji RS232". Okno to dostępne jest po wybraniu menu "Opcje->RS232". Po zmianie prędkości złącza RS232 należy zresetować urządzenie.
Parametr automatyczne uruchamianie programu użytkownika ustawiany jest w oknie dialogowym "Parametry urządzenia" dostępny w menu "Opcje->Parametry urządzenia". W oknie tym można włączać lub wyłączać automatyczne uruchamianie programu użytkownika.
W menu "Opcje->Ustawienia" definiuje się parametry pracy aplikacji takie jak:

- położenie pliku komend
- prędkość sprawdzania obecności urządzenia

Plik komend to plik w formacie "*.ini". Zapisane są w nim wszystkie komendy dostępne w aplikacji MAS-SC16A i zaimplementowane w mikrokontrolerze. Szczegółowy opis komend, pisania programów i ich realizacji jest opisany w rozdziale 2.8. Opisane są tam również komendy i instrukcje których zmienić nie można, gdyż są zaimplementowane na stałe w aplikacji MAS-SC16A.

Aplikacja realizuje również pełną obsługę trybów pracy urządzenia - menu "Tryb pracy". Menu "Tryby pracy->Sprawdź stan" realizuje zapytanie o aktualny tryb pracy urządzenia.

Pełna obsługa programów pisanych w aplikacji zawarta jest w menu "Wykonaj". Menu wykonaj zawiera również nadzór nad zawartością pamięci programu mikrokontrolera w zakresie Systemu i Programu.

Opis operacji realizowanych przez aplikację jest dostępny również z poziomu aplikacji w jej pomocy.

2.7 Pakiet funkcji programu MATLAB

Pakiet programu MATLAB zawiera wszystkie wymienione w rozdziale 2.8.2 funkcje SYSTEMU. Szczegółowy opis instrukcji zaimplementowanych w MATLABIE znajduje się w rozdziale 11.
Wykaz instrukcji dołączonych dodatkowo do pakietu MATLAB:

- **StartSC(Port,Prędkość)** – uruchomienie portu COM i uruchomienie urządzenia
- **StopSC** – wyłączenie urządzenia i zamknięcie portu COM
- **RunCommand(Komenda,Wynik)** – przesłanie komendy zgodnie z przyjętym protokołem transmisji
- **GetTryb** - zwraca aktualny tryb pracy urządzenia jako tekst
- **SetTryb(Tryb)** - przełącza urządzenie do pracy w podanym trybie

2.8 Programowanie układu

W rozdziale tym zawarty jest pełen opis zaawansowanego programowania urządzenia.
Zasada pisania i język programowania jest zbliżony do zasad obowiązujących w języku C.

- Każda linia programu kończy się średnikiem ';'.
- Istotna jest wielkość liter w poszczególnych komendach.
- Grupy komend znajdują się pomiędzy znakami '{' i '}'
- Operatory logiczne to: '<', '>', '<=', '>=', '==', '!='.
- Przypisanie wartości do zmiennej realizuje znak '='.
- Operacje matematyczne to: '+', '-', '*', '/'.

2.8.1 Komendy i dyrektywy niedefiniowalne

Zmienne definiowane po dyrektywie "Var:" w następujący sposób:
Var: zmienna1,zmienna2,...;
Do zmiennej można przypisać również wartość zwracaną przez funkcje zwracające wynik swego działania. Istotny jest tu fakt, iż nie definiuje się typu zmiennej a każda zmienna jest to wartość całkowita (integer) lub dwu bajtowa wartość w mikrokontrolerze (Program użytkownika).
Inna nazwa znajdująca się przed znakiem ':' prócz "Var" traktowana jest jako etykieta np:
Etykieta:
Praca programu rozpoczyna się od etykiety "Main:". Jeśli etykieta ta nie zostanie zdefiniowana wtedy początkiem jest pierwsza linia tekstu dokumentu. Etykiety "INT0" oraz "INT1" są przeznaczone do definiowania funkcji realizowanych po wystąpieniu przerwań int0 i int1.
Komenda realizująca przejście do danej etykiety to:
GoTo(Etykieta);

Instrukcja warunkowa to "if (warunek) komendy else komendy". Pozycja warunek to logiczne wyrażenie dające wynik true lub false. Pozycja komendy to jedna komenda zakończona średnikiem lub grupa komend zamieszczona między nawiasami '{' i '}'. Przykład:

```
Var:X;
X=2*3;
if(X>1){
Wait(10);
}
else
Wait(100);
```

Inną komendą warunkową jest "while (warunek) komendy". W nawiasie znajduje się dowolna komenda warunkowa (typu true/false). Pozycja komendy natomiast to jedna komenda zakończona średnikiem bądź lista komend umieszczonych między nawiasami " i ". Przykład:

```
Var: X;
X=0;
while(X<5)
Wait(10);
X=X+1;
```

Wywołanie funkcji realizują komendy "Call(Etykieta);" oraz "Run(Etykieta);". Różnica między tymi komendami polega na tym, iż "Call" nie zmienia wartości zmiennych a "Run" to wykonanie poleceń znajdujących się po tej etykiecie. Powrót z funkcji to komenda "Return();". Komenda ta zrealizowana z poziomu głównego aplikacji realizuje jej zakończenie analogicznie jak polecenie "Exit();".

Komendy kończące pracę:

"RetINT();" - zakończenie funkcji obsługującej przerwanie.

"Return();" - zakończenie funkcji.

"Exit();" - zakończenie programu.

Komenda kończąca pracę programu to "Exit();". Użycie tej komendy gwarantuje bezwarunkowe przerwanie pracy programu.

Komenda "Wait(czas);" to zatrzymanie pracy programu na czas[ms].

Polecenie "Reset(zmienna)" usuwa zmienną z pamięci układu.

Komenda	Opis	Uwagi
Var:	Definiowanie zmiennych.	-
Nazwa:	Etykieta o podanej Nazwie.	"Main:" - początek programu
GoTo	Przejście do Etykiety.	-
if	Instrukcja warunkowa.	-
while	Pętla warunkowa.	-
Run	Wykonanie poleceń po podanej etykiecie.	-
Call	Jak "Run" tylko nie zmienia zmiennych.	-
Return	Powrót z polecenia "Call" i "Run".	-
Exit	Bezwarunkowe przerwanie pracy programu.	W MK powrót do Systemu
Wait	Wstrzymanie pracy na czas[ms].	-
Reset	Usunięcie zmiennej z pamięci.	-

Tabela 2: Lista niedefiniowalnych poleceń

2.8.2 Komendy Systemowe

W rozdziale tym przedstawione są wszystkie komendy zaimplementowane w Systemie.

1. Komendy wejść analogowych:
   - AIxRead(numer); - odczyt wartości na wejściu analogowym AInumer.
2. Komendy wejść analogowo-cyfrowych:
   - ADIAxRead(numer); - odczyt analogowej wartości na wejściu ADInumer.
   - ADIDxGet(numer); - odczyt stanu na wejściu analogowo-cyfrowym ADInumer.
   - ADIDGet(); - odczyt stanu na wejściach ADI.
3. Komendy wyjść cyfrowych
   - DOxSet(nummer,stan) - ustawienie stanu wyjścia cyfrowego DOnumer=stan.
   - DOxGet(numer) - odczyt stanu wyjścia cyfrowego DOnumer.
   - DOSet(wartość) - ustawienie wartości/stanów wyjść cyfrowych DO=wartość.
   - DOGet() - odczyt wartości stanów cyfrowych DO.
4. Przerwania
   - INT0Get() - odczyt przerwania INT0.
   - INT0Clr() - wyczyszczenie przerwań INT0.
   - INT1Get() - odczyt przerwania INT1.
   - INT1Clr() - wyczyszczenie przerwań INT1.
5. Komendy serwomechanizmów

- ServoSet(numer,krok,czas,soft,pozycja) - zadanie pozycji serwa numer z zadanym krokiem lub gdy krok==0 w podanym czasie, parametr soft to łagodny start i stop - łagodny przyrost kroku (krok = krok +/- soft).
- ServaSet(Krok, Czas, Soft, Pozycja1, Pozycja2, Pozycja3, Pozycja4, Pozycja5, Pozycja6, Pozycja7, Pozycja8) - zadanie pozycji serw od numeru 1 do 8 z zadanym krokiem lub gdy krok==0 w podanym czasie, parametr soft to łagodny start i stop - łagodny przyrost kroku (krok = krok +/- soft).
- ServoRead(numer) - odczyt aktualnej pozycji serwa numer.
- ServoReadS(numer) - odczyt aktualnej pozycji serwa numer w stopniach.

6. Komendy obsługujące diodę
    - LED(stan) - włączenie (stan=1) lub wyłączenie (stan=0) diody.
    - LEDFlash(czas) - zapalenie diody na czas [ms].
7. Inne komendy
    - Send(znak) - komenda wysyła jeden znak z poziomu mikrokontrolera poprzez RS.

Uniwersalizm definiowania komend w aplikacji MAS-SC16A pozwala na zmianę nazw komend przedstawionych powyżej. Zmiana ich parametrów jak również wprowadzenie nowej komendy wymaga wprowadzenia stosownych zmian w Systemie.

3 Instrukcja obsługi programu MAS-SC16A

3.1 Wstęp

MAS-SC16A to aplikacja przeznaczona do pełnej obsługi serwokontrolera MAS-SC16A. Za pomocą aplikacji można uaktualniać i modyfikować w pełnym zakresie oprogramowanie urządzenia. Za pomocą komend menu "Wykonaj" realizowane są operacje zarządzające pracą układu. Konfiguracja i edycja parametrów to urządzenia to menu "Ustawienia" gdzie zaimplementowano również konfigurację parametrów aplikacji.

W dalszej części instrukcji opisane są podstawowe informacje o pracy urządzenia a następnie poszczególne komendy menu aplikacji.

3.2 Urządzenie

Urządzenie MAS-SC16A charakteryzuje się czterema trybami pracy:

1. BootLoader - Podstawowy tryb pracy urządzenia. Każde uruchomienie urządzenia rozpoczyna się od tego trybu mimo, iż jego uruchomienie nie jest zauważalne. Tryb BootLoader pozwala na zarządzanie zawartością pamięci programu mikrokontrolera wliczając zapis, aktualizację bądź usunięcie Systemu lub Programu użytkownika.

2. System - W trybie tym układ nic nie wykonuje utrzymując jedynie aktywną obsługę serwomechanizmów i wyjść cyfrowych. Kod źródłowy Systemu natomiast to program, gdzie zaimplementowana została cała obsługa wyjść i wejść urządzenia.

3. Komendy - Tryb w którym układ wykonuje komendy odbierane poprzez złącze RS232. Tryb ten wykorzystywany jest do wykonywania programu z poziomu aplikacji.

4. Program - Tryb oznaczający, iż aktualnie układ wykonuje program użytkownika zapisany w pamięci mikrokontrolera.

Menu "Tryb pracy" pozwala na realizację przejścia urządzenia do odpowiedniego trybu pracy. Przejście między każdym z trybów do dowolnego innego nie zawsze jest możliwe ponieważ nie każdy tryb jest zawsze dostępny i nie z każdego trybu pracy możemy przejście do dowolnego innego.

Pełne zarządzanie pamięcią programu mikrokontrolera urządzenia zrealizowane zostało poprzez komendy menu "Wykonaj".

3.3 Programowanie urządzenia

Programowanie urządzenia z poziomu aplikacji to nie tylko pełna obsługa pamięci programu mikrokontrolera urządzenia czy konfiguracja jego parametrów pracy. Programowanie z poziomu aplikacji to głównie pisanie inteligentnych programów realizujących urządzenia sterujące pracą inteligentnych maszyn.

3.3.1 Programy realizowane w aplikacji

Programy realizowane w aplikacji stanowią tekst zawierający ciąg komend realizujących poszczególne operacje zaimplementowane w mikrokontrolerze urządzenia. Styl i forma programów realizowanych w aplikacji to zasady obowiązujące w programach pisanych w języku "C".

Początek programu to etykieta "Main:". W przypadku, gdy w programie nie zostanie zdefiniowana etykieta "Main:" wtedy działanie programu rozpoczyna się od pierwszego wiersza. W programie jest również możliwe definiowanie obsługi przerwań sprzętowych. W celu zdefiniowania sekwencji komend obsługujących przerwania należy je umieścić po etykiecie "INT0:" oraz "INT1:". Powrót z przerwań realizuje komenda "RetINT();" lub "Return();".

Główne zasady dotyczące pisania programów w aplikacji:

- Koniec linii to ";".
- Wielkość liter jest istotna.
- Znak przyrównania wartości do zmiennej to "=".
- Znaki równań logicznych: ">", ">=", "==", "<=", "<", "!=".
- Znaki działań "+", "-", "*", "/".
- Etykieta to ciąg znaków alfanumerycznych zakończona znakiem ":".
- Etykieta "Var:" to dyrektywa po której definiowane są zmienne ("Var:x,y;" - zdefinowanie zmiennych x i y).
- Etykieta "asm:komendy" to dyrektywa po której definiowane są komendy assemblera.
- Komendy sprzętowe (niedefiniownalne) Systemu:
  1. GoTo(Etykieta); - Przejście do danej Etykiety.
  2. Run(Etykieta); - Wykonanie kodu programu znajdującego się po podanej etykiecie do momentu wystąpienia komendy "Return();".
  3. Call(Etykieta); - Podobnie jak Run z tą tylko różnicą iż wartości zmiennych po wykonaniu instrukcji "Return();" są takie same jak przed wykonaniem danej instrukcji.
  4. Return(); - Powrót po wywołaniu komend "Call" i "Run". W prypadku gdy nie nastąpiło takie wywołanie komenda realizuje zakończenie pacy programu użytkownika.
  5. Wait(Czas); - Wstrzymanie wykonywania programu na określony Czas[s].
  6. RetINT(); - Powrót z przerwania (może być też Return();)
  7. if(warunek)instrukcje else if(warunek) instrukcje else instrukcje - Komenda warunkowa.
  8. while(warunek)instrukcje - Pętla warunkowa.
- Komendy systemowe (definiowalne) - Zaimplementowane w urządzeniu komendy, które mogą być w dowolny sposób (nazwa, parametry) definiowane w aplikacji.

3.3.2 Obsługa programów

Po napisaniu programu można w pierwszej kolejności sprawdzić jego poprawność poprzez uruchomienie z poziomu aplikacji. Uruchomienie z poziomu aplikacji może odbywać się w sposób: Uruchom - ciągłe wykonywanie programu, Krok - wykonanie jednego kroku (jednej komendy programu). W przypadku poprawnego działania programu można go skompilować i umieści w pamięci mikrokontrolera by układ mógł pracować zupełnie niezależnie.

3.3.3 Kompilacja programu

Proces kompilacji programu polega na jego przetłumaczeniu na język assemblera a następnie jego skompilowanie za pomocą firmowego kompilatora firmy Atmel. Po pozytywnej realizacji procesu kompilacji i w przypadku, gdy urządzenie jest prawidłowo podłączone do komputera można kod wynikowy zapisać w pamięci programu mikrokontrolera. Po zapisaniu programu w pamięci programu mikrokontrolera może on być wykonywany w urządzeniu zupełnie niezależnie od jakichkolwiek urządzeń zewnętrznych.

W przypadku, gdy po kompilacji program nie został zapisany do pamięci mikrokontrolera jest możliwość późniejszej realizacji tej czynności. Wynikowy kod programu zapisywany jest w pliku "NazwaProgramu.hex" w formacie IntelHEX.

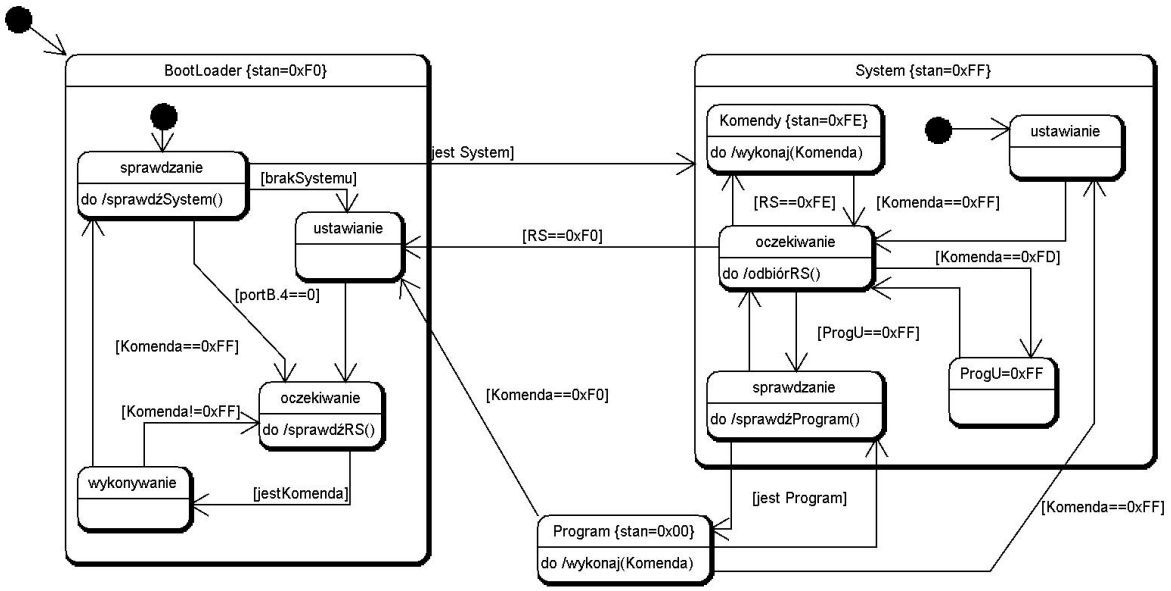
4 Instrukcja obsługi dla twórców własnych aplikacji zarządzających

4.1 Wstęp

Instrukcja ta to szczegółowy opis komunikacji urządzenia z komputerem. Pracę urządzenia można podzielić na cztery części w zależności od tego w jakim trybie pracy się ono znajduje: BootLoader, System, Komendy, Program. W dalszej części instrukcji opisana jest komunikacja we wszystkich trybach pracy urządzenia.

4.2 Omówienie pracy układu

Podstawowy i najbardziej precyzyjny opis pracy układu przedstawiony jest na diagramie stanów (rys. 5).



Rysunek 5: Diagram stanów mikrokontrolera

Zgodnie z informacjami zamieszczonymi na powyższym diagramie stanów pracę mikrokontrolera możemy podzielić na trzy główne tryby pracy :BootLoader, System i Program. Stan Komendy jest podstanem stanu System. To właśnie w stanie System zaimplementowany i w pełni realizowany jest stan Komendy.
Nie wszystkie jednak stany osiągalne są w każdym momencie racy układu.

W mikrokontrolerze brak jest przejść między niektórymi jego stanami.

Tryby Komendy oraz Program osiągalne są tylko i wyłącznie z trybu System. Tryb BootLoader natomiast jest nieosiągalny z trybu Komendy. Jedynie do trybu System można przejść z każdego innego trybu pracy w każdym momencie pracy układu.

4.3 Komendy i komunikacja

Wejściu do dowolnego trybu pracy mikrokontrolera towarzyszy potwierdzenie tej operacji poprzez wysłanie bajtu identyfikującego dany tryb. Wysłanie tego bajtu realizowane jest tylko wtedy, gdy dany tryb pracy został pomyślnie i do końca uruchomiony. Mimo, iż każde uruchomienie mikrokontrolera rozpoczyna sie od BootLoadera to nie zawsze wysyła on znak 0xF0 identyfikujący tryb pracy BootLoader. Dzieje się tak ponieważ znak identyfikujący tryb pracy wysyłany jest w momencie pełnego uruchomienia mikrokontrolera w danym trybie. Uruchomienie układu mimo, iż rozpoczyna się od BootLoadera nie oznacza wcale, iż BootLoader jest w pełni uruchamiany. BootLoader jest w pełni uruchomiony wtedy, gdy w pamięci programu mikrokontrolera nie ma Systemu.

4.3.1 BootLoader

Po wejściu w tryb BootLoadera układ wysyła znak identyfikujący tryb pracy (znak 0xF0). Komunikacja w trybie BootLoadera odbywa się zgodnie z następującym protokołem komunikacji:

- PC => Identyfikator komendy => MK
- PC => Argumenty => MK
- MK => Wynik/Wartość zwracana => PC

Tabela 3: Komendy dostępne w trybie BootLoader

ID	Opis	Argumenty	Wynik
0x00	Pytanie o tryb prazy	BRAK	0xF0
0x0A	Zmiana prędkości RSa	ID nowej prędkości	BRAK
'E'	Skasowanie Programu użytkownika i Systemu	BRAK	Wykonano 0x01
'e'	Skasowanie Programu użytkownika	BRAK	Wykonano 0x01
'R'	Odczyt Systemu	BRAK	Kod System
'r'	Odczyt Programu użytkownika	BRAK	Kod Programu
'w'	Zapis strony Programu lub Systemu do pamięci programu MK	AdresStrony, 64 x Dane	Wykonano 0x01
0xFF	Przejście do trybu System	BRAK	Tryb pracy

Tabela 4: Argumenty komendy zmieniającej prędkości złącza RS

Wartość	143	71	47	35	23	17	11	8	5	2
Pr. [kbps]	4,8	9,6	14,4	19,2	28,8	38,4	57,6	76,8	115,2	230,4

4.3.2 System

Po wejściu w tryb System układ wysyła znak identyfikujący tryb pracy (znak 0xFF). Komunikacja w trybie System odbywa się zgodnie z następującym protokołem komunikacji:

- PC => Identyfikator komendy => MK
- PC => Argumenty => MK
- MK => Wynik/Wartość zwracana => PC

Tabela 5: Komendy dostępne w trybie System

ID	Opis	Argumenty	Wynik
0x00	Pytanie o tryb prazy	BRAK	0xFF
0x01	Pytanie o automatyczny start programu użytkownika	BRAK	Wartość parametru
0x02	Definiowanie automatycznego startu programu użytkownika	Wartość parametru	Wykonano 0x01
'v'	Pytanie o wersje Systemu	BRAK	Numer wersji Systemu (TEXT)
0xF0	Przejście do trybu BootLoader	BRAK	Wykonano 0xF0
0xFG	Przejście do trybu Program	BRAK	Wykonano 0xFG
0xFE	Przejście do trybu Komendy	BRAK	Wykonano 0xFE

Wartości argumentu komendy 0x02:
0x00 - domyślne uruchamianie programu użytkownika
0xFF - domyślne pozostawanie w Systemie

4.3.3 Komendy

Po wejściu w tryb Komendy układ wysyła znak identyfikujący tryb pracy (znak 0xF0). Pytanie o tryb pracy to komenda 0x00. Odpowiedzią na znak 0x00 w trybie Komend jest 0xFE.
Koniec trybu Komendy do znak 0xFF. Po tym znaku realizowane jest przejście do trybu System czyli zakończone powodzeniem wykonanie tej komendy to potwierdzenie przejścia do trybu System (znak: 0xFF).
W trybie Komendy realizowana jest obsługa komend wysyłanych poprzez złącze RS232. Przesyłanie poleceń w trybie Komendy odbywa się zgodnie z następującym protokołem:

- PC => identyfikatorKomendy => MK
- PC => parametr1 => MK
- PC => parametr2 => MK
- PC => . . . => MK
- MK => sumaKontrolna => PC
- PC => 0x01 => MK - wykonaj
- MK => 0x01 => PC - wykonano

- MK => wynik/wartośćZwracana =>PC

Tabela 6: Komendy zaimplementowane i dostępne w trybie Komendy

ID	Opis	Argumenty	Wynik
0x00	Pytanie o tryb pracy	BRAK	0xF0
'a'	Odczyt wartości na wejściu analogowym AInumer	numer	wartość
'A'	Odczyt analogowej wartości na wejściu ADInumer	numer	wartość
'c'	Odczyt stanu na wejściu analogowo-cyfrowym ADInumer	numer	stan
'C'	Odczyt stanu na wejściach ADI	BRAK	stan
'D'	Ustawienie stanu wyjścia cyfrowego DOnumer=stan	numer, stan	BRAK
'd'	Odczyt stanu wyjścia cyfrowego DOnumer	numer	stan
'O'	Ustawienie wartości/stanów wyjść cyfrowych DO=wartość	wartość	BRAK
'o'	Odczyt wartości stanów cyfrowych wyjść DO	BRAK	stany
'i'	Odczyt przerwania INT0	BRAK	0/1
'p'	Wyczyszczenie przerwań INT0	BRAK	BRAK
'I'	Odczyt przerwania INT1	BRAK	0/1
'P'	Wyczyszczenie przerwań INT1	BRAK	BRAK
'S'	Zadanie pozycji serwa numer z zadanym krokiem w podanym czasie z soft startem	numer, krok, czas, soft, pozycja	BRAK
'M'	Zadanie pozycji serw o numerach 1...8 z zadanym krokiem w podanym czasie z soft startem	krok, czas, soft, Pozycja1, Pozycja2, Pozycja3, Pozycja4, Pozycja5, Pozycja6, Pozycja7, Pozycja8	BRAK
's'	Odczyt aktualnej pozycji serwa numer	numer	pozycja
'l'	Włączenie (stan=1) lub wyłączenie (stan=0) diody	stan	BRAK
'L'	Zapalenie diody na czas [ms]	czas	BRAK
0xFF	Przejście do trybu System	BRAK	Wykonano 0xFF

4.3.4 Program

Po wejściu w tryb Program układ wysyła znak identyfikujący tryb pracy (znak 0x00). Protokół komunikacji w trybie Program:

- PC => identyfikatorKomendy => MK
- MK => wynik/wartośćZwracana =>PC

Tabela 7: Komendy w trybie Program

ID	Opis	Argumenty	Wynik
0x00	Pytanie o tryb pracy	BRAK	0x00
0xF0	Przejście do trybu BootLoadera	BRAK	0xF0
0xFF	Przejście do trybu System	BRAK	0xFF

5 Instrukcja dla twórców Systemu

5.1 Wstęp

Instrukcja ta to szczegółowy opis wytycznych dla twórców Systemu.

5.2 Lokalizacja

Pamięć programu mikrokontrolera jest podzielona na trzy części:

- System (0x0000 - 0x06FF)
- Program (0x0700 - 0x1EFF)
- BootLoader (0x1F00 - 0x1FFF)

Według powyższych założeń System znajduje się w dolnej części pamięci programu mikrokontrolera. Można naruszyć strukturę pamięci programu przyjętą powyżej. Wymaga to jednak by nie przekraczać zakresu 0x0000 - 0x1EFF oraz zrealizować możliwość powrotu do BootLoadera. Istotne jest również to, iż kasowanie całej pamięci programu (oprócz BootLoadera) realizuje w BootLoaderze komenda 'E'.

5.3 Wytyczne i ograniczenia

Wytyczne i ograniczenia stawiane Systemowi zaprezentowane poniżej to warunki, które muszą być zrealizowane za względu na poprawność działania urządzenia i tworzonego Systemu.

- Sprzętowe zabezpieczenie uruchomienia BootLoadera - BootLoader automatycznie przy pierwszym uruchomieniu mikrokontrolera realizuje przejście do Systemu. Jeśli nie zostanie zrealizowane na początku Systemu zabezpieczenie sprzętowe powrotu do BootLoadera wtedy może nastąpić awaria układu. Możliwa ona jest do usunięcia tylko poprzez sprzętowe (serwisowe) programowanie mikrokontrolera. Niedostępne poprzez złącze RS232. (Np: pinB.4=1)
- Przejście do trybu BootLoader poprzez komendę RS - Możliwość programowej obsługi urządzenia.

Dodatkowe wytyczne i ograniczenia, których spełnienie się zaleca. Konieczne jest ich uwzględnienie w przypadku, gdy realizowana jest aktualizacja Systemu bądź pisany jest nowy System o działaniu podobnym do pierwotnie zrealizowanego.

- Położenie Systemu w zakresie pamięci programu mikrokontrolera od 0x0000 do 0x06FF.
- Prędkość interfejsu USART (RS232) zgodna ze zdefiniowaną w BootLoaderze - Lokalizacja parametru w pamięci EEPROM to 0x00. Wartości odpowiadające stosownym prędkością RSa podane zostały w instrukcji dla twórców aplikacji.

Jeśli System ma być kontynuacją pierwotnej jego wersji należy kontynuować bądź aktualizować czy udoskonalać bieżącą wersję Systemu. Dokumentacja projektu Systemu została zawarta w projekcie oprogramowania mikrokontrolera.

5.4 Wskazówki zaawansowanego programowania układu

System może być zrealizowany całkowicie dowolnie w oryginalnym kompilatorze producenta mikrokontrolera - firmy Atmel (aplikacja dostępna jest na stronie: "www.atmel.com"). Samo programowanie układu jest możliwe do zrealizowania poprzez złącze RS232. Wymaga to jednak spełnienia wytycznych dotyczących BootLoadera zaprezentowanych w rozdziale 5.3.
Jakiekolwiek zmiany w BootLoaderze wymagają serwisowego programowania mikrokontrolera. Prócz serwisowego programowania istotne jest wtedy również zrealizowanie nowego lub zaktualizowanego Systemu. Zmiana BootLoadera to tak naprawdę realizacja nowego oprogramowania dla omawianego urządzenia.

Część II
Projekt układu SerwoKontrolera

6 Projekt układu

6.1 Wytyczne i ograniczenia stawiane urządzeniu

SerwoKontroler to urządzenie przeznaczone do realizacji inteligentnych maszyn. Inteligentna maszyna natomiast musi być w szerokim zakresie programowalna czyli umożliwiająca implementację różnych algorytmów sterowania. Jeśli chodzi o elementy wykonawcze inteligentnych maszyn to sterowania ruchomymi elementami realizowane jest najczęściej poprzez serwomechanizmy. Serwomechanizm to urządzenie sterowane często sygnałem o modulacji szerokości impulsu (PWM). Wytyczne dla układu SerwoKontrolera:

- sterowania szesnastoma serwomechanizmami (wyjścia PWM),
- sterowanie ośmioma wyjściami cyfrowymi (typu Wł./Wył),
- pomiar wartości analogowych,
- pomiar sygnałów cyfrowych,
- wejścia rejestrujące zmianę stanu (przerwania).

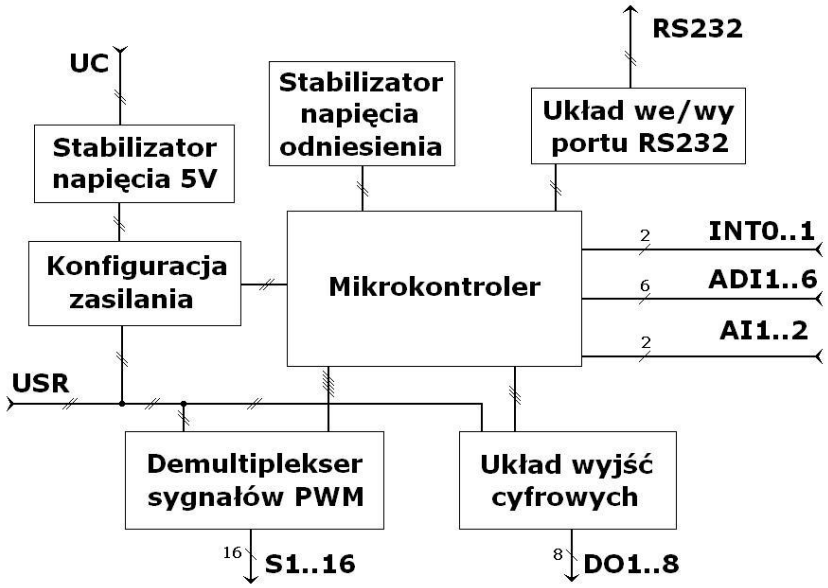
Układ SerwoKontrolera musi być wyposażony w port RS232 umożliwiający komunikację z komputerem czy też innymi urządzeniami lub systemami.

Projektowany SerwoKontroler musi być wyposażony w możliwość szerokiego zastosowania we wszelkiego rodzaju inteligentnych maszynach. Zgodnie z wymaganiami stawianymi urządzeniu musi to być układ programowalny z komunikacją i pełnym uniwersalizmem w zrealizowanym oprogramowaniu.

Głównym celem - wytyczną dla projektowanego SerwoKontrolera jest jego przeznaczenie: do zastosowań edukacyjnych czyli programowanie, sterowanie i testowanie dowolnych algorytmów pozwalających na realizację inteligentnych maszyn dowolnego rodzaju i budowy.

6.2 Schemat blokowy

Schemat blokowy przedstawiony na rysunku 6 prezentuje budowę modułową układu. Centralnym elementem układu jest mikrokontroler. Źródło zasilania mikrokontrolera to konfigurowalny wybór między napięciem zasilającym serwomechanizmy USR lub stabilizowanym napięciem zasilania UC (5V). Stabilizator napięcia odniesienia to moduł składający się z elementów LC tłumiących wszelkie oscylacje napięć zasilania i odniesienia przetwornika AC. Komunikacja układu zrealizowana została poprzez złącze RS232 z wykorzystaniem układu dopasowującego poziomy napięć złącza RS232 w komputerze do wartości 0V, 5V (poziom napięć w mikrokontrolerze). INTx, ADIx oraz AIx to wejścia podłączone bezpośrednio do mikrokontrolera. Wyjścia cyfrowe DOx buforowane są poprzez układ zbudowany na 8-bitowym rejestrze przesuwającym z kluczowanym wyjściem (układ typu 74HC595). Wartości określające stany wyjść cyfrowych przesyłane są



Rysunek 6: Schemat blokowy SerwoKontrolera

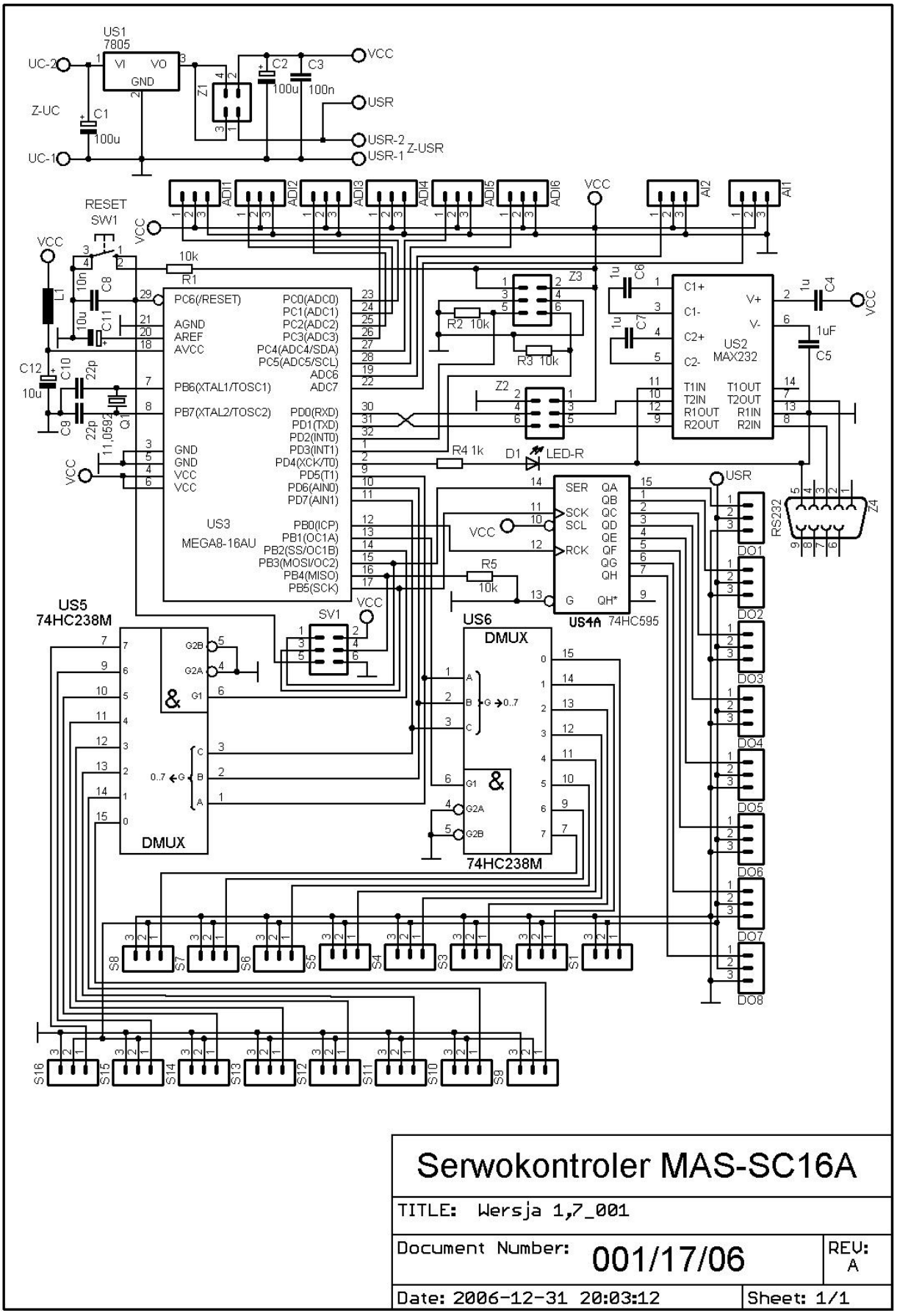
do tego układu poprzez wyjście MOSI mikrokontrolera. Ostatnią ważną częścią układu jest moduł demultiplekserów wybierających jedno z 8 wyjść sterujących kolejnymi serwomechanizmami.

Sterowanie serwomechanizmami odbywa się poprzez wykorzystanie sprzętowego generatora impulsów PWM. Sygnał sterujący serwomechanizmu to impuls o długości 0,5...2,5ms taktowany częstotliwością 50Hz. Częstotliwość 50Hz to 20ms. Natomiast
20ms/maksymalny_czas_impulsu_sterującego=20/2,5=8
Świadczy to o tym, że bez żadnych strat w sposób ciągły można korzystając z jednego wyjścia PWM wysterować maksymalnie 8 serwomechanizmów. W zastosowanym mikrokontrolerze zaimplementowane są dwa wyjścia PWM rejestru T1: OC1A oraz OC1B. Dwa wyjścia licznika T1 pozwalają na sprzętową realizację sterowania 16 serwomechanizmami.

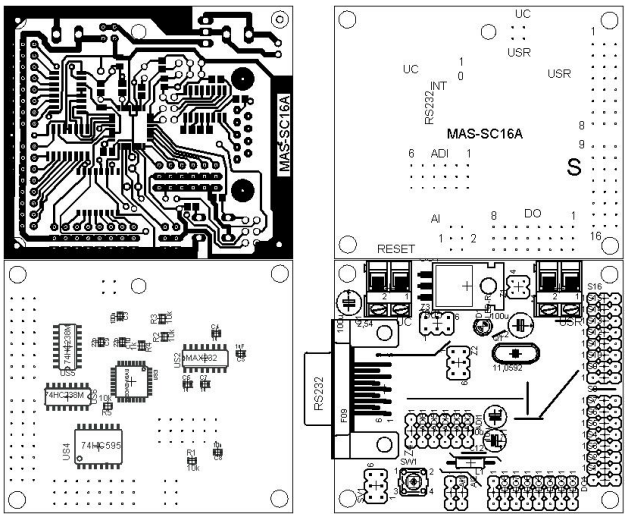
6.3 Schemat ideowy

Rysunek 7 przedstawia schemat ideowy układu MAS-SC16A. Schemat ideowy jest układem odpowiadającym przedstawionemu w poprzednim rozdziale schematowi blokowemu.
Główną jednostkę urządzenia stanowi mikrokontroler ATmega8-16AU. Poszczególne bloki urządzenia zostały zrealizowane na następujących elementach:

1. Blok zasilania: stabilizator napięcia 7805
2. Blok stabilizatora napięcia odniesienia: elementy LC
3. Blok we/wy RS232: układ MAX232



Rysunek 7: Schemat ideowy SerwoKontrolera



Rysunek 8: Płytka drukowana

4. Blok wyjść cyfrowych: układ 74HC595
5. Blok wyjść serwomechanizmów: układy 74HC238

6.4 Główne cechy układu

1) Złącze 4-pin ZW1 przeznaczone do konfiguracji napięcia zasilania.
2) Złącze 6-pin Z2 łączące mikrokontroler ze złączem RS232
3) Zegar rezonatora kwarcowego mikrokontrolera: 11,0592MHz.
4) Przycisk RESET.
5) Rejestr B:

- 0: sygnał kluczujący wyjścia DO rejestru przesuwającego
- 1: wyjście OC1A sterujące grupą serwomechanizmów 1-8
- 2: wyjście OC1B sterujące grupą serwomechanizmów 9-16
- 3: wyjście MOSI służące do przesyłania wartości DO poprzez SPI
- 4: NIE PODŁĄCZONE
- 5: wyjście zegara do przesyłu danych do wyjść DO poprzez SPI
- 6, 7: rezonator kwarcowy

6) Rejestr C:

- 0 - 5: konfigurowalne wejścia analogowe lub cyfrowe ADI

7) Rejestr D:

- 0, 1: wejście i wyjście portu RS232
- 2, 3: wejścia sygnałów generujących przerwania INT0, INT1

- 4: wyjście sterujące diodą LED.
- 5 - 7: wyjścia określające wyjścia demultiplekserów sterujących serwomechanizmami

8) Wejścia analogowe AI1(ADC6), AI2(ADC7)

6.5 Wykaz elementów

Lista elementów:
Rezystory SMD-M0805:

R1	$10k\Omega$	R2	$10k\Omega$	R3	$10k\Omega$	R4	$1k\Omega$

Kondensatory ([1]-Elektrolityczny, [2]-SMD C0805):

C1	$100\mu F/16V^1$	C2	$100\mu F/6V^1$	C3	$100nF^2$	C4	$1\mu F^2$
C5	$1\mu F^2$	C6	$1\mu F^2$	C7	$1\mu F^2$	C8	$10nF^1$
C9	$22pF^2$	C10	$22pF^2$	C11	$10\mu F/6V^1$	C12	$10\mu F/6V^1$

Złącza:

Z1	zworka 2x2	Z2	zworka 3x2	Z3	3x2
Z4	9-pin RS	Z5-Z36	3x1	Z-UC, Z-USR	Zas. 2-pin

Wszystkie złącza 2x2, 3x2 i 3x1 to listwa 50x1 (3szt.)
Układy scalone (SMD):

US1	7805	US2	MAX232	US3	AT mega 8-16AU
US4	74HC595	US5	74HC238	US6	74HC238

Inne:

L1	cewka $10nH$	Q1	kwarc $11,0592MHz$	SW1	przycisk monostab.
D1	dioda LED				

Elementy wykonawcze:
Płytka drukowana
Zworki 2-pin (3szt.)

6.6 Ustawienia sprzętowe mikrokontrolera

Poniżej przedstawiono konfigurację bitów rejestrów ustawień sprzętowych mikrokontrolera.

UWAGA : Wartość 1 oznacza bit niezaprogramowany.

6.6.1 Rejestr ustawień pamięci programu i danych

Bit	Wartość	Opis
BLB12 BLB11	0 0	SPM LPM for BootLoader are not allowed from Application
BLB02 BLB01	1 1	No restriction for SPM LPM
LB2 LB1	1 1	No memory lock

6.6.2 Bity rejestru hFuse

Bit	Wartość	Opis
RSTDISBL	1	PC6 = RESET
WDTON	1	WDT enabled by WDTCR
SPIEN	0	SPI enabled
CKOPT	0	Frequency > 8MHz
EESAVE	1	EEPROM not preserved
BOOTSZ1	1	BOOTSZ='11'
BOOTSZ0	1	(BootLoader size = 128 words = 4 pages)
BOOTRST	0	RESET vector (Boot Loader)

6.6.3 Bity rejestru lFuse

Bit	Wartość	Opis
BODLEVEL	1	BOD trigger level
BODEN	0	Brown out Detector Enabled (low power Reset)
SUT1	1	Delay from RESET CKSELo=1
SUT0	1	65ms (crystal resonator)
CKSEL3	1	
CKSEL2	1	Freq > 8MHz
CKSEL1	1	CKOPT = 0
CKSEL0	1	

Część III
Oprogramowanie SerwoKontrolera

MAS-SC16A to układ przeznaczony do inteligentnego sterowania serwomechanizmami. Oprócz serwomechanizmów można za jego pomocą realizować sterowanie typy Wł./Wył. Można również podłączając na odpowiednie wejścia sygnały analogowe bądź cyfrowe realizując pomiar sygnałów z różnych czujników zewnętrznych.

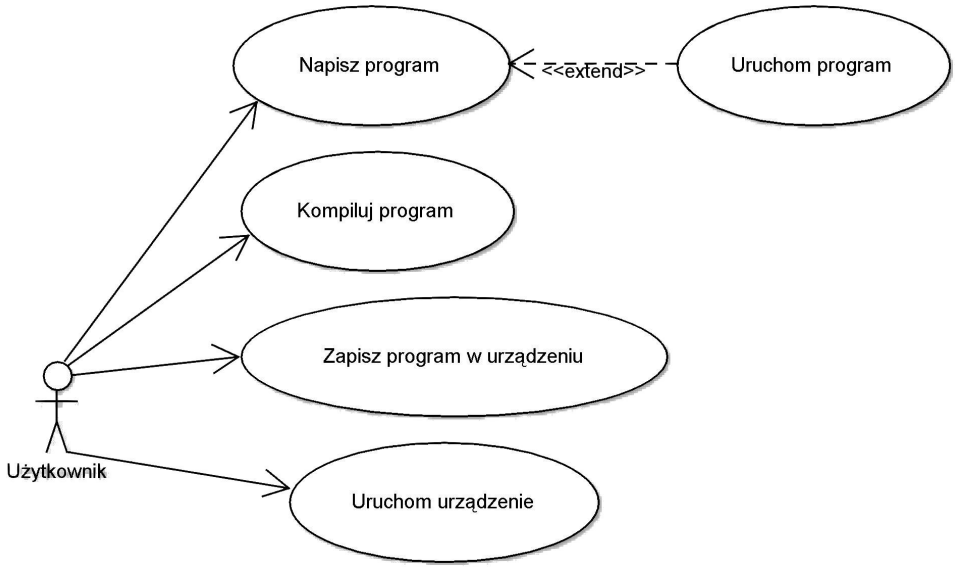
Najważniejszym założeniem projektowym dla oprogramowania jest implementacja inteligentnego programowania układu. Wskazana jest realizacja oprogramowania w bardzo uniwersalny sposób z możliwością aktualizacji, zmiany i modyfikacji za pośrednictwem złącza RS232.

Oprogramowanie układu zostało podzielone na dwie główne części:

- Mikrokontroler
- Komputer PC

Obie części oprogramowania to narzędzia realizujące między innymi komunikację pomiędzy komputerem a SerwoKontrolerem. Zasady i protokoły komunikacji zostały omówione dla poszczególnych programów oddzielnie.

Rysunek 9: Ogólny opis działania urządzenia

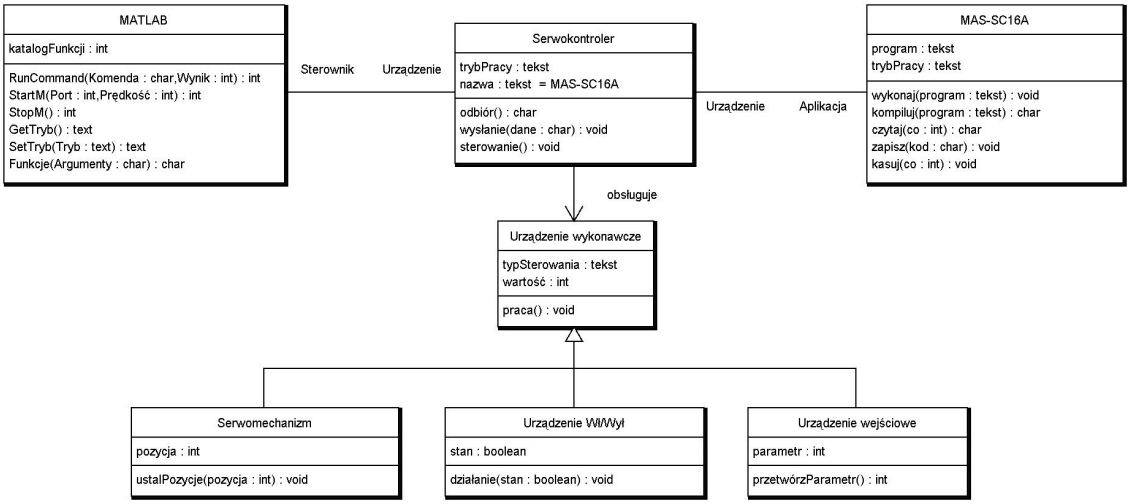


Główny ciąg zdarzeń:
- Użytkownik pisze program korzystając z jego uruchamiania
- Użytkownik kompiluje program
- Użytkownik zapisuje program
- Użytkownik uruchamia urządzenie.

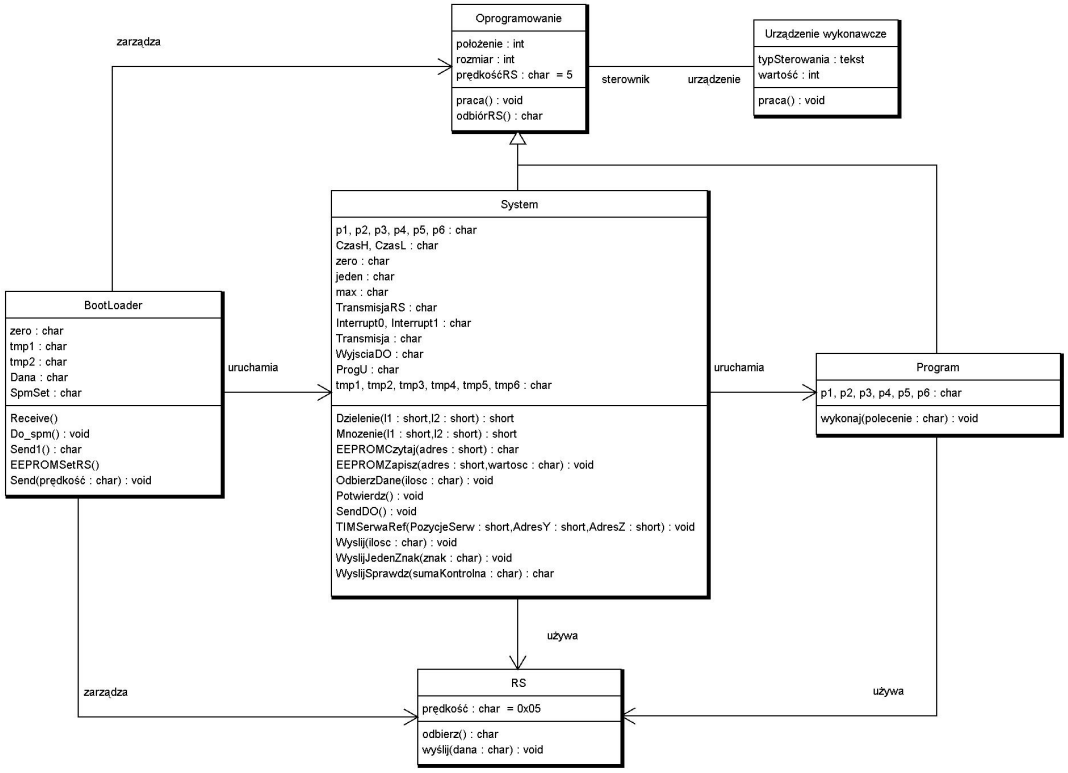
Alternatywny ciąg zdarzeń:
- Użytkownik pisze program korzystając z jego uruchamiania
- Użytkownik kompiluje program
- Użytkownik nie zapisuje programu

Alternatywny ciąg zdarzeń:
- Użytkownik pisze program korzystając z jego uruchamiania
- Użytkownik kompiluje program bez powodzenia
Alternatywny ciąg zdarzeń:
- Użytkownik pisze program i uruchamia go tylko z poziomu aplikacji

Rysunek 10: Serwokontroler



7 Mikrokontroler

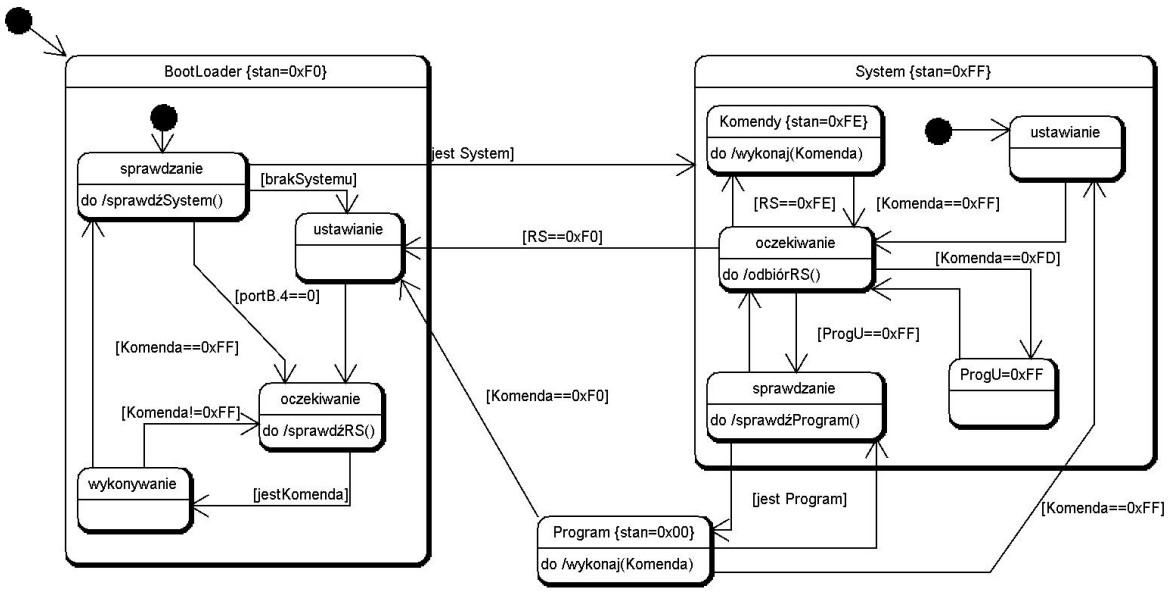


Rysunek 11: Obiekty mikrokontrolera

Pamięć programu mikrokontrolera została podzielona na trzy części:

- BootLoader
- System
- Program

BootLoader
to górny obszar pamięci programu mikrokontrolera (0x1F00-0x1FFF). Program tam się znajdujący służy do zapisu zarówno Systemu jak i Programu użytkownika do pamięci programu.
System
to dolny obszar pamięci programu mikrokontolera (0x0000-0x037F). Program tam się znajdujący obsługuje pracę urządzenia. To w nim zaimplementowane są wszystkie Komendy oraz obsługa podstawowych elementów układu.
Program
to środkowy obszar pamięci programu mikrokontrolera (0x0380-0x1EFF). W tej części pamięci programu jest umieszczany program użytkownika.



Rysunek 12: Stany pracy mikrokontrolera

Na rysunku 12 został przedstawiony diagram stanów opisujący pracę oprogramowania mikrokontrolera. Zaprezentowane tam komendy, stany oraz przejścia między poszczególnymi blokami programowymi zostały zaimplementowane w oprogramowaniu mikrokontrolera.

Ważną cechą zaimplementowanego oprogramowania jest brak przejść między niektórymi stanami mikroprocesora.

Tryby Komendy oraz Program osiągalne są tylko i wyłącznie z trybu System. Tryb BootLoader natomiast jest nieosiągalny z trybu Komendy. Niedostępność przejścia do niektórych trybów pracy została również zaimplementowana w oprogramowaniu przeznaczonym na komputery PC.
Trybem w którym uruchamiany jest mikrokontroler jest BootLoader. Tryb BootLoader przy starcie sprawdza czy System jest w pamięci programu mikrokontrolera i jeśli tak to uruchamiany jest System. Dlatego właśnie po uruchomieniu układu z zainstalowanym Systemem pierwszym trybem jest System. Jako pierwszy może również uruchamiać się tryb Program. Ma to miejsce wtedy, gdy System i Program znajdują się w pamięci programu mikrokontrolera oraz w Systemie będzie ustawiony parametr automatycznego przejścia do Programu użytkownika. Mimo, iż pierwszym zgłaszającym się nie zawsze jest BootLoader to jednak od niego rozpoczyna się praca mikrokontrolera.

7.1 BootLoader

7.1.1 Kontekst

BootLoader to program znajdujący się w górnej części pamięci programu mikrokontrolera. W programie tym zrealizowane jest zarządzanie pamięcią programu mikrokontrolera. BootLoader za pomocą złącza RS obsługuje komendy:

- 0x00 odpowiedź na pytanie o tryb pracy (zwraca 0xF0)
- 0x0A zmiana prędkości RSa (argumenty: 143=4800bps, 71=9600bps, 47=14,4kbps, 35=19,2kbps, 23=28,8kbps , 17=38,4kbps, 11=57,6kbps, 8=76,8kbps, 5=115,2kbps, 2=230,4kbps)
- 'E' skasowanie Systemu i Programu użytkownika (zwraca 0x01 gdy ukończono)
- 'e' skasowanie Programu użytkownika (zwraca 0x01 gdy ukończono)
- 'R' odczyt Systemu (zwraca cały System)
- 'r' odczyt Programu użytkownika (zwraca cały Program użytkownika)
- 'w' zapis strony Programu, Systemu do pamięci programu układu (argumenty: AdresStrony,Dane1,Dana2,... - po wykonaniu zwraca 0x01)
- 0xFF przejście do trybu System (jeśli System jest obecny)

Protokół komunikacji:
PC=>komenda=>MK
PC=>argumenty=>MK
MK=>wynik/wartoscZwracana=>PC
Po wejściu w tryb BootLoader układ wysyła 0x0F (numer aktualnego trybu pracy).

7.1.2 Wymagania

Rozdział zawiera opis wymagań stawianych realizowanemu SerwoKontrolerowi w trybie BootLoader.

Tabela 8: Definicje akronimy i skróty

Termin	Objaśnienie	Synonimy (nie zalecane)
BootLoader	Program realizujący zapis odczyt i kasowanie pamięci procesora układu.	System, Program
System	Program realizujący obsługę urządzenia i zawierający implementację funkcji.	Program
Program	Program napisany, skompilowany czy też uruchamiany praz użytkownika.	BRAK
Serwomechanizm	Elektronicznie sterowany układ poruszający wałkiem obrotowym w zakresie $0...180^{\circ}$	Serwo

Układ jest odpowiednikiem serwokontrolera zaprezentowanego na stronie:

"http://www.lynxmotion.com"
Zbudowany jest on na mikrokontrolerze firmy ATMEL ATmega8. Dostępna pełna dokumentacja na stronie:

"http://www.atmel.com"

BootLoader to program rozpoczynający pracę mikrokontrolera. Na początku swojej pracy musi on dokonać identyfikacji Systemu. Jeśli System znajduje się w pamięci programu mikrokontrolera powinien zostać uruchomiony. Należy tu zaznaczyć, iż w przypadku, gdy do trybu BootLoader przechodzi się z Systemu należy pominąć identyfikację i przejście do Systemu. Po dokonaniu identyfikacji Systemu i stwierdzeniu jego braku realizowana jest konfiguracja mikrokontrolera. Konfiguracja mikrokontrolera to ustawienie prędkości złącza RS. Następnie wysłanie informacji o rozpoczęciu pracy w trybie BootLoader. Główną pętlą programu jest odbiór i obsługa komend przekazywanych przez komputer za pomocą złącza RS. Istotną informacją jest to, iż właśnie BootLoader zarządza pamięcią mikrokontrolera w zakresie Systemu i Programu.

W dalszej części rozdziału przedstawiono specyficzne wymagania stawiane programowi BootLoader.

Wymagania funkcjonalne

Nazwa	Receive
Opis	Odbiór danych poprzez RSa.
Dane wejściowe	BRAK
Źródło danych wejściowych	BRAK
Wynik	Funkcja oczekująca na daną z RSa. Odbiera tą daną i zwraca jako wynik.
Warunek wstępny	BRAK
Warunek końcowy	Została odebrana jakaś dana.
Efekty uboczne	BRAK
Powód	Realizacja komunikacji mikrokontroler komputer

Nazwa	Send
Opis	Wysłanie znaku
Dane wejściowe	Znak
Źródło danych wejściowych	Rejestr tmp1
Wynik	Wysłanie danej poprzez port RS.
Warunek wstępny	Wolny bufor portu RS mikrokontrolera
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Realizacja komunikacji mikrokontroler komputer

Nazwa	Send1
Opis	Wysłanie 0x01
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	Wysłanie znaku 0x01
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Wysłanie znaku potwierdzającego wykonanie komendy

Nazwa	EnableRWW
Opis	Uruchomienie właściwości RWW
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	Ustawienie właściwości odczytu pamięci programu mikrokontrolera podczas zapisu
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Uruchomienie właściwości RWW po zapisaniu strony programu do pamięci programu mikrokontrolera

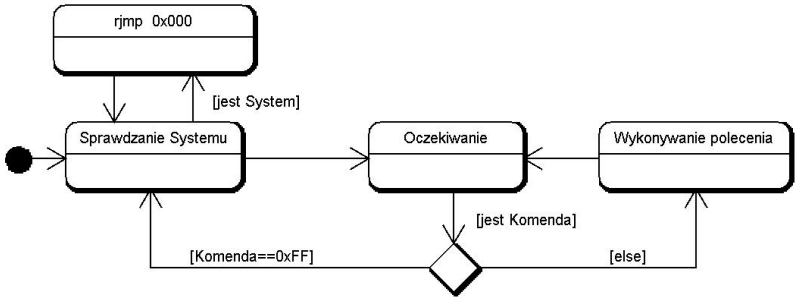
Nazwa	Do_spm
Opis	Zapis lub kasowanie pamięci programu mikrokontrolera
Dane wejściowe	Konfiguracja zapisu/kasowania strony danych, adres i 16 bitowe dana
Źródło danych wejściowych	Informacje odebrane poprzez złącze RS
Wynik	Zapis do określonego miejsca (STOS lub pamięć programu) danej/strony danych programu lub Kasowanie strony pamięci programu mikrokontrolera
Warunek wstępny	Dostępna pamięci programu - SPMEN w rejestrze SPMCR
Warunek końcowy	BRAK
Efekty uboczne	Zmiana pamięci programu
Powód	Realizacja obsługi zarządzania zawartością pamięci programu mikrokontrolera

Wymagania niefunkcjonalne

Lp	Wymaganie	Opis	Motywacja	Uwagi
1	Zgodność złącza RS232 ze standardem PC	Wartości i parametry złącza RS232 zgodne z prędkościami i parametrami dostępnymi w komputerach PC	Realizacja uniwersalnej obsługi urządzenia	Prędkości zgodne z wartościami dostępnymi w mikrokontrolerze i komputerze PC
2	Automatyczne przejście do Systemu	Przejście do Systemu, gdy jest on obecny.	Przejście do Systemu na starcie programu.	Przejście niezależne od poprawności Systemu.

7.1.3 Projekt programu

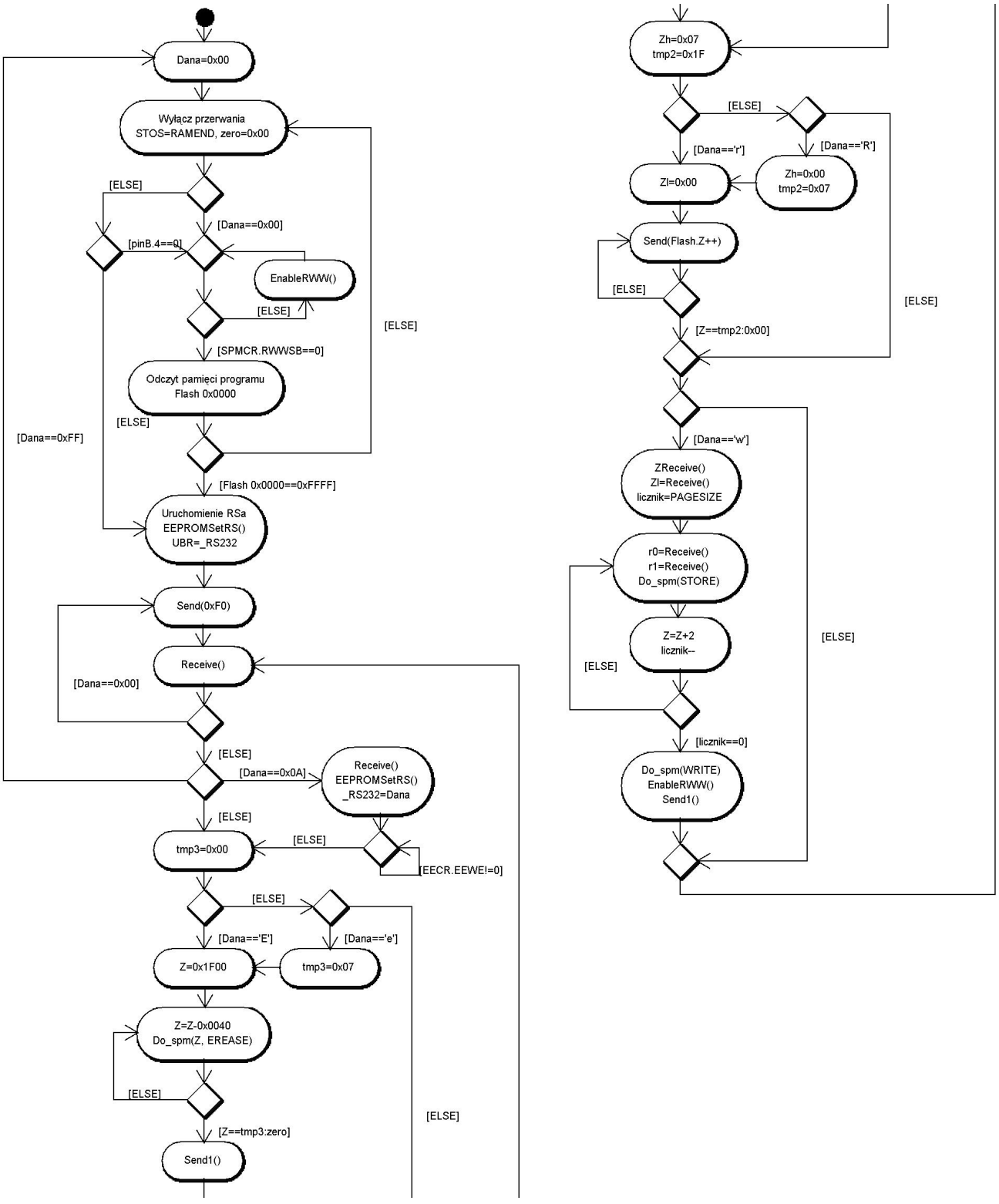
Na rysunku 13 przedstawiony jest diagram stanów opisujący pracę mikrokontrolera w trybie BootLoader w odniesieniu do całego oprogramowania mikrokontrolera. Na diagramie stanów widoczne jest przejście do Systemu na początku pracy programu BootLoader. Przejście to realizowane jest tylko wtedy, gdy w pamięci programu mikrokontrolera pod adresem 0x0000 znajduje się inna wartość niż 0xFFFF.



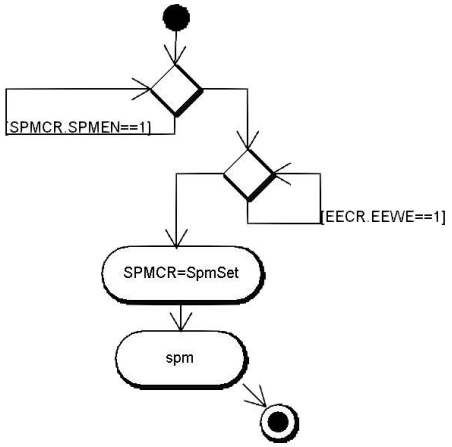
Rysunek 13: Stany pracy mikrokntrolera w zakresie trybu BootLoader

7.1.4 Program BootLoader

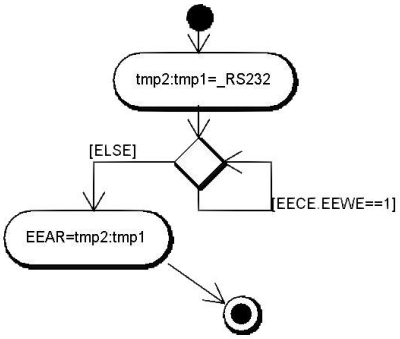
Rysunek 14 przedstawia diagram czynności opisujący główny program BootLoadera w mikrokontrolerze. Kolejne rysunki przedstawiają diagramy czynności opisujące szczegóły realizacji funkcji BootLoadera.



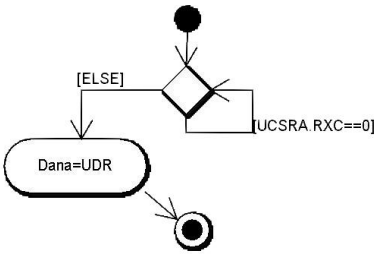
Rysunek 14: BootLoader



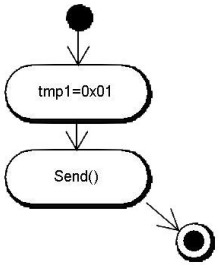
Rysunek 15: Funkcja Do_spm



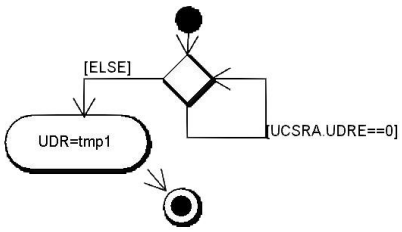
Rysunek 16: Funkcja EEPROMSetRS



Rysunek 17: Funkcja Receive



Rysunek 18: Funkcja Send1



Rysunek 19: Funkcja Send

7.2 System

7.2.1 Kontekst

System to program znajdujący się w dolnej części pamięci programu mikrokontrolera. W programie tym zrealizowane są wszystkie funkcje dostępne w urządzeniu. Wśród tych funkcji należy wymienić zaimplementowane funkcje użytkownika obsługujące wejścia i wyjścia urządzenia, obsługę programu użytkownika oraz podstawowe polecenia dostępne w tym trybie pracy. To w Systemie zrealizowana jest podstawowa obsługa komend RS w trybie Program oraz cała obsługa trybu Komendy. Komendy jest podtrybem trybu System. Program to tryb pracy dla którego obsługę komend RS zrealizowano w Systemie. W Systemie za pomocą złącza RS obsługiwane są komendy:

- 0x00 odpowiedź na pytanie o tryb pracy (zwraca 0xFF w trybie System, 0xFE w trybie Komendy, 0x01 w trybie Program)
- 0x01 pytanie o ustawienia zezwolenia na program użytkownika i uruchamianie trybu Komendy (System)
- 0x02 definiowanie zezwolenia na program użytkownika i uruchamianie trybu Komendy (System)
- 'v' pytanie o wersję Systemu (dwuznakowa wartość)
- 0xF0 przejście do trybu BootLoader (System i Program)
- 0xFG przejście do trybu Program (System)
- 0xFE przejście do trybu Komendy (System)
- 0xFF przejście do trybu System (Komendy, Program)

Protokoły komunikacji w poszczególnych trybach pracy:
System:
PC=>komenda=>MK
PC=>argumenty=>MK
MK=>wynik/wartoscZwracana=>PC
Komendy (protokół przesyłania komend):
PC=>identyfikatorKomendy=>MK
PC=>parametr1, parametr2, ...=>MK
MK=>sumaKontrolna=>PC
PC=>0x01(wykonaj)=>MK
MK=>0x01(wykonano)=>PC
MK=>wynik/wartośćZwracana=>PC
Program:
PC=>komenda=>MK
MK=>wynik/wartośćZwracana=>PC

Po wejściu w każdy z omawianych trybów pracy układ wysyła identyfikator trybu który został uruchomiony: System: 0xFF, Komendy: 0xFE, Program: 0x00.

7.2.2 Wymagania

Rozdział zawiera opis wymagań stawianych realizowanemu SerwoKontrolerowi w trybach System, Komendy i Program.

System to program w którym zrealizowano całą obsługę wszystkich zadań stawianych urządzeniu. To w Systemie zaimplementowane zostały wszystkie funkcje i procedury użytkowe realizujące inteligentne zadania stawiane urządzeniu.

W dalszej części rozdziału przedstawiono specyficzne wymagania stawiane programowi System.

Wymagania funkcjonalne

Nazwa	EXT_INT0
Opis	Obsługa przerwania INT0.
Dane wejściowe	Interrupt0
Źródło danych wejściowych	Rejestry MK
Wynik	Zwiększenie liczby przerwań o jeden.
Warunek wstępny	BRAK
Warunek końcowy	Interrupt0≤0xFF
Efekty uboczne	BRAK
Powód	Rejestracja liczby wystąpienia przerwań systemowych mikrokontrolera.

Nazwa	EXT_INT1
Opis	Obsługa przerwania INT1.
Dane wejściowe	Interrupt1
Źródło danych wejściowych	Rejestry MK
Wynik	Zwiększenie liczby przerwań o jeden.
Warunek wstępny	BRAK
Warunek końcowy	Interrupt1≤0xFF
Efekty uboczne	BRAK
Powód	Rejestracja liczby wystąpienia przerwań systemowych mikrokontrolera.

Nazwa	TIM1_CAPT
Opis	Funkcja obsługująca przerwania Timera1 realizującego obsługę sterowania serwomechanizmami.
Dane wejściowe	PozycjeSerw, numerSerwa.
Źródło danych wejściowych	Pamięć mikrokontrolera i jego rejestry.
Wynik	Przełączenie aktualnie obsługiwanego serwa i aktualizacja pozycji serwa.
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	Zmiana parametrów serw.
Powód	Dynamiczna obsługa równoległej obsługi 16 serwomechanizmów.

Nazwa	TIM0_OVF
Opis	Funkcja wykonuje czynności realizujące obsługę komendy Wait.
Dane wejściowe	CzasL, CzasH
Źródło danych wejściowych	Rejestry mikrokontrolera
Wynik	Zmniejszenie wartości czasu komendy Wait
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	Zmniejszanie wartości CzasL i CzasH
Powód	Realizacja komendy Wait.

Nazwa	SPI_STC
Opis	Funkcja finalizująca aktualizację wyjść cyfrowych.
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	Potwierdzenie przesłania wartości wyjść cyfrowych.
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Uruchomienie aktualizacji wartości wyjść cyfrowych.

Nazwa	USART_RXC
Opis	Funkcja obsługująca komunikację
Dane wejściowe	ProgU
Źródło danych wejściowych	Rejestry mikrokontrolera
Wynik	Odbiór informacji przekazywanej portem RS232
Warunek wstępny	ProgU==1
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Obsługa portu RS232

Nazwa	Dzielenie
Opis	Dzielenie Funkcja realizująca dzielenie.
Dane wejściowe	tmp2:tmp1, tmp4:tmp3
Źródło danych wejściowych	Rejestry mikrokontrolera
Wynik	W rejestrach tmp2:tmp1 wartość będąca wynikiem tmp2:tmp1/tmp4:tmp3
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Realizacja dzielenia dwóch liczb

Nazwa	EEPROMCzytaj
Opis	Funkcja odczytująca pamięć EEPROM
Dane wejściowe	Adres: tmp2:tmp1
Źródło danych wejściowych	Rejestry mikrokontrolera
Wynik	Wartość znajdująca się w pamięci EEPROM pod podanym adresem
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	Nie dotyczy
Powód	Realizacja odczytu wartości z pamięci EEPROM

Nazwa	EEPROMZapisz
Opis	Funkcja zapisująca pamięć EEPROM
Dane wejściowe	Adres, Wartość
Źródło danych wejściowych	Adres=tmp2:tmp1, Wartość=tmo3
Wynik	Zapisanie Wartości w pamięci EEPROM pod Adresem
Warunek wstępny	BRAK
Warunek końcowy	Zapis został zrealizowany
Efekty uboczne	BRAK
Powód	Realizacja zapisu wartości do pamięci EEPROM mikrokontrolera

Nazwa	Mnozenie
Opis	Funkcja realizująca mnożenie
Dane wejściowe	Liczba1, Liczba2
Źródło danych wejściowych	Liczba1=tmp2:tmp1, Liczba2=tmp4:tmp3
Wynik	tmp2:tmp1=liczba1*liczba2
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Realizacja operacji mnożenia

Nazwa	OdbierzDane
Opis	Funkcja odbierająca dane wysłane przez RSa
Dane wejściowe	Ilość danych
Źródło danych wejściowych	tmp1
Wynik	Odbiór danych poprzez port RS w podanej ilości.
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	Zapisanie danych na stos.
Powód	Odbiór pakietu danych

Nazwa	Potwierdz
Opis	Potwierdzenie wykonania komendy
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	Wysłanie przez RSa znaku 0x01 potwierdzającego wykonanie komendy
Warunek wstępny	Nie dotyczy
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Potwierdzenie wykonania komendy

Nazwa	SendDO
Opis	Funkcja sterująca wyjściami cyfrowymi
Dane wejściowe	Bajt określający wartości wyjść cyfrowych
Źródło danych wejściowych	WyjsciaDO
Wynik	Zadanie wartości wyjść cyfrowych zgodnych z bitami danej wejściowej
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	Uruchomienie przesłania danej przez port SPI
Powód	Sterowanie wyjściami cyfrowymi

Nazwa	TIMSerwaRef
Opis	Funkcja obsługująca aktualizację pozycji serw
Dane wejściowe	Adres pozycji aktualnie obsługiwanego serwa, Adres parametrów serwa, PozycjeSerw, PozycjeSerwZadane, Krok, KrokZadany, Soft
Źródło danych wejściowych	Rejestr Y, rejestr Z, pamięć SRAM mikrokontrolera
Wynik	Aktualizacja pozycji serwa
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	Aktualizacja parametrów serwa
Powód	Dynamiczna obsługa procesu sterowania serwami

Nazwa	Wyslij
Opis	Funkcja wysyłająca dane na rs
Dane wejściowe	Ilość zmiennych, Dana1, Dana2, ...
Źródło danych wejściowych	tmp1, STOS
Wynik	Wysłanie danych poprzez port RS
Warunek wstępny	UCSRA.UDRE==1
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Przesyłanie danych portem RS

Nazwa	WyslijJedenZnak
Opis	Funkcja wysyłająca jeden znak
Dane wejściowe	Znak
Źródło danych wejściowych	tmp1
Wynik	Wysłanie jednego znaku portem RS
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Zautomatyzowanie przesyłu jednego bajtu.

Nazwa	WyslijSprawdz
Opis	Funkcja sprawdzająca sumę kontrolną przez RSa
Dane wejściowe	Suma kontrolna
Źródło danych wejściowych	tmp1
Wynik	tmp1- potwierdzenie
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Realizacja potwierdzenia poprawności sumy kontrolnej wykonywanej komendy

Nazwa	Komendy
Opis	Funkcja realizująca komendy w zakresie systemu
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	BRAK
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Obsługa komend odbieranych poprzez port RS

Funkcje użytkowe

Nazwa	AIxRead
Opis	Funkcja zwraca wartość tmp2:tmp1 będącą wynikiem pomiaru wejść analogowych AI.
Dane wejściowe	nrWejścia
Źródło danych wejściowych	p1
Wynik	Pomiar wejścia analogowego o podanym numerze zawarty w rejestrach tmp2:tmp1
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Pomiar wejść analogowych

Nazwa	ADIAxRead
Opis	Funkcja realizująca pomiar wejść analogowych ADI
Dane wejściowe	numer wejścia ADI
Źródło danych wejściowych	p1
Wynik	Funkcja zwraca wartość będącą wynikiem pomiaru wartości analogowej sygnału na wejściu o podanym numerze
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Funkcja pozwalająca na realizacje pomiaru wartości sygnału analogowego na wejściu ADI o podanym numerze

Nazwa	ADIDGet
Opis	Funkcja zwracająca stan wejść ADI
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	tmp1 - bajt stanu wejść ADI
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Realizacja pomiaru wartości sygnałów cyfrowych na wejściach ADI

Nazwa	ADIDxGet
Opis	Funkcja zwracająca stan wejścia ADIx
Dane wejściowe	nr wejścia ADI
Źródło danych wejściowych	p1
Wynik	tmp1 - wartość 0 lub 1 odpowiadająca wejściu o podanym numerze
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Realizacja identyfikacji stanu na wejściu ADI o podanym numerze

Nazwa	DOGet
Opis	Funkcja zwracająca stan wyjść cyfrowych DO
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	tmp1 - stan wyjść cyfrowych DO
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Implementacja funkcji zwracającej stan wszystkich wyjść cyfrowych DO

Nazwa	DOSet
Opis	Funkcja ustalająca stan wyjść cyfrowych DO
Dane wejściowe	Stan
Źródło danych wejściowych	p1
Wynik	Ustawienie stanu wyjść cyfrowych DO
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Ustawianie stanu wszystkich wyjść cyfrowych DO

Nazwa	DOxGet
Opis	Funkcja zwracająca stan wyjścia cyfrowego DOx
Dane wejściowe	numer wyjścia
Źródło danych wejściowych	p1
Wynik	odczyt stanu wyjścia cyfrowego DO o podanym numerze
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Odczyt stanu wyjścia cyfrowego DO

Nazwa	DOxSet
Opis	Funkcja ustalająca stan wyjścia cyfrowego DOx
Dane wejściowe	numer wyjścia, stan
Źródło danych wejściowych	p1, p2
Wynik	Zmiana stanu wyjścia cyfrowego DO o podanym numerze
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Realizacja obsługi zmiany stanu pojedynczych wyjść cyfrowych

Nazwa	INT0Clr
Opis	Wyzerowanie liczby przerwań INT0
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	Wyzerowanie liczby przerwań INT0
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Obsługa przerwań INT0

Nazwa	INT0Get
Opis	Funkcja zwracająca informacje czy wystąpiło przerwanie INT0
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	Dekrementacja liczby przerwań INT0
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Obsługa przerwań INT0

Nazwa	INT1Clr
Opis	Wyzerowanie liczby przerwań INT1
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	Wyzerowanie liczby przerwań INT1
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Obsługa przerwań INT1

Nazwa	INT1Get
Opis	Funkcja zwracająca informacje czy wystąpiło przerwanie INT1
Dane wejściowe	BRAK
Źródło danych wejściowych	Nie dotyczy
Wynik	Dekrementacja liczby przerwań INT1
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Obsługa przerwań INT1

Nazwa	ServoRead
Opis	Funkcja zwracająca aktualną pozycję serwa
Dane wejściowe	numer serwa
Źródło danych wejściowych	p1
Wynik	tmp2:tmp1 - aktualna pozycja serwa
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Odczyt pozycji serwa

Nazwa	ServoSet
Opis	Funkcja ustalająca pozycję serwa z uwzględnieniem kroku i czasu
Dane wejściowe	numer serwa, krok, czas, soft, pozycja
Źródło danych wejściowych	p1, p2, p3, p4, p6:p5
Wynik	Ustalenie parametrów serwa
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	Zmiana parametrów serwa
Powód	Inteligentna obsługa serwomechanizmów

Nazwa	LEDSet
Opis	Funkcja włączająca bądź wyłączająca diodę
Dane wejściowe	Nowy stan diody
Źródło danych wejściowych	p1
Wynik	Zapalenie bądź zgaszenie diody
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	BRAK
Powód	Obsługa diody

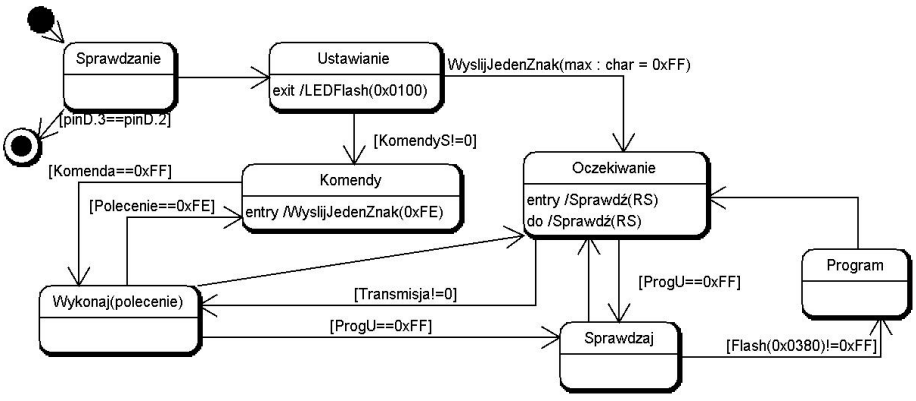
Nazwa	LEDFlash
Opis	Zapalenie diody na określony czas
Dane wejściowe	Czas [ms]
Źródło danych wejściowych	p2:p1
Wynik	Impuls diody przez podaną długość czasu
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	Zatrzymanie urządzenia
Powód	Obsługa diody

Nazwa	Wait
Opis	Funkcja realizująca pauzę
Dane wejściowe	Czas w ms
Źródło danych wejściowych	p2:p1
Wynik	Zatrzymanie programu na określony czas
Warunek wstępny	BRAK
Warunek końcowy	BRAK
Efekty uboczne	Zatrzymanie programu na określony czas
Powód	Realizacja pauzy

Wymagania niefunkcjonalne

Lp	Wymaganie	Opis	Motywacja	Uwagi
1	Zgodność złącza RS232 ze standardem PC	Wartości i parametry złącza RS232 po stronie mikrokontrolera zgodne z prędkościami i parametrami dostępnymi w komputerach PC	Realizacja uniwersalnej obsługi urządzenia	Prędkości zgodne z wartościami dostępnymi w mikrokontrolerze i komputerze PC
2	Sprzętowe zabezpieczenie powrotu do trybu BootLoader	Po zrealizowaniu pewnej operacji System na początku sprawdza jej obecność i nie rozpoczyna pracy. Układ wraca do trybu BootLoader.	Ochrona przed awarią Systemu.	BRAK
3	Ustawienie domyślnych parametrów złącza RS przy sprzętowym anulowaniu Systemu	W momencie, gdy wystąpi sprzętowe przerwanie Systemu układ ustawia domyślne wartości dla złącza RS232 (115,2kbps)	Realizacja zabezpieczenia	Zostanie wprowadzona nowa prędkość złącza RS232

7.2.3 Projekt programu

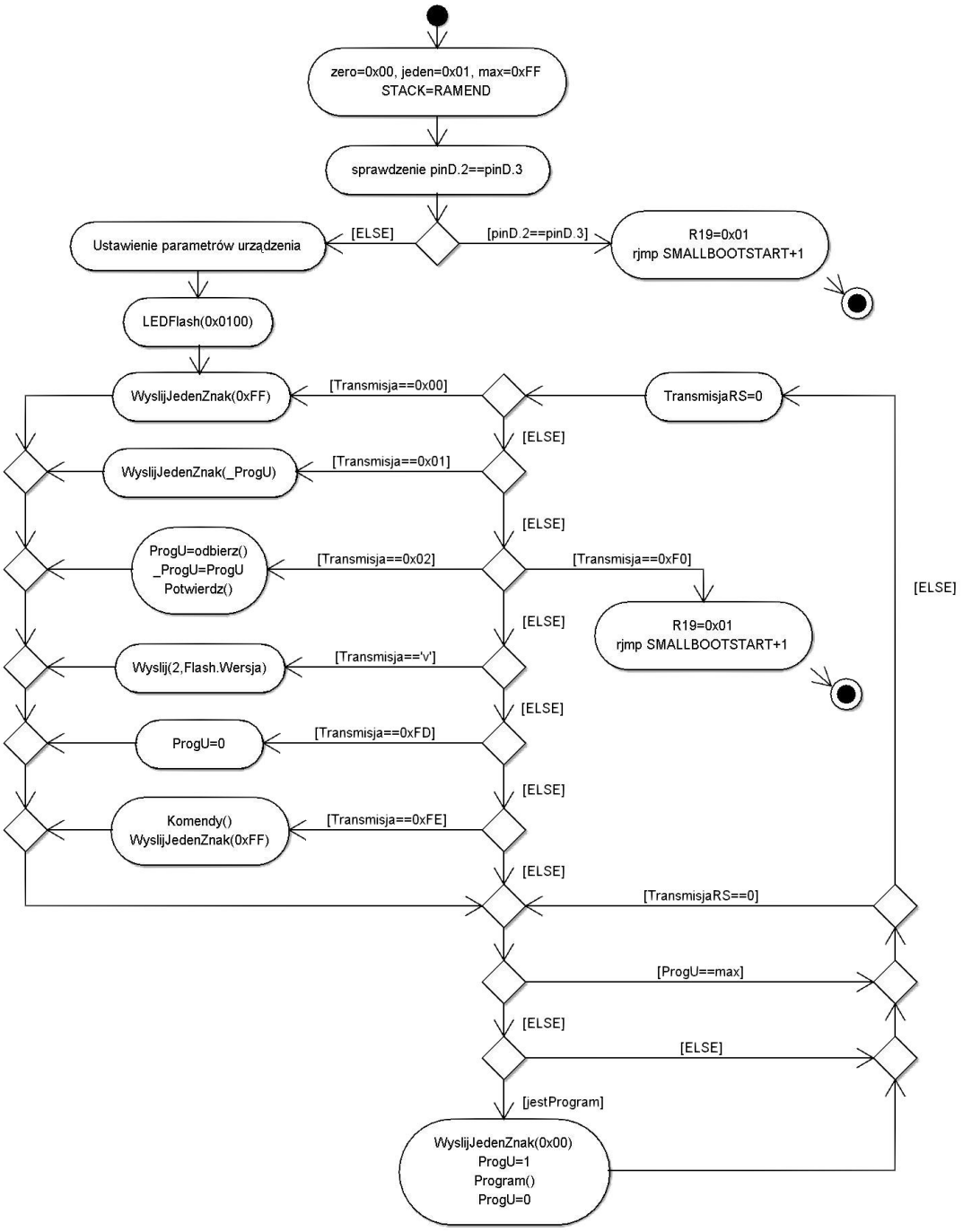


Rysunek 20: Stany pracy układu w trybie System

Na rysunku 20 przedstawiony jest diagram stanów opisujący pracę mikrokontrolera w trybie System w odniesieniu do całego oprogramowania mikrokontrolera. Na diagramie stanów widoczna jest sprzętowa realizacja automatycznego przerwania Systemu na początku jego uruchamiania. Przerwanie uruchamiania Systemu realizowane w tym momencie to operacja bezwzględnie przerywająca działanie mikrokontrolera w trybie

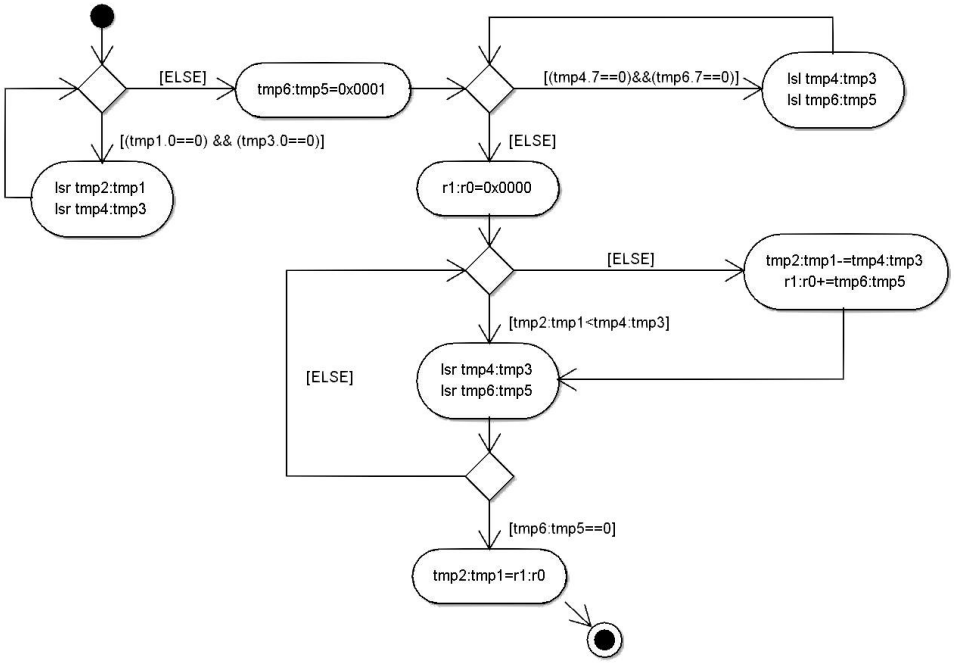
System i automatyczne przejście do trybu BootLoader. Istotne jest to, iż BootLoader powinien w tym momencie być uruchomiony z pominięciem identyfikacji Systemu i tym samym przejścia do niego.

7.2.4 Program System

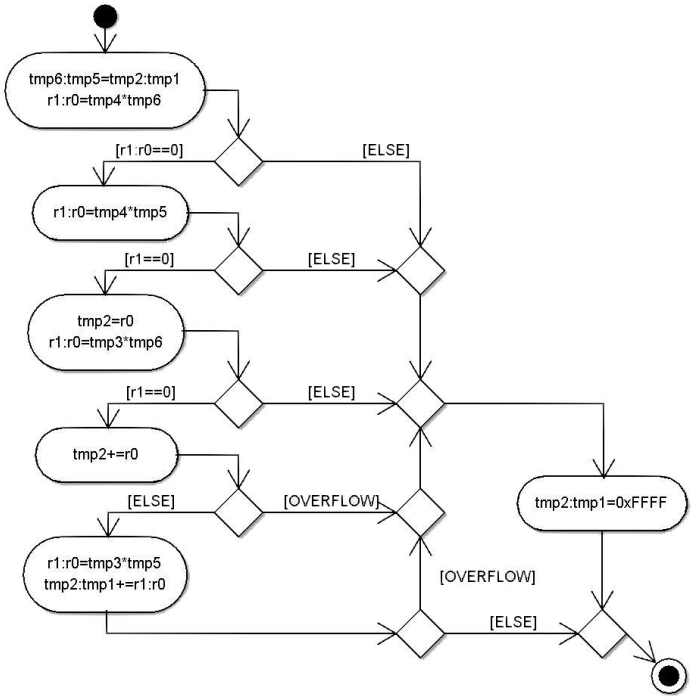


Rysunek 21: Systemu

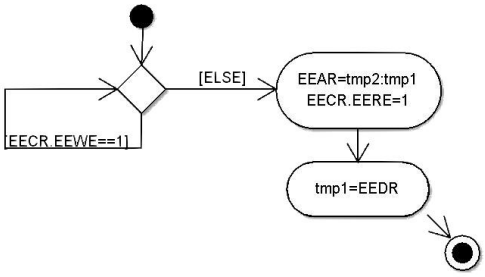
Rysunek 21 przedstawia diagram czynności opisujący System mikrokontrolera. Kolejne rysunki przedstawiają diagramy czynności opisujące szczegóły realizacji funkcji Systemu.



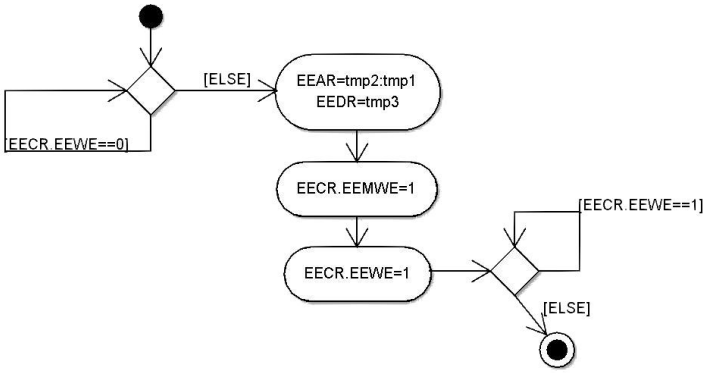
Rysunek 22: Funkcja Dzielenie



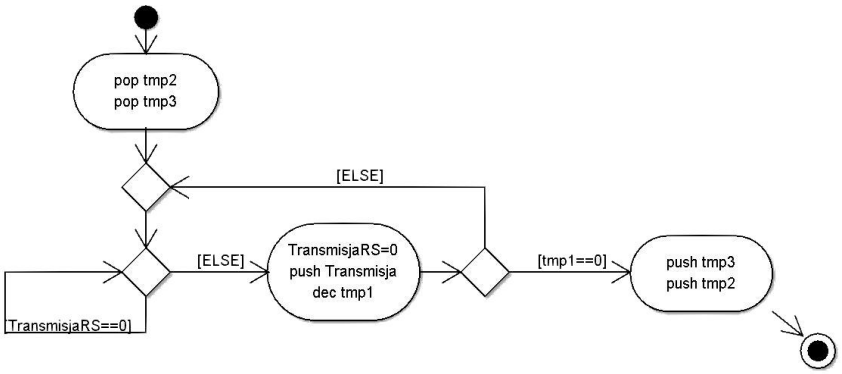
Rysunek 23: Funkcja Mnożenie



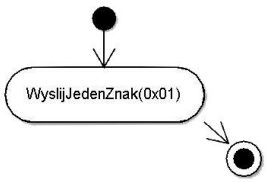
Rysunek 24: Funkcja EEPROMCzytaj



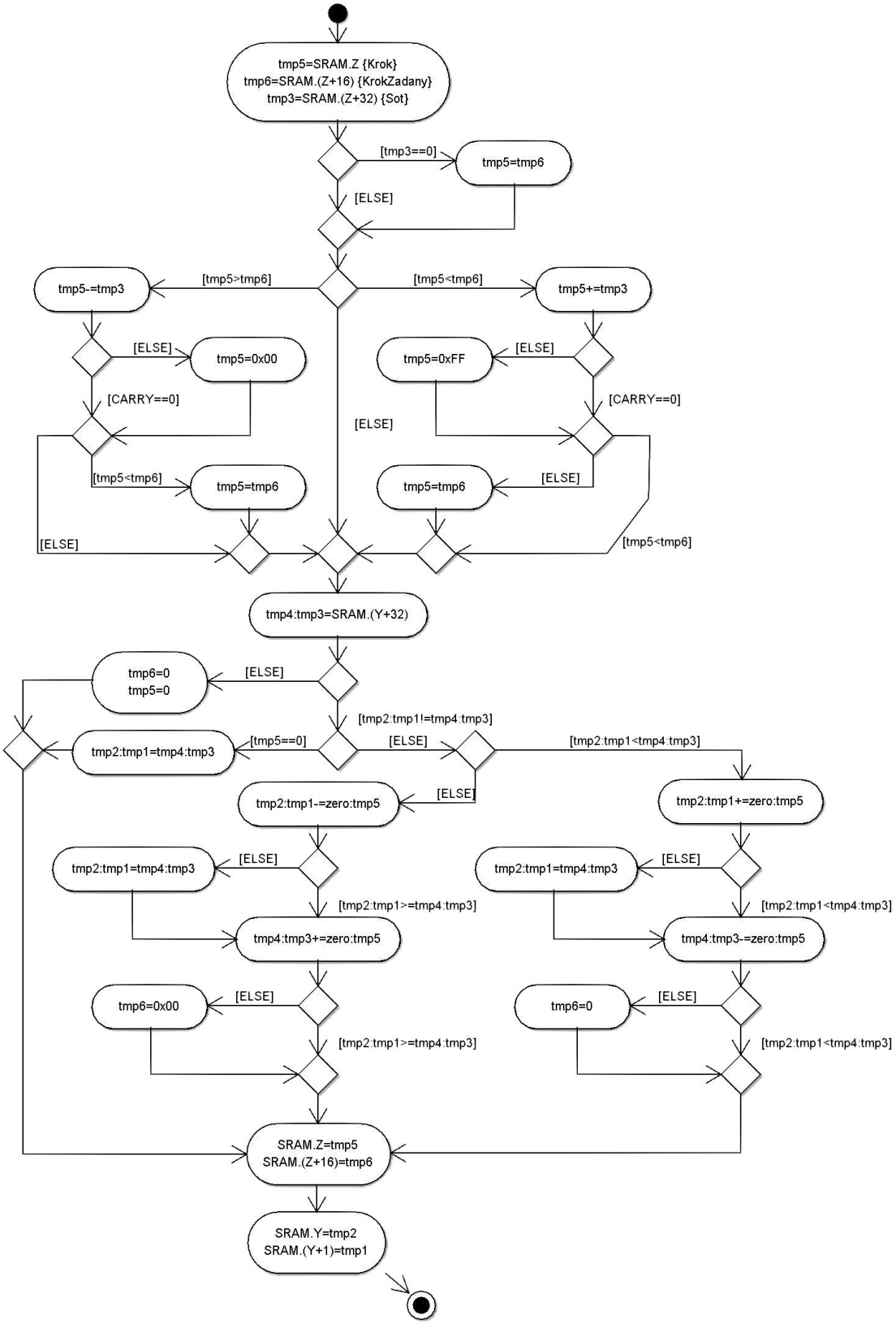
Rysunek 25: Funkcja EEPROMZapisz



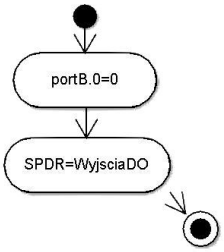
Rysunek 26: Funkcja OdbierzDane



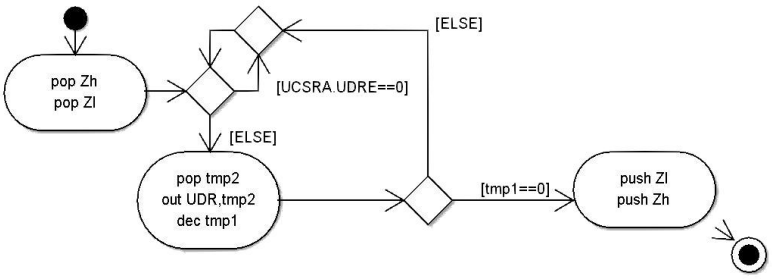
Rysunek 27: Funkcja Potwierdz



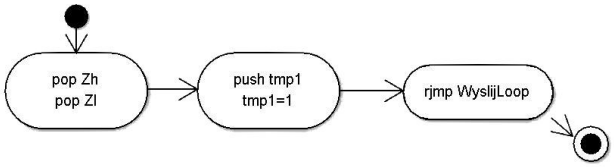
Rysunek 28: Funkcja TIMSerwaRef



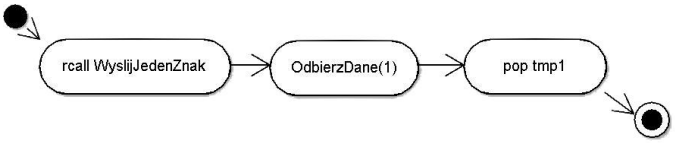
Rysunek 29: Funkcja SendDO



Rysunek 30: Funkcja Wyslij

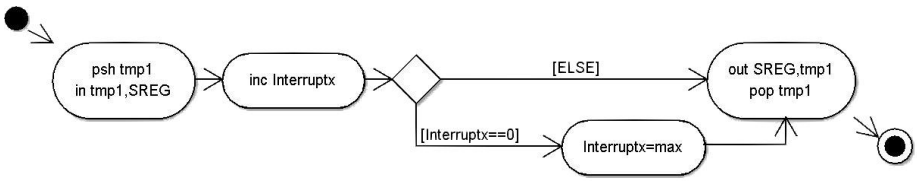


Rysunek 31: Funkcja WyslijJedenZnak

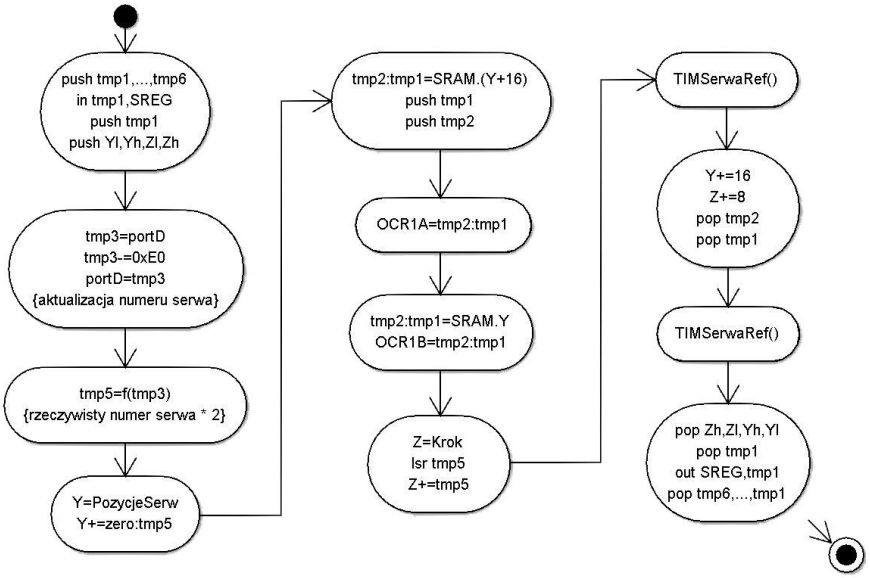


Rysunek 32: Funkcja WyslijSprawdz

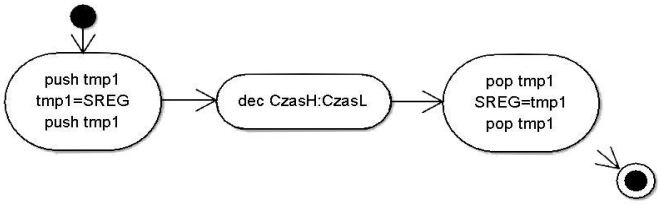
Funkcje przerwań:



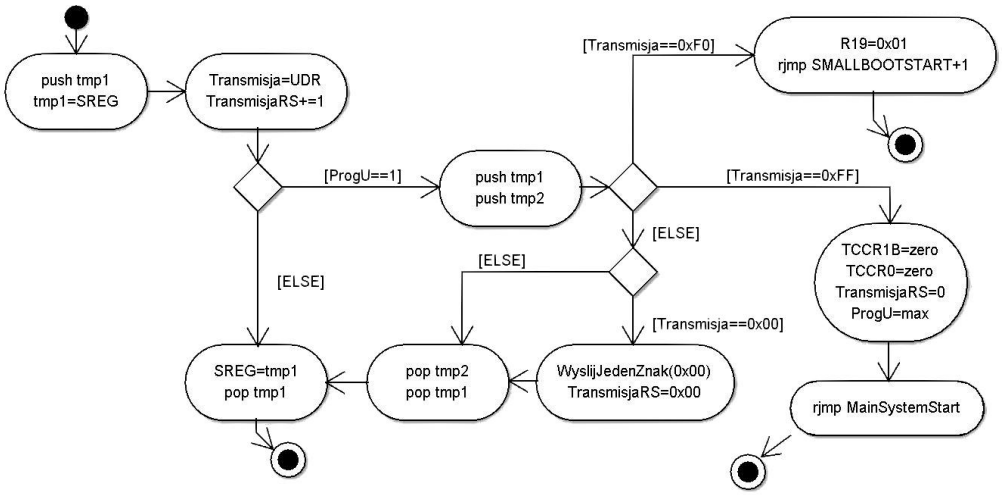
Rysunek 33: Funkcje EXT_INT0 i EXT_INT1



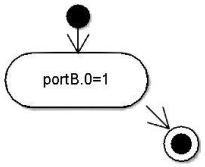
Rysunek 34: Funkcja TIM1_CAPT



Rysunek 35: Funkcja TIM0_OVF

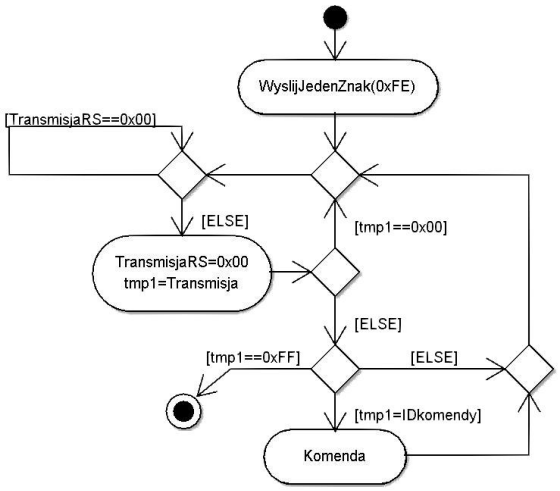


Rysunek 36: Funkcja USART_RXC

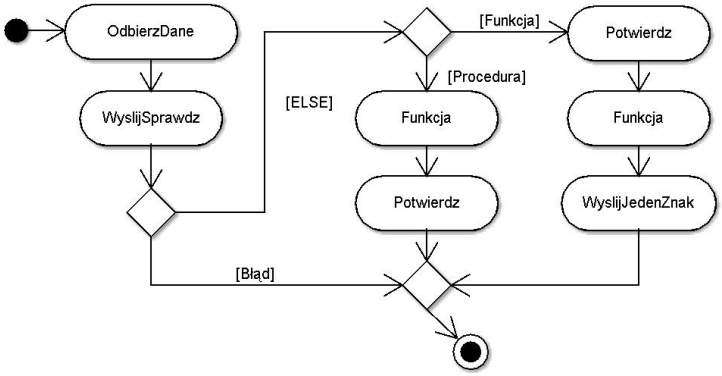


Rysunek 37: Funkcja SPI_STC

Funkcje realizujące obsługę komend



Rysunek 38: Funkcja Komendy



Rysunek 39: Szablon funkcji poszczególnych komend

8 Aplikacja MAS-SC16A

8.1 Ogólny opis aplikacji

Rozdiał ten to dokumentacja projektu aplikacji MAS-SC16A realizującej pełną obsługę urządzenia MAS-SC16A.
Aplikacja MAS-SC16A to program realizujący:

- Zapisywanie i kasowanie pamięci przeznaczonej dla Systemu oraz pamięci przeznaczonej dla Programu mikrokontrolera.
- Przełączanie urządzenia w różne tryby pracy (dostępność trybów zależna od aktualnego trybu pracy).
- Edycja parametrów i ustawień urządzenia.
- Realizacja komend poprzez złącze RS232.
- Pisanie programów obsługujących urządzenie.
- Uruchamianie, Kompilacja oraz Kasowanie i Zapis programu użytkownika do pamięci programu mikrokotrolera.

Język programowania przyjęty dla realizowanej aplikacji jest zgodny z semantyką języka C. Każda linia programu kończy się znakiem ';'. Grupa komend to polecenia znajdujące się pomiędzy znakami " i ". Instrukcja przypisania danej wartości do danej zniemmej to znak '=', natomiast instrukcje warunkowe to '<', '<=', '==', '>=', '>'.

9 Wymagania

Rozdział zawiera opis wymagań stawiany realizowanemu ServoController'owi.

Wersja	Autor	Firma	Data	Podpis
Wersja 1	Dr inż. Maciej Sławiński	Politechnika Warszawska	25-11-2005	

9.1 Wstęp

Rozdział wymagania opisuje wytyczne i ograniczenia stawiane ServoController'owi i programowi przeznaczonemu do nadzorowania jego pracy.

9.1.1 Definicje akronimy i skróty

Termin	Objaśnienie	Synonimy (nie zalecane)
Parametry urządzenia	Do parametrów urządzenia wliczamy zmienne definiowane programowo czyli: opcjonalne uruchamianie programu na starcie oraz rozpoczynanie pracy od trybu Komendy.	Parametry układu
Parametry programu	Parametry do których zaliczamy: położenie pliku Komend, polecenia odnoszące się do komend serwomechanizmów oraz czas i konfigurację ustawień kontroli serwokontrolera.	Parametry aplikacji
Parametry komunikacji	Parametry definiujące ustawienia złącza RS232 czyli: prędkość i numer portu COM.	Parametry RSa
Parametry kompilacji	Parametry dotyczące kompilacji programu użytkownika. Do tych parametrów zaliczamy położenie plików pamięci i konfiguracji oraz kompilatora.	Kompilator
BootLoader	Program realizujący zapis odczyt i kasowanie pamięci procesora układu.	System
Program	Program napisany, skompilowany czy też uruchaminy praz użytkownika.	
System	Program w którym została zrealizowana pełna obsługa funkcji i zarządzający jego pracą.	System operacyjny
Tryb pracy	Jeden z trybów pracy urządzenia: BootLoader, System, Komendy, Program.	

9.1.2 Referencje, odsyłacze do innych dokumentów

Układ jest odpowiednikiem serwokontrolera zaprezentowanego na stronie:
"'http://www.lynxmotion.com"'
Zbudowany jest on na mikrokontrolerze firmy ATMEL ATmega8. Dostępna pełna dokumentacja na stronie:
"'http://www.atmel.com"'

9.2 Ogólny opis

Aplikacja jest to standardowy edytor tekstu wyposażony w funkcje wspomagające edycję tekstu. Należy tu wymienić zarówno obsługę plików tekstowych, procesu drukowania oraz samo wspomaganie edycji tekstu. Do standardowo wliczonych funkcji zaliczamy również zarządzanie widokami okna dialogowego - paski narzędziowe i pasek stanu.

9.3 Specyficzne wymagania

9.3.1 Wymagania funkcjonalne

Opis funkcji programu przeznaczonych do obsługi serwokontrolera:

Nazwa	OnEdycjaSekwencji
Opis	Generowanie sekwencji komend dla wyjść PWM $(S1..S16)$.
Dane wejściowe	Numery serwomechanizmów i ich pozycje.
Źródło danych wejściowych	Użytkownik.
Wynik	Uruchomienie okna dialogowego realizującego generowanie sekwencji komend.
Warunek wstępny	Układ jest w Trybie pracy "Komendy".
Warunek końcowy	Jak wyżej.
Efekty uboczne	Zmiana kodu programu.
Powód	Możliwość definiowania komend poprzez obsługę on-line.

Nazwa	OnEdycjaPozycji
Opis	Generowanie komend realizujących pozycję/poziom sygnałów sterujących wyjściami PWM (S1..S16).
Dane wejściowe	Numery serwomechanizmów i ich pozycje.
Źródło danych wejściowych	Użytkownik.
Wynik	Uruchomienie okna dialogowego realizującego generowanie komend.
Warunek wstępny	Układ jest w Trybie pracy "Komendy".
Warunek końcowy	Jak wyżej.
Efekty uboczne	Zmiana kodu programu.
Powód	Możliwość definiowania komend poprzez obsługę on-line.

Nazwa	OnOpcjeParametryUrzadzenia
Opis	Edycja Parametrów Urządzenia.
Dane wejściowe	Parametry urządzenia.
Źródło danych wejściowych	Użytkownik.
Wynik	Zmiana parametrów urządzenia.
Warunek wstępny	Urządzenie jest dostępne.
Warunek końcowy	Jak wyżej.
Efekty uboczne	Zmiana parametrów urządzenia.
Powód	Edycja parametrów urządzenia.

Nazwa	OnOpcje
Opis	Edycja Parametrów portu RS232.
Dane wejściowe	Parametry komunikacji.
Źródło danych wejściowych	Użytkownik.
Wynik	Zmiana parametrów komunikacji.
Warunek wstępny	BRAK.
Warunek końcowy	BRAK.
Efekty uboczne	Zmiana parametrów komunikacji.
Powód	Edycja ustawień komunikacji.

Nazwa	OnOpcjeUstawienia
Opis	Edycja Parametrów Programu.
Dane wejściowe	Parametry Programu.
Źródło danych wejściowych	Użytkownik.
Wynik	Zapisanie nowych parametrów aplikacji.
Warunek wstępny	BRAK.
Warunek końcowy	BRAK.
Efekty uboczne	BRAK.
Powód	Edycja parametrów pracy programu.

Nazwa	OnOpcjeParametryKompilacji
Opis	Edycja Parametrów Kompilacji.
Dane wejściowe	Parametry kompilacji.
Źródło danych wejściowych	Użytkownik.
Wynik	Zmiana ustawień programu dotyczących procesu kompilacji programu użytkownika.
Warunek wstępny	BRAK.
Warunek końcowy	BRAK.
Efekty uboczne	BRAK.
Powód	Edycja parametrów kompilacji programu użytkownika.

Nazwa	OnOpcjeKomendy
Opis	Edycja Komend programu.
Dane wejściowe	Komendy programu.
Źródło danych wejściowych	Wykaz komend programu.
Wynik	Zdefiniowanie komend programu.
Warunek wstępny	BRAK.
Warunek końcowy	BRAK.
Efekty uboczne	Pojawienie się bądź usunięcie komendy.
Powód	Edycja komend programu.

Nazwa	SprawdzStan
Opis	Sprawdzenie aktualnego trybu pracy urządzenia.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Funkcja zwraca tryb pracy urządzenia.
Warunek wstępny	BRAK.
Warunek końcowy	BRAK.
Efekty uboczne	BRAK.
Powód	Możliwość identyfikowania i kontroli pracy urządzenia.

Nazwa	OnTrybBootLoader
Opis	Przełączanie urządzenia do trybu BootLoader.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zmiana trybu pracy urządzenie.
Warunek wstępny	BRAK.
Warunek końcowy	Urządzenie jest w trybie System lub BootLoader.
Efekty uboczne	Zmiana trybu pracy urządzenia.
Powód	Kontrola pracy urządzenia.

Nazwa	OnTrybSystem
Opis	Przełączanie urządzenia do trybu System.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zmiana trybu pracy urządzenie.
Warunek wstępny	BRAK.
Warunek końcowy	Urządzenie jest w trybie pracy.
Efekty uboczne	Zmiana trybu pracy urządzenia.
Powód	Kontrola pracy urządzenia.

Nazwa	OnTrybKomendy
Opis	Przełączanie urządzenia do trybu Komendy.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zmiana trybu pracy urządzenie.
Warunek wstępny	BRAK.
Warunek końcowy	Urządzenie jest w trybie System.
Efekty uboczne	Zmiana trybu pracy urządzenia.
Powód	Kontrola pracy urządzenia.

Nazwa	OnTrybProgram
Opis	Przełączanie urządzenia do trybu Program.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zmiana trybu pracy urządzenie.
Warunek wstępny	BRAK.
Warunek końcowy	Urządzenie jest w trybie System.
Efekty uboczne	Zmiana trybu pracy urządzenia.
Powód	Kontrola pracy urządzenia.

Nazwa	ResetRS232
Opis	Funkcja wyłącza a następnie włącza port RS232 odświeżając stan tego portu.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	BRAK.
Warunek wstępny	BRAK.
Warunek końcowy	BRAK.
Efekty uboczne	BRAK.
Powód	Resetowanie portu komunikacyjnego RS232.

Nazwa	Uruchom
Opis	Uruchomienie aktualnego programu.
Dane wejściowe	Treść programu.
Źródło danych wejściowych	Okno aplikacji.
Wynik	Przejście urządzenia do trybu komend i wykonywanie komend przekazywanych przez program.
Warunek wstępny	Niezerowa treść programu.
Warunek końcowy	Urządzenie znajduje się w trybie komend.
Efekty uboczne	Zablokowanie edycji treści programu.
Powód	Usuwanie usterek i błędów.

Nazwa	Restart
Opis	Ponowne uruchomienie debagowanego programu.
Dane wejściowe	Treść programu.
Źródło danych wejściowych	Okno aplikacji.
Wynik	Zresetowanie ustawień i przejście do pierwszej linii programu.
Warunek wstępny	Niezerowa treść programu.
Warunek końcowy	Urządzenie znajduje się w trybie komend.
Efekty uboczne	BRAK.
Powód	Restart programu.

Nazwa	WykonajKrok
Opis	Wykonanie kolejnej komendy programu.
Dane wejściowe	Treść programu.
Źródło danych wejściowych	Okno aplikacji.
Wynik	Wykonanie kolejnej komendy programu.
Warunek wstępny	Niezerowa treść programu.
Warunek końcowy	Urządzenie znajduje się w trybie komend.
Efekty uboczne	BRAK.
Powód	Debugowanie programu.

Nazwa	Pauza
Opis	Zatrzymanie działającego programu.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zatrzymanie wykonywania programu na danej linii.
Warunek wstępny	Program jest wykonywany.
Warunek końcowy	Urządzenie znajduje się w trybie komend.
Efekty uboczne	BRAK.
Powód	Pauza programu.

Nazwa	Stop
Opis	Zatrzymanie debagowanego programu.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zatrzymanie aktualnie wykonywanego programu.
Warunek wstępny	BRAK.
Warunek końcowy	BRAK.
Efekty uboczne	BRAK.
Powód	Zatrzymanie programu.

Nazwa	Kompiluj
Opis	Skompilowanie i ewentualne załadowanie kodu programu do pamięci mikrokontrolera.
Dane wejściowe	Treść programu.
Źródło danych wejściowych	Okno aplikacji.
Wynik	Kod wykonywalny i załadowanie programu do pamięci układu.
Warunek wstępny	Urządzenie pracuje w trybie System.
Warunek końcowy	Kompilacja programu przebiegła bezbłędnie.
Efekty uboczne	Nadpisanie pamięci programu mikrokontrolera.
Powód	Możliwość autonomicznego uruchamiania programu.

Nazwa	ProgramZapisz
Opis	Funkcja zapisuje program użytkownika do pamięć programu mikrokontrolera.
Dane wejściowe	Program użytkownika - plik HEX.
Źródło danych wejściowych	Kompilator lub okno dialogowe.
Wynik	Zmiana zawartości pamięci programu układu.
Warunek wstępny	Niezerowy plik wejściowy.
Warunek końcowy	Układ w trybie BootLoader.
Efekty uboczne	Zapisanie pamięci programu mikrokontrolera.
Powód	Zarządzanie pamięcią układu.

Nazwa	ProgramPobierz
Opis	Funkcja odczytuje program użytkownika z pamięci programu mikrokontrolera.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zapisanie pamięci programu układu do pliku.
Warunek wstępny	BRAK.
Warunek końcowy	Układ w trybie BootLoader.
Efekty uboczne	Odczyt pamięci programu mikrokontrolera.
Powód	Zarządzanie pamięcią układu.

Nazwa	ProgramKasuj
Opis	Funkcja kasuje program użytkownika w pamięci programu mikrokontrolera.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Wyczyszczenie zawartości pamięci programu układu.
Warunek wstępny	BRAK.
Warunek końcowy	Układ w trybie BootLoader.
Efekty uboczne	Wyczyszczenie pamięci programu mikrokontrolera.
Powód	Zarządzanie pamięcią układu.

Nazwa	SystemWersja
Opis	Odczyt wersji Systemu.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zwrócenie wartości identyfikującej wersję Systemu.
Warunek wstępny	BRAK.
Warunek końcowy	Układ jest w trybie BootLoadera.
Efekty uboczne	BRAK.
Powód	Identyfikacja Systemu.

Nazwa	SystemZapisz
Opis	Funkcja zapisuje system do pamięć programu mikrokontrolera.
Dane wejściowe	System - plik HEX.
Źródło danych wejściowych	Kompilator lub okno dialogowe.
Wynik	Zmiana zawartości pamięci programu układu.
Warunek wstępny	Niezerowy plik wejściowy.
Warunek końcowy	Układ w trybie BootLoader.
Efekty uboczne	Zapisanie pamięci programu mikrokontrolera.
Powód	Zarządzanie pamięcią układu.

Nazwa	SystemPobierz
Opis	Funkcja odczytuje system z pamięci programu mikrokontrolera.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zapisanie pamięci programu układu do pliku.
Warunek wstępny	BRAK.
Warunek końcowy	Układ w trybie BootLoader.
Efekty uboczne	Odczyt pamięci programu mikrokontrolera.
Powód	Zarządzanie pamięcią układu.

Nazwa	SystemKasuj
Opis	Funkcja kasuje system w pamięci programu mikrokontrolera.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Wyczyszczenie zawartości pamięci programu układu.
Warunek wstępny	BRAK.
Warunek końcowy	Układ w trybie BootLoader.
Efekty uboczne	Wyczyszczenie pamięci programu mikrokontrolera.
Powód	Zarządzanie pamięcią układu.

9.3.2 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne zostały opracowane na podstawie projektu serwokontrolera.

Lp	Data	Opis	Motywacja	Uwagi
1	07-07-10	Sygnał sterujący PWM taktowany z częstotliwością $50Hz$.	Parametr zgodny z wytycznymi producenta.	BRAK
2	07-07-10	Czas trwania impulsu PWM w stanie wysokim wynosi od $0,5ms$ do $2,5ms$.	Parametr zgodny z wytycznymi producenta.	BRAK
3	07-07-10	Dozwolone prędkości komunikacji RS232 to 4800, 9600, 14,4k, 19,2k, 28,8k, 38,4k, 57,6k, 115,2k oraz 230,4k.	Wartości optymalne dla mikrokontrolera i RSa w komputerze PC.	BRAK
4	07-07-10	Wszystkie dostępne funkcje zaimplementowane są w Systemie mikrokontrolera.	Wytyczne dotyczące budowy urządzenia.	BRAK

Lp	Data	Opis	Motywacja	Uwagi
5	07-07-10	Ustawienie portu RS232 we wszystkich trybach pracy jest takie samo.	Uniwersalizm ustawień.	BRAK
6	07-07-10	Definiowanie parametrów pracy układu z poziomu aplikacji.	Łatwa obsługa ustawień konfigurujących serwokontroler.	Ustawienia uruchomieniowe Systemu są widoczne po restarcie układu.

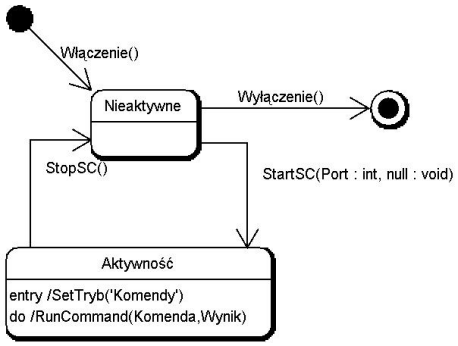
10 Analiza

10.1 Funkcje i czynności

11 Pakiet funkcji programu MATLAB

11.1 Praca urządzenia z poziomu aplikacji MATLAB

Rysunek 40 przedstawia stany pracy serwokontrolera w odniesieniu do zastosowań w programie MATLAB.



Rysunek 40: Praca układu z poziomu programu MATLAB

Ważnym czynnikiem jest to, iż każda rozpoczęta sesja (Aktywność) urządzenia musi być zakończona komendą StopSC. Komenda to realizuje wyłączenie dostępności serwokontrolera z poziomu programu MATLAB.

11.2 Opis pakietu funkcji

Pakiet funkcji programu MATLAB to pliki (M-file) zawierające implementację wszystkich funkcji systemowych zdefiniowanych i dostępnych w aplikacji MAS-SC16A. Funkcje dostępne w M plikach:

1. Komendy wejść analogowych:
    - AIxRead(numer) - odczyt wartości na wejściu analogowym AInumer.
2. Komendy wejść analogowo-cyfrowych:
    - ADIAxRead(numer) - odczyt analogowej wartości na wejściu ADInumer.
    - ADIDxGet(numer) - odczyt stanu na wejściu analogowo-cyfrowym ADInumer.
    - ADIDGet - odczyt stanu na wejściach ADI.
3. Komendy wyjść cyfrowych
    - DOxSet(nummer,stan) - ustawienie stanu wyjścia cyfrowego DOnumer=stan.
    - DOxGet(numer) - odczyt stanu wyjścia cyfrowego DOnumer.
    - DOSet(wartość) - ustawienie wartości/stanów wyjść cyfrowych DO=wartość.
    - DOGet - odczyt wartości stanów cyfrowych DO.
4. Przerwania
    - INT0Get - odczyt przerwania INT0.
    - INT0Clr - wyczyszczenie przerwań INT0.

- INT1Get - odczyt przerwania INT1.
- INT1Clr - wyczyszczenie przerwań INT1.

5. Komendy serwomechanizmów

   - ServoSet(numer,krok,czas,soft,pozycja) - zadanie pozycji serwa numer z zadanym krokiem lub gdy krok==0 w podanym czasie, parametr soft to łagodny start i stop - łagodny przyrost kroku (krok = krok +/- soft).
   - ServaSet(Krok, Czas, Soft, Pozycja1, Pozycja2, Pozycja3, Pozycja4, Pozycja5, Pozycja6, Pozycja7, Pozycja8) - zadanie pozycji serw od numeru 1 do 8 z zadanym krokiem lub gdy krok==0 w podanym czasie, parametr soft to łagodny start i stop - łagodny przyrost kroku (krok = krok +/- soft).
   - ServoRead(numer) - odczyt aktualnej pozycji serwa numer.

6. Komendy obsługujące diodę

   - LED(stan) - włączenie (stan=1) lub wyłączenie (stan=0) diody.
   - LEDFlash(czas) - zapalenie diody na czas [ms].

Oprócz tych funkcji dostępne są jeszcze:

1. Komendy zarządzające pracą urządzenia:

   - StartSC(Port,Prędkość) - otwarcie portu RS i uruchomienie urządzenia
   - StopSC - zatrzymanie urządzenia i zamknięcie portu RS
   - RunCommand(Komenda,Wynik) – wykonanie Komendy i zwrot wartości o podanym rozmiarze
   - GetTryb – zwraca tekst odpowiadający aktualnemu trybowi pracy układu
   - SetTryb(Tryb) – przełącza urządzenie w podany tryb pracy
   - CF - Zamknięcie palcy
   - OF - Otwarcie palcy
   - SF(Operacja) - Włączenie/wyłączenie palcy

Korzystając z polecenia RunCommand można zaimplementować dowolną funkcję urządzenia.

11.3 Funkcje zarządzające pracą urządzenia

Ze względu na uniwersalizm dotyczący realizacji funkcji systemowych urządzenia w dalszej części szczegółowo zostaną omówione tylko i wyłącznie funkcje zarządzające pracą układu.

11.3.1 Wymagania

Nazwa	StartSC
Opis	Włączenie urządzenia.
Dane wejściowe	Port, Prędkość.
Źródło danych wejściowych	Sprzęt.
Wynik	Otwarcie portu RS o podanym numerze z podaną prędkością i włączenie urządzenia.
Warunek wstępny	BRAK
Warunek końcowy	Urządzenie jest dostępne poprzez dostępny port RS.
Efekty uboczne	Otwarcie portu RS.
Powód	Uruchomienie i rozpoczęcie pracy urządzenia.

Nazwa	StopSC
Opis	Wyłączenie urządzenia.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Wyłączenie urządzenia i zamknięcie portu RS.
Warunek wstępny	BRAK
Warunek końcowy	BRAK.
Efekty uboczne	BRAK.
Powód	Zakończenie pracy urządzenia.

Nazwa	GetTryb
Opis	Pobranie informacji o aktualnym trybie pracy urządzenia.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Odpowiedź reprezentująca aktualny tryb pracy układu.
Warunek wstępny	BRAK
Warunek końcowy	BRAK.
Efekty uboczne	BRAK.
Powód	Otrzymanie informacji o aktualnym trybie pracy urządzenia.

Nazwa	SetTryb
Opis	Ustawienie trybu pracy urządzenia.
Dane wejściowe	Nowy tryb pracy.
Źródło danych wejściowych	Użytkownik.
Wynik	Ustawienie trybu pracy na zadany.
Warunek wstępny	Tryb odpowiada rzeczywistym wartością dostępnym w urządzeniu
Warunek końcowy	BRAK.
Efekty uboczne	Możliwość przełączenia w niepożądany tryb pracy.
Powód	Zmiana trybu pracy urządzenia.

Nazwa	RunCommand
Opis	Wykonanie podanej komendy.
Dane wejściowe	Komenda, Wynik.
Źródło danych wejściowych	Użytkownik.
Wynik	Realizacja podanej komendy i zwrot wartości odpowiadającej poprawności wykonania lub wartości zwracanej przez daną funkcję o rozmiarze Wynik.
Warunek wstępny	Komenda nie może być pustym ciągiem znaków a wynik nie może być ujemny
Warunek końcowy	BRAK.
Efekty uboczne	BRAK.
Powód	Realizacja komendy zgodnie z przyjętym protokołem komunikacji (rozdział 7.2.1) urządzenia.

Nazwa	CF
Opis	Zamknięcie palcy manipulatora.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zamknięcie palcy manipulatora.
Warunek wstępny	Nie dotyczy.
Warunek końcowy	Palce chwytaka manipulatora są włączone.
Efekty uboczne	Przełączenie serwomechanizmów 1 i 2 na ciągłą pracę.
Powód	Obsługa palcy manipulatora.

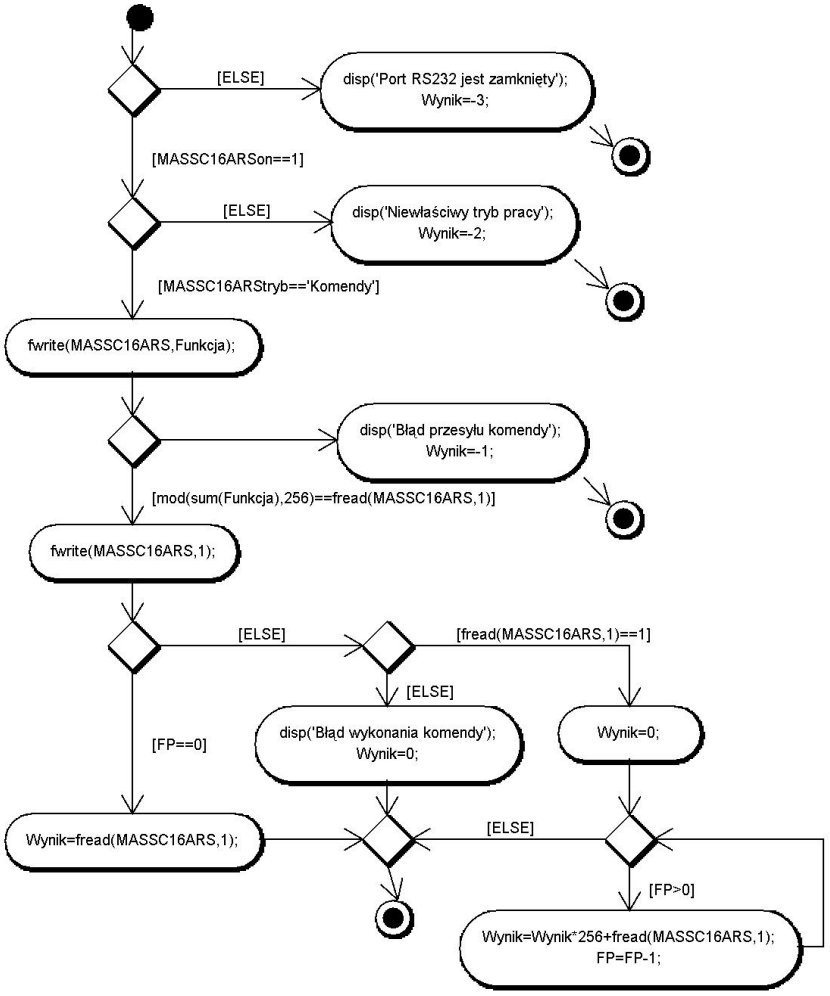
Nazwa	OF
Opis	Otwarcie palcy manipulatora.
Dane wejściowe	BRAK.
Źródło danych wejściowych	Nie dotyczy.
Wynik	Zamknięcie palcy chwytaka manipulatora.
Warunek wstępny	Nie dotyczy.
Warunek końcowy	Palce chwytaka manipulatora są włączone.
Efekty uboczne	Przełączenie serwomechanizmów na ciągłą pracę przy otwrciu cgwytaka.
Powód	Obsługa palcy manipulatora.

Nazwa	SF
Opis	Włączenie lub wyłączenie palcy manipulatora.
Dane wejściowe	Operacja.
Źródło danych wejściowych	Użytkownik.
Wynik	Uruchomienie lub wyłączenie serwomechanizmów 1 i 2.
Warunek końcowy	BRAK.
Efekty uboczne	Przełączenie trybu pracy serwomechanizmów 1 i 2.
Powód	Obsługa palcy manipulatora.

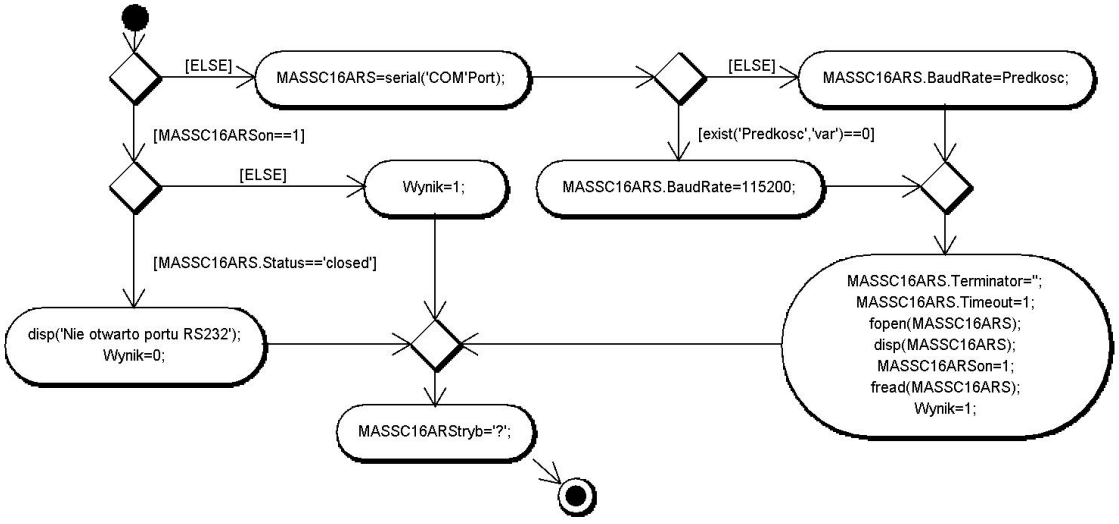
Realizacja pozostałych komend to wykorzystywanie i realizacja funkcji zarządzających pracą urządzenia z odpowiednimi parametrami.

11.3.2 Szczegółowy opis funkcji

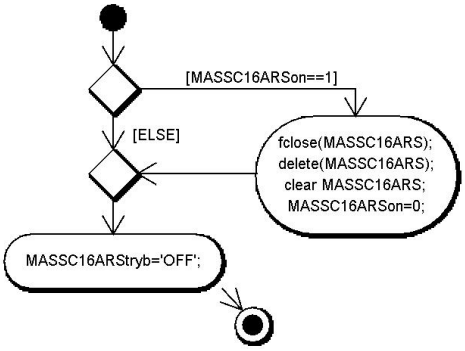
Na rysunkach 41-45 przedstawiono diagramy czynności opisujące realizację poszczególnych funkcji zarządzających, zaprezentowanych w rozdziale 11.3.1.



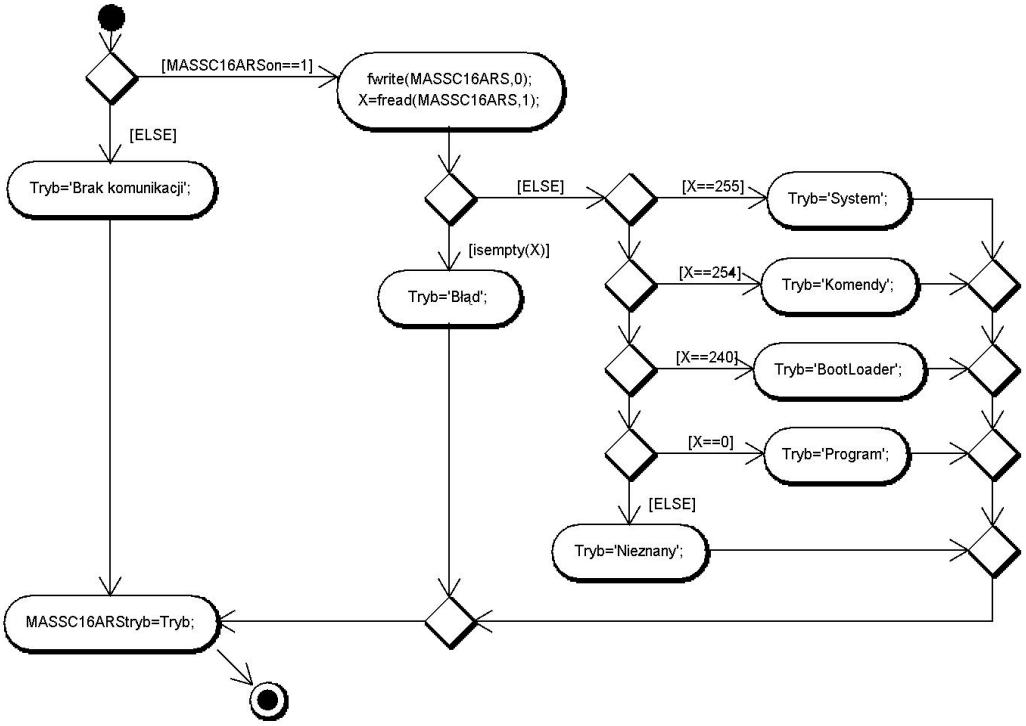
Rysunek 41: Funkcja RunCommand(Komenda,Wynik)



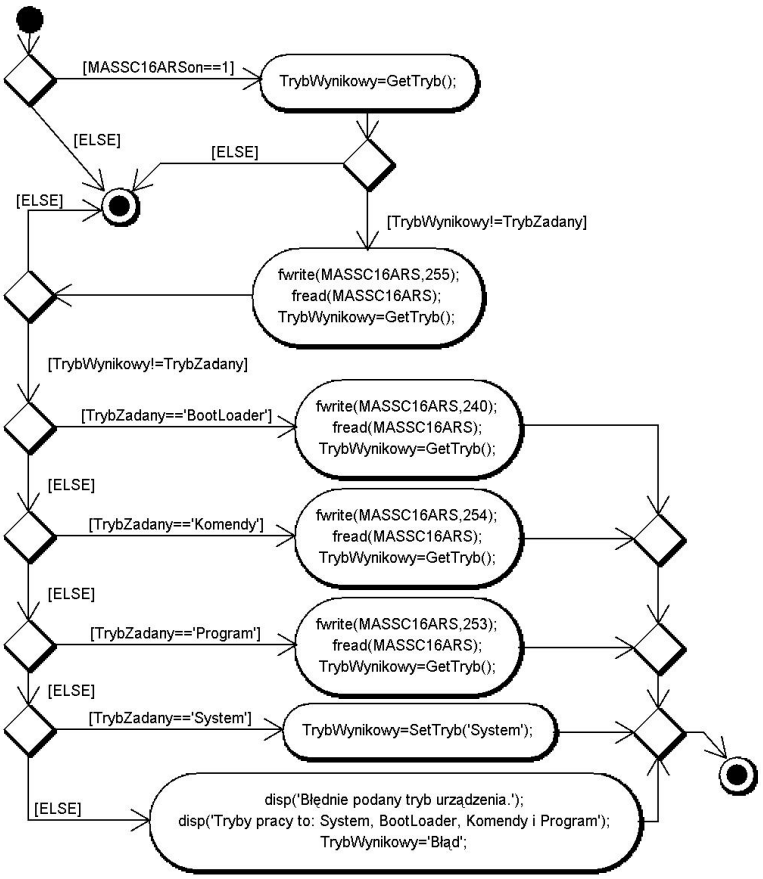
Rysunek 42: Funkcja StartSC(Port,Prędkość)



Rysunek 43: Funkcja StopSC()



Rysunek 44: Funkcja GetTryb()



Rysunek 45: Funkcja SetTryb(Tryb)