BOOTLOADER 3.4

1. Cechy

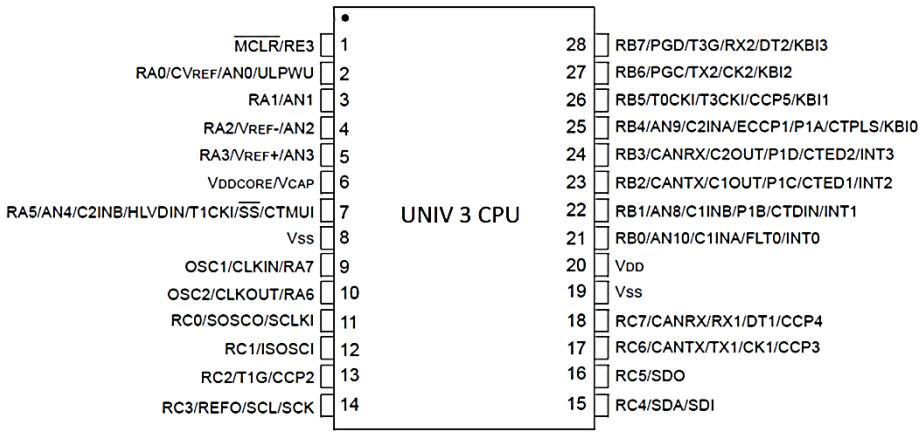
- Bootloader procesora uniwersalnego UNIV 3 CPU.
- Wersja bootloader-a: 3.4
- Trzy tryby pracy bootloader-a:
  - komunikacja przez magistralę HAPCAN
  - komunikacja przez magistralę HAPCAN i/lub port szeregowy UART
  - tryb programowania przez magistralę HAPCAN i/lub port szeregowy UART
- 7 wiadomości UART
- 14 wiadomości CAN
- Możliwość programowania pamięci FLASH i EEPROM przez porty UART lub CAN bez konieczności używania sprzętowych programatorów
- Możliwość tworzenia własnych programów funkcyjnych

2. Opis

Bootloader to program, który jest wykonywany zaraz po włączeniu zasilania procesora. Główne zadanie tego programu to umożliwienie komunikacji z procesorem np. z komputera PC, bez użycia specjalnych programatorów sprzętowych. Komunikacja z procesorem może być nawiązana poprzez port szeregowy UART (RS232) lub magistralę CAN.

Bootloader umożliwia też wgranie do procesora i konfigurowanie oprogramowania funkcyjnego, które powoduje, że procesor działa jako konkretne urządzenie typu przycisk, ściemniacz itp. Dzięki bootloader-owi, komunikacja z procesorem jest możliwa nawet, jeśli nie jest wgrane do niego żadne oprogramowanie funkcyjne firmware lub jest ono nieprawidłowe.

Uzupełnieniem tego dokumentu jest oryginalna dokumentacją procesora PIC18F26K80 firmy Microchip dostępna na stronie microchip.com.



Rysunek 1. Układ wyprowadzeń procesora UNIV 3 CPU

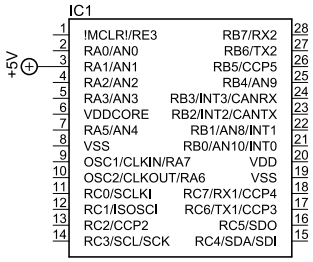
3. Spis treści

4. Tryby pracy bootloader-a

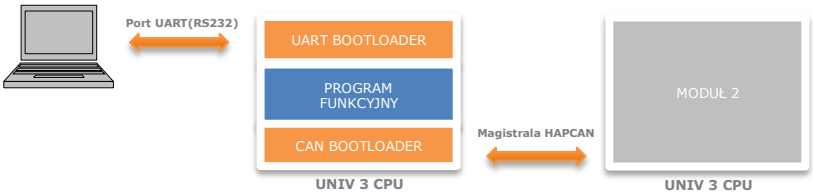
Bootloader może pracować w jednym z trzech trybów pracy:
1. Tryb normalnej pracy CAN i UART (32MHz)
2. Tryb normalnej pracy CAN (8MHz)
3. Tryb programowania

4.1. Tryb CAN i UART (32 MHz)

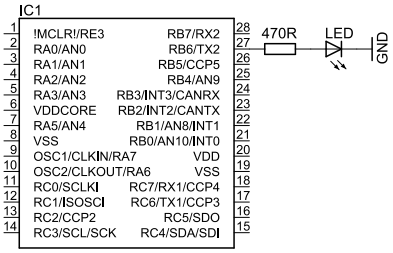
Ten tryb umożliwia komunikację z procesorem poprzez magistralę HAPCAN lub port szeregowy UART (*universal asynchronous receiver/transmitter*) procesora. Procesor w tym trybie pracy wykorzystywany jest głównie do budowy interfejsów pomiędzy komputerem PC a magistralą HAPCAN.
Tryb ten jest włączany sprzętowo poprzez podłączenie pinu 3 procesora UNIV 3 CPU z dodatnim biegunem zasilania (+5V) i restarcie procesora (Rysunek 2). Oprócz aktywacji CAN i UART bootloadera, zostaje zwiększona czterokrotnie częstotliwość pracy rezonatora i procesor taktowany jest zegarem 32 MHz.
Po restarcie procesora, tryb pracy bootloader-a sygnalizowany jest na wyprowadzeniu PIN27 UNIV 3 CPU (Rysunek 5).



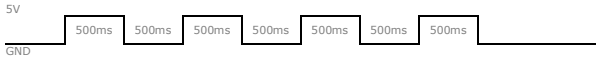
Rysunek 2. Podłączenie procesora UNIV 3 CPU do pracy w trybie CAN i UART



Rysunek 3. Procesor UNIV 3 w trybie CAN i UART bootloader-a



Rysunek 4. Pin sygnalizujący trybu pracy CAN i UART bootloader-a

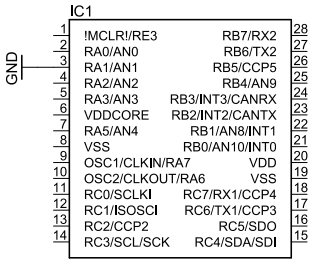


Bootloader w trybie CAN i UART (32 MHz) – możliwa jest komunikacja z procesorem poprzez magistralę HAPCAN lub od strony portu szeregowego UART (RS232). Częstotliwość zegara taktującego procesor to 32 MHz.

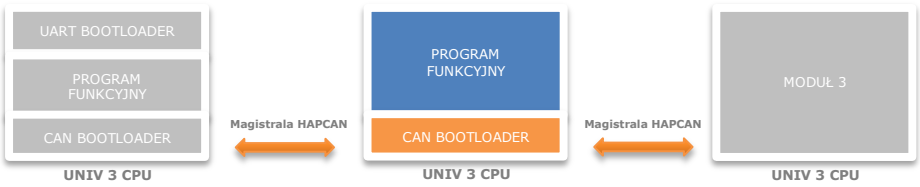
Rysunek 5. Przebieg sygnalizacji trybu pracy CAN i UART bootloader-a

4.2. Tryb CAN (8 MHz)

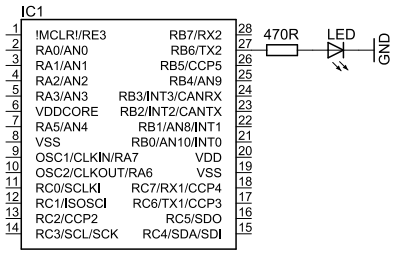
Tryb umożliwia komunikację z procesorem tylko poprzez magistralę HAPCAN. Aby komunikować się z takim procesorem wymagany jest moduł interfejsu komputera PC, który będzie przekazywał komunikaty PC na magistralę HAPCAN.
Tryb ten jest włączany sprzętowo poprzez podłączenie pinu 3 procesora UNIV 3 CPU z ujemnym biegunem zasilania (GND) i restarcie procesora (Rysunek 6).
Po restarcie procesora, tryb pracy bootloader-a sygnalizowany jest na wyprowadzeniu PIN27 UNIV 3 CPU (Rysunek 9).



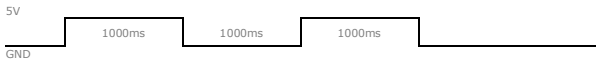
Rysunek 6. Podłączenie procesora UNIV 3 CPU do pracy w trybie CAN



Rysunek 7. Procesor UNIV 3 w trybie CAN bootloader-a



Rysunek 8. Pin sygnalizujący trybu pracy CAN i UART bootloader-a

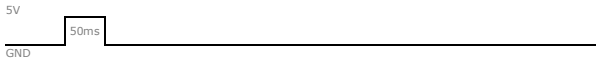


Bootloader w trybie CAN (8 MHz) – możliwa jest komunikacja z procesorem tylko poprzez magistralę HAPCAN. Częstotliwość zegara taktującego procesor to 8 MHz.

Rysunek 9. Sygnalizacja trybu pracy CAN bootloader-a

4.3. Tryb programowania

W tym trybie istnieje możliwość zmiany pamięci programu i danych procesora. Wykorzystywany jest on do wgrywania firmware i konfiguracji modułu.
Jest to tryb przełączany programowo poprzez wysłanie wiadomości do procesora. Jeśli procesor sprzętowo (przez zwarcie pinu 3 do +5V) zdefiniowany jest do pracy z UART i CAN bootloader-em, wtedy tryb programowania umożliwi zmianę pamięci programu i danych zarówno poprzez komunikację przez złącze szeregowe UART (RS232) jak i magistralę HAPCAN. Zdefiniowanie sprzętowe (przez zwarcie pinu 3 do GND) do pracy w trybie CAN bootloader-a, umożliwi programowanie procesora tylko przez magistralę HAPCAN.
Po restarcie procesora, tryb programowania sygnalizowany jest na wyprowadzeniu PIN27 UNIV 3 CPU (Rysunek 10).

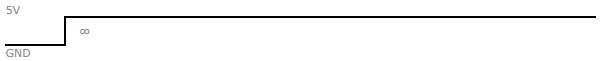


Bootloader w trybie programowania

Rysunek 10. Sygnalizacja trybu programowania

4.4. Tryb błędu

Procedura inicjująca działanie bootloader-a polega na samosprawdzeniu. Jeśli wynik tego testu jest pozytywny bootloader przechodzi do trybu pracy normalnej: CAN lub UART i CAN.
Tryb błędu sygnalizowany jest na wyprowadzeniu PIN27 UNIV 3 CPU nieprzerwanym stanem wysokim (Rysunek 11).



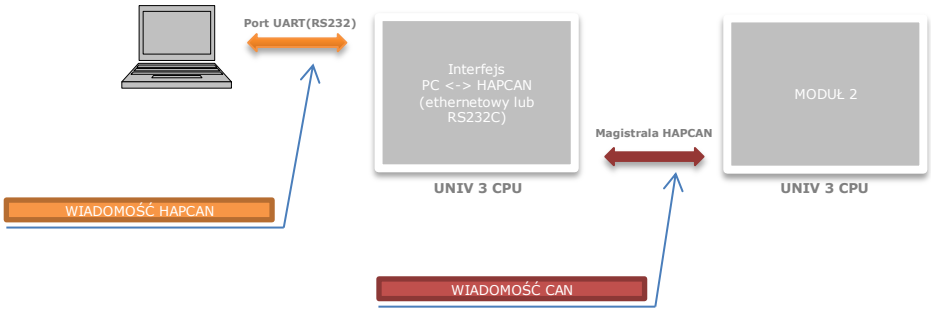
Bootloader w trybie błędu

Rysunek 11. Sygnalizacja trybu błędu

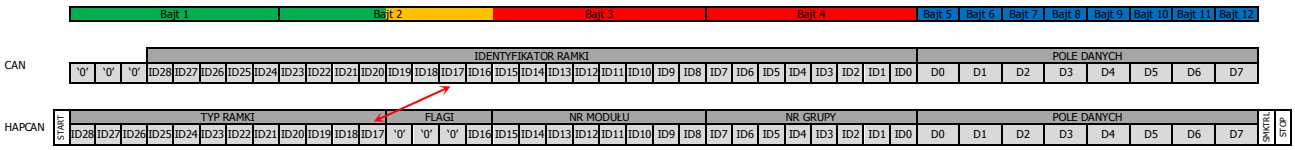
5. Komunikacja z bootloader-em

5.1. Budowa wiadomości HAPCAN

Wiadomości HAPCAN to wiadomości, które widziane są od strony komputera PC. Są one nieznacznie zmodyfikowane (z oryginalnych wiadomości CAN) w interfejsie HAPCAN<->PC. Ramka HAPCAN zbudowana jest z 12 bajtów. 4 pierwsze to identyfikator ramki CAN, a pozostałe 8 to bajty danych. Różnica pomiędzy ramką na magistrali CAN i ramką HAPCAN przesłaną do komputera polega na przesunięciu bitów IDENTYFIKATORA RAMKI (Tabela 1). Ponadto ramka przesyłana z magistrali CAN do komputera, jest powiększona w interfejsie HAPCAN<->PC o bajty startu stopu i sumy kontrolnej.



Rysunek 12. HAPCAN i CAN wiadomości



START - Bajt o wartość 0xAA sygnalizujący początek ramki
TYP RAMKI - Typ ramki danych definiujący znaczenie bajtów w POLU DANYCH
FLAGA ID16 - Flaga odpowiedzi. Jeśli ramka zostaje wysłana jako odpowiedź na otrzymane zapytanie to FLAGA ODP=1. W innym przypadku ODP=0. Pozostałe flagi są nieużywane i zawsze równe 0.
NR MODUŁU - Numer modułu nadającego
NR GRUPY - Numer grupy modułu nadającego
POLE DANYCH - 8 bajtów danych. Znaczenie tych bajtów zależne jest od TYPU RAMKI
SMKTRL - suma kontrolna ramki. Jest to suma wartości bajtów od 1 do 12
STOP - Bajt o wartość 0xA5 sygnalizujący koniec ramki

Tabela 1. Różnice pomiędzy ramką CAN i HAPCAN

5.2. Typy wiadomości systemu HAPCAN

W systemie HAPCAN występują dwa typy wiadomości:
- systemowe, które używane są do sterowania i programowania systemu z komputera;
- wiadomości normalne używane do komunikacji modułów między sobą.

Typ wiadomości identyfikowany jest po 12 pierwszych bitach ramki tzw. TYPIE RAMKI. Dotychczas używane typy wiadomości zebrano w tabeli poniżej. Bootloader obsługuje tylko wiadomości zacieniowane w poniższej tabeli. Pozostałe wiadomości mogą być używane w oprogramowaniu funkcyjnym.

WIADOMOŚCI SYSTEMOWE	
0x010 - pytanie do wszystkich procesorów o wyście z trybu programowania	Wiadomości obsługiwane przez **bootloader w trybie programowania**
0x020 - pytanie do procesora o wyście z trybu programowania	
0x030 - ramka adresująca	
0x040 - ramka danych	
0x100 - pytanie do procesora o wejście w tryb programowania	Wiadomości obsługiwane przez **bootloader w trybie normalnym**
0x101 - pytanie do grupy procesorów o restart	
0x102 - pytanie do procesora o restart	
0x103 - pytanie do grupy procesorów o hardware	
0x104 - pytanie do procesora hardware	
0x105 - pytanie do grupy procesorów o firmware	
0x106 - pytanie do procesora o firmware	
0x107 - pytanie do procesora o ustawienie domyślnych numerów modułu i grupy	
0x108 - pytanie do grupy procesorów o status	Wiadomości obsługiwane w trybie normalnym bootloader-a przez **oprogramowanie funkcyjne firmware**
0x109 - pytanie do procesora o status	
0x10A - wiadomość sterująca	
0x10B - pytanie do grupy procesorów o napięcie zasilania	Wiadomości obsługiwane przez **bootloader w trybie normalnym**
0x10C - pytanie do procesora o napięcie zasilania	
0x10D - pytanie do grupy procesorów o opis	
0x10E - pytanie do procesora o opis	
0x10F - pytanie do grupy procesorów o DEV ID	
0x111 - pytanie do procesora o DEV ID	
0x112 - pytanie do grupy procesorów o czas od startu	Wiadomości obsługiwane w trybie normalnym bootloader-a przez **oprogramowanie funkcyjne firmware**
0x113 - pytanie do procesora o czas od startu	
0x114 - pytanie do grupy procesorów o diagnostykę	
0x115 - pytanie do procesora o diagnostykę	

WIADOMOŚCI NORMALNE	
0x301 - wiadomość modułu przycisk	Wiadomości obsługiwane w trybie normalnym bootloader-a przez **oprogramowanie funkcyjne firmware**
0x302 - wiadomość przekaźnika	
0x303 - wiadomość odbiornika podczerwieni	
0x304 - wiadomość czujnika temperatury	
0x305 - wiadomość nadajnika podczerwieni	
0x306 - wiadomość ściemniacza	
0x307 - wiadomość sterownika rolet	
0x308 - wiadomość sterownika RGB	

Tabela 2. Lista wiadomości używanych w systemie HAPCAN

5.3. Wiadomości systemowe UART

Są to wiadomości używane do komunikacji z bootloader-em poprzez złącze szeregowe UART (RS232) procesora. Komunikacja z procesorem wymaga komputera PC z portem RS232C i układu transformującego poziomy napięć pomiędzy procesorem i PC – np. opartego na układzie scalonym MAX232.
Tylko część z wiadomości systemowych obsługiwana jest przez bootloader (Tabela 3). Pozostałe mogą być zaimplementowane w oprogramowaniu funkcyjnym firmware. Poniżej opisany jest sposób komunikacji z bootloader-em poprzez port szeregowy UART (RS232) procesora.

0x100 - pytanie do procesora o wejście w tryb programowania	Wiadomości obsługiwane przez bootloader w trybie normalnym UART (nie programowania)
0x102 - pytanie do procesora o restart	
0x104 - pytanie do procesora hardware	
0x106 - pytanie do procesora o firmware	
0x109 - pytanie do procesora o status	
0x10A - wiadomość sterująca	
0x10C - pytanie do procesora o napięcie zasilania	Wiadomości obsługiwane przez bootloader w trybie normalnym UART (nie programowania)
0x10E - pytanie do procesora o opis	
0x111 - pytanie do procesora o DEV ID	
0x113 - pytanie do procesora o czas od startu	
0x115 - pytanie do procesora o diagnostykę	

Tabela 3. Lista wiadomości systemowych używanych do komunikacji przez port szeregowy UART procesora

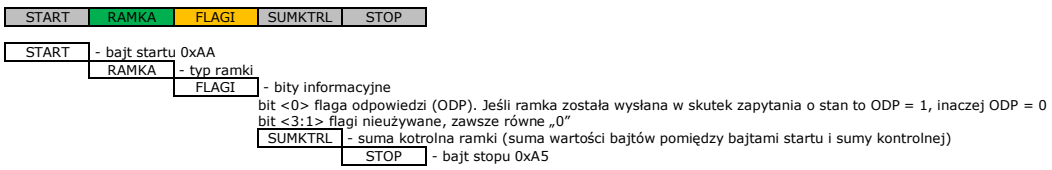


Tabela 4. Budowa ramki do komunikacji przez port UART

0x100 – *Pytanie do procesora o wejście w tryb programowania*

Jest to pytanie o wejście w tryb programowania, w którym możliwe jest dokonywanie zmian w pamięci danych i programu procesora. W odpowiedzi procesor wyśle ramkę potwierdzającą wykonanie polecenia. Wyjście z trybu programowania opisane jest w sekcji *5.5. Wiadomości w trybie programowania UART*.

Do portu UART ⇨	0xAA	0x100	0x0	0x10	0xA5								
Odpowiedź ⇦	0xAA	0x104	0x1	0xFF	0xFF	BVER1	BVER2	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

BVER1 | BREV2 – wersja bootloader-a w formacie BVER1.BREV2

0x102 – *Pytanie do procesora o restart*

Wiadomość spowoduje restart procesora. Procesor nie wyśle żadnej odpowiedzi.

Do portu UART ⇨	0xAA	0x102	0x0	0x30	0xA5
Odpowiedź ⇦	BRAK				

0x104 – *Pytanie do procesora o hardware*

Jest to pytanie o dane urządzenia. W odpowiedzi procesor wyśle typ urządzenia, wersję i numer seryjny.

Do portu UART ⇨	0xAA	0x104	0x0	0x50	0xA5								
Odpowiedź ⇦	0xAA	0x104	0x1	HARD1	HARD2	HVER	0xFF	ID0	ID1	ID2	ID3	SUMKTRL	0xA5

HARD1 | HARD2 – 0x3000 – procesor lub moduł uniwersalny UNIV
HVER – 0x03 – wersja procesora
IDx – numer seryjny

0x106 – *Pytanie do procesora o firmware*

Jest to pytanie o wersję oprogramowania funkcyjnego. W odpowiedzi procesor wyśle informacje o wgranym oprogramowaniu firmware. Jeśli brak oprogramowania lub jest ono nieprawidłowe wysłana zostanie ramka błędu.

Do portu UART ⇨	0xAA	0x106	0x0	0x70	0xA5								
Odpowiedź ⇦	0xAA	0x106	0x1	HARD1	HARD2	HVER	ATYPE	AVERS	FVERS	BVER1	BREV2	SUMKTRL	0xA5

HARD1 | HARD2 – 0x3000 – firmware dla procesora uniwersalnego UNIV
HVER – 0x03 – wersja procesora
ATYPE – typ aplikacji
0x01 - przycisk
0x02 - przekaźnik
0x03 – odbiornik podczerwieni
0x04 – czujnik temperatury
0x05 – nadajnik podczerwieni
0x06 – ściemniacz
0x07 – sterownik rolet
0x08 – sterownik LED
AVERS – wersja aplikacji
FVERS – wersja firmware
BVER1 | BREV2 – wersja bootloader-a BVER1.BREV2

0x1F1 – *Błąd firmware*

Jeżeli wgrany firmware jest nieprawidłowy, procesor wyśle poniższą ramkę.

Odpowiedź ⇦	0xAA	0x1F1	0x1	FIRMFLAGS	FSUM2	FSUM1	FSUM0	0xFF	0xFF	BVER1	BREV2	SUMKTRL	0xA5

FIRMFLAGS – kod błędu
bit <0> - błąd firmware
bit <1> - bit nieużywany
bit <2> - nieprawidłowa suma kontrolna firmware
bit <3> - firmware nieodpowiedni dla tej wersji procesora (UNIV 3)
bit <4:7> - bity nieużywane
FSUM2 | FSUM1 | FSUM0 – spodziewana przez bootloader suma kontrolna firmware
BVER1 | BREV2 – wersja bootloader-a BVER1.BREV2

0x10C – *Pytanie do procesora o napięcie zasilania*

Jest to pytanie o napięcie zasilające procesor. W odpowiedzi procesor wyśle wartości napięć na magistrali HAPCAN i zasilającym procesor.

Do portu UART ⇨	0xAA	0x10C	0x0	0xD0	0xA5								
Odpowiedź ⇦	0xAA	0x10C	0x1	VOLBUS1	VOLBUS2	VOLCPU1	VOLCPU2	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

VOLBUS1 | VOLBUS2 – napięcie magistrali U_{BUS}=(VOLCPU1*256+VOLCPU2)*30,5/65472
VOLCPU1 | VOLCPU2 – napięcie rdzenia procesora U_{CPU}=(VOLCPU1*256+VOLCPU2)*5/65472

0x10E – *Pytanie do procesora o opis*

Jest to pytanie o definiowany przez użytkownika 16-to znakowy opis procesora.

Do portu UART ⇨	0xAA	0x10E	0x0	0xF0	0xA5								
Odpowiedź ⇦	0xAA	0x10E	0x1	abc0	abc1	abc2	abc3	abc4	abc5	abc6	abc7	SUMKTRL	0xA5
	0xAA	0x10E	0x1	abc8	abc9	abc10	abc11	abc12	abc13	abc14	abc15	SUMKTRL	0xA5

abcx – 16 znaków opisu procesora

0x111 – *Pytanie do procesora o DEV ID*
Jest to pytanie o numer identyfikacyjny procesora fabrycznie zapisany przez Microchip. W odpowiedzi procesor wyśle dane, w których zawarte są: typ procesora i wersja uaktualnienia. Więcej informacji nt DEV ID znajduje się w dokumentacji procesora PIC18F26K80 firmy Microchip.

Do portu UART ⇨	0xAA	0x111	0x0	0x21	0xA5								
Odpowiedź ⇦	0xAA	0x111	0x1	DEV ID1	DEV ID2	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

DEV ID1 | DEV ID2 - fabryczny numer identyfikacyjny procesora firmy Microchip

5.4. Wiadomości systemowe CAN

Są to wiadomości używane do komunikacji z bootloader-em poprzez magistralę HAPCAN. Komunikacja z magistralą wymaga komputera PC i interfejsu HAPCAN RS232C lub ethernet-owego. Tylko część z wiadomości systemowych obsługiwana jest przez bootloader (Tabela 5). Pozostałe mogą być zaimplementowane w oprogramowaniu funkcyjnym firmware. Poniżej opisany jest sposób komunikacji z bootloader-em procesora poprzez magistralę HAPCAN.

0x100 - pytanie do procesora o wejście w tryb programowania	Wiadomości obsługiwane przez bootloader CAN w trybie normalnym (nie programowania)
0x101 - pytanie do grupy procesorów o restart	
0x102 - pytanie do procesora o restart	
0x103 - pytanie do grupy procesorów o hardware	
0x104 - pytanie do procesora hardware	
0x105 - pytanie do grupy procesorów o firmware	
0x106 - pytanie do procesora o firmware	
0x107 - pytanie do procesora o ustawienie domyślnych numerów modułu i grupy	
0x108 - pytanie do grupy procesorów o status	
0x109 - pytanie do procesora o status	
0x10A - wiadomość sterująca	
0x10B - pytanie do grupy procesorów o napięcie zasilania	Wiadomości obsługiwane przez bootloader CAN w trybie normalnym (nie programowania)
0x10C - pytanie do procesora o napięcie zasilania	
0x10D - pytanie do grupy procesorów o opis	
0x10E - pytanie do procesora o opis	
0x10F - pytanie do grupy procesorów o DEV ID	
0x111 - pytanie do procesora o DEV ID	
0x112 - pytanie do grupy procesorów o czas od startu	
0x113 - pytanie do procesora o czas od startu	
0x114 - pytanie do grupy procesorów o diagnostykę	
0x115 - pytanie do procesora o diagnostykę	

Tabela 5. Lista wiadomości systemowych używanych do komunikacji przez magistralę HAPCAN

START	RAMKA	FLAGI	NR MOD	NR GRUP	D0	D1	D2	D3	D4	D5	D6	D7	SUMKTRL	STOP

START - bajt startu 0xAA
RAMKA - typ ramki
FLAGI - bity informacyjne
bit <0> flaga odpowiedzi (ODP). Jeśli ramka została wysłana w skutek zapytania o stan to ODP = 1, inaczej ODP = 0
bit <3:1> flagi nieużywane
NR MOD - numer modułu nadającego
NR GRUP - numer grupy modułu nadającego
Dx - bajty danych
SUMKTRL - suma wartości bajtów pomiędzy bajtem startu i sumy kontrolnej
STOP - bajt stopu 0xA5

Tabela 6. Budowa ramki do komunikacji przez magistralę HAPCAN

0x100 – *Pytanie do procesora o wejście w tryb programowania*
Jest to pytanie o wejście w tryb programowania, w którym możliwe jest dokonywanie zmian w pamięci danych i programu procesora. W odpowiedzi procesor wyśle ramkę potwierdzającą wykonanie polecenia. Wyjście z trybu programowania opisane jest w sekcji *5.6. Wiadomości w trybie programowania CAN*.

Do portu CAN ⇨	0xAA	0x100	0x0	NR MOD	NR GRUP	0xXX	0xXX	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x100	0x1	MODUŁ	GRUPA	0xFF	0xFF	BVER1	BVER2	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

NR MOD - numer modułu nadającego
MODUŁ - numer modułu odpowiadającego
NR GRUP - numer grupy modułu nadającego
GRUPA - numer grupy modułu odpowiadającego
0xXX - dane nieistotne mogą być dowolnej wartości
MODUŁ - numer pytanego modułu
GRUPA - numer grupy pytanego modułu
BVER1 | BVER2 - wersja bootloader-a w formacie BVER1.BVER2

0x101 – *Pytanie do grupy procesorów o restart*

Wiadomość spowoduje restart procesorów w danej grupie lub wszystkich grupach. Procesory nie wyślą żadnych odpowiedzi.

restart jednej grupy

Do portu CAN ⇨	0xAA	0x101	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	BRAK														

restart wszystkich grup

Do portu CAN ⇨	0xAA	0x101	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	0x00	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	BRAK														

0x00	- oznacza wszystkie moduły w grupie
GRUPA	- numer grupy do restartowania
0x00	- oznacza wszystkie grupy

0x102 – *Pytanie do procesora o restart*

Wiadomość spowoduje restart jednego procesora. Procesor nie wyśle żadnej odpowiedzi.

Do portu CAN ⇨	0xAA	0x102	0x0	NR MOD	NR GRUP	0xXX	0xXX	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	BRAK														

MODUŁ	- numer modułu do zrestartowania
GRUPA	- numer grupy modułu do zrestartowania

0x103 – *Pytanie do grupy procesorów o hardware*

Jest to pytanie o dane urządzenia. W odpowiedzi procesory wyślą typy urządzeń, wersje i numery seryjne.

pytanie do wszystkich modułów w jednej grupie

Do portu CAN ⇨	0xAA	0x103	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x103	0x1	MODUŁ	GRUPA	HARD1	HARD2	HVER	0xFF	ID0	ID1	ID2	ID3	SUMKTRL	0xA5

pytanie do wszystkich grup

Do portu CAN ⇨	0xAA	0x103	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	0x00	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x103	0x1	MODUŁ	GRUPA	HARD1	HARD2	HVER	0xFF	ID0	ID1	ID2	ID3	SUMKTRL	0xA5

NR MOD	- numer modułu nadającego
MODUŁ	- numer modułu odpowiadającego
NR GRUP	- numer grupy modułu nadającego
GRUPA	- numer grupy modułu odpowiadającego
0xXX	- dane nieistotne mogą być dowolnej wartości
HARD1	HARD2 - 0x3000 – procesor uniwersalny UNIV
0x00	- oznacza wszystkie moduły w grupie
HVER	- 0x03 – wersja procesora
GRUPA	- numer pytanej grupy
0x00	- oznacza wszystkie grupy
IDx	- numer seryjny

0x104 – *Pytanie do procesora o hardware*

Jest to pytanie o dane urządzenia. W odpowiedzi procesor wyśle typ urządzenia, wersję i numer seryjny.

Do portu CAN ⇨	0xAA	0x104	0x0	NR MOD	NR GRUP	0xXX	0xXX	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x104	0x1	MODUŁ	GRUPA	HARD1	HARD2	HVER	0xFF	ID0	ID1	ID2	ID3	SUMKTRL	0xA5

NR MOD	- numer modułu nadającego
MODUŁ	- numer modułu odpowiadającego
NR GRUP	- numer grupy modułu nadającego
GRUPA	- numer grupy modułu odpowiadającego
0xXX	- dane nieistotne mogą być dowolnej wartości
HARD1	HARD2 - 0x3000 – procesor uniwersalny UNIV
MODUŁ	- numer pytanego modułu
HVER	- 0x03 – wersja procesora
GRUPA	- numer grupy pytanego modułu
IDx	- numer seryjny

0x105 – *Pytanie do grupy procesorów o firmware*

Jest to pytanie o oprogramowania funkcjonalnego. W odpowiedzi procesory wyślą typy i wersje wgranych oprogramowań. Jeśli wgrane firmware są nieprawidłowe, procesory wyślą ramki błędu firmware.

pytanie do wszystkich modułów w jednej grupie

Do portu CAN ⇨	0xAA	0x105	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x105	0x1	MODUŁ	GRUPA	HARD1	HARD2	HVER	ATYPE	AVERS	FVERS	BVER1	BVER2	SUMKTRL	0xA5

pytanie do wszystkich grup

Do portu CAN ⇨	0xAA	0x105	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	0x00	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x105	0x1	MODUŁ	GRUPA	HARD1	HARD2	HVER	ATYPE	AVERS	FVERS	BVER1	BVER2	SUMKTRL	0xA5

NR MOD - numer modułu nadającego
MODUŁ - numer modułu odpowiadającego
NR GRUP - numer grupy modułu nadającego
GRUPA - numer grupy modułu odpowiadającego
0xXX - dane nieistotne mogą być dowolnej wartości
HARD1 HARD2 - 0x3000 – procesor uniwersalny UNIV
0x00 - oznacza wszystkie moduły w grupie
HVER - 0x03 – wersja procesora
GRUPA - numer pytanej grupy
0x00 - oznacza wszystkie grupy
ATYPE - typ aplikacji
0x01 - przycisk
0x02 - przekaźnik
0x03 – odbiornik podczerwieni
0x04 – czujnik temperatury
0x05 – nadajnik podczerwieni
0x06 – ściemniacz
0x07 – sterownik rolet
0x08 – sterownik LED
AVERS - wersja aplikacji
FVERS - wersja firmware
BVER1 BVER2 - wersja bootloadera BVER1.BVER2

0x106 – *Pytanie do procesora o firmware*

Jest to pytanie o oprogramowania funkcjonalnego W odpowiedzi procesor wyśle typ i wersję aktualnie wgranego oprogramowania. Jeśli wgrany firmware jest nieprawidłowy, procesor wyśle ramkę błędu firmware.

Do portu CAN ⇨	0xAA	0x106	0x0	NR MOD	NR GRUP	0xXX	0xXX	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x106	0x1	MODUŁ	GRUPA	HARD1	HARD2	HVER	ATYPE	AVERS	FVERS	BVER1	BVER2	SUMKTRL	0xA5

NR MOD - numer modułu nadającego
MODUŁ - numer modułu odpowiadającego
NR GRUP - numer grupy modułu nadającego
GRUPA - numer grupy modułu odpowiadającego
0xXX - dane nieistotne mogą być dowolnej wartości
HARD1 HARD2 - 0x3000 – procesor uniwersalny UNIV
MODUŁ - numer pytanego modułu
HVER - 0x03 – wersja procesora
GRUPA - numer grupy pytanego modułu
ATYPE - typ aplikacji
0x01 - przycisk
0x02 - przekaźnik
0x03 – odbiornik podczerwieni
0x04 – czujnik temperatury
0x05 – nadajnik podczerwieni
0x06 – ściemniacz
0x07 – sterownik rolet
0x08 – sterownik LED
AVERS - wersja aplikacji
FVERS - wersja firmware
BVER1 BVER2 - wersja bootloadera BVER1.BVER2

0x1F1 – *Ramka błędu firmware*

Jeżeli wgrany firmware jest nieprawidłowy, procesor wyśle poniższą ramkę.

Odpowiedź ⇦	0xAA	0x1F1	0x1	MODUŁ	GRUPA	FIRFLAGS	FSUM2	FSUM1	FSUM0	0xFF	0xFF	BVER1	BVER2	SUMKTRL	0xA5

MODUŁ - numer modułu odpowiadającego
GRUPA - numer grupy modułu odpowiadającego
FIRFLAGS - kod błędu
bit <0> - błąd firmware
bit <1> - bit nieużywany
bit <2> - nieprawidłowa suma kontrolna firmware
bit <3> - firmware nieodpowiedni dla tej wersji procesora (UNIV 3)
bit <4:7> - bity nieużywane
FSUM2 FSUM1 FSUM0 - spodziewana przez bootloader suma kontrolna firmware
BVER1 BVER2 - wersja bootloadera BVER1.BVER2

0x107 – *Pytanie do procesora o ustawienie domyślnych numerów modułu i grupy*

Jest to pytanie o zmianę numerów modułu i grupy (identyfikatora modułu w sieci). Numery modułu i grupy zostaną zamienione na domyślne (odpowiednio na bajty ID2 i ID3 numeru seryjnego). Funkcja użyteczna, gdy dwa lub więcej modułów w sieci ma nadany ten sam identyfikator. Funkcja ta pozwala wtedy na ich rozróżnienie. W odpowiedzi procesor wyśle ramkę z aktualnym numer identyfikującym moduł w sieci.

Do portu CAN ⇨	0xAA	0x107	0x0	NR MOD	NR GRUP	0xXX	0xXX	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x107	0x1	ID2	ID3	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

NR MOD - numer modułu nadającego
ID2 - aktualny numer modułu odpowiadającego
NR GRUP - numer grupy modułu nadającego
ID3 - aktualny numer grupy modułu odpowiadającego
0xXX - dane nieistotne mogą być dowolnej wartości
MODUŁ - numer pytanego modułu (przed zmianą)
GRUPA - numer grupy pytanego modułu (przed zmianą)

0x10B – *Pytanie do grupy procesorów o napięcie zasilania*

Jest to pytanie o wartości napięć. W odpowiedzi procesory wyślą wartości napięć odczytywane na magistrali i na zasilaniu samych procesorów.

pytanie do wszystkich modułów w jednej grupie

Do portu CAN ⇨	0xAA	0x10B	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x10B	0x1	MODUŁ	GRUPA	VOLBUS1	VOLBUS2	VOLCPU1	VOLCPU2	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

pytanie do wszystkich grup

Do portu CAN ⇨	0xAA	0x10B	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	0x00	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x10B	0x1	MODUŁ	GRUPA	VOLBUS1	VOLBUS2	VOLCPU1	VOLCPU2	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

- NR MOD – numer modułu nadającego
- MODUŁ – numer modułu odpowiadającego
- NR GRUP – numer grupy modułu nadającego
- GRUPA – numer grupy modułu odpowiadającego
- 0xXX – dane nieistotne mogą być dowolnej wartości
- VOLBUS1, VOLBUS2 – napięcie magistrali U_{BUS}=(VOLCPU1*256+VOLCPU2)*30,5/65472
- 0x00 – oznacza wszystkie moduły w grupie
- VOLCPU1, VOLCPU2 – napięcie rdzenia procesora U_{CPU}=(VOLCPU1*256+VOLCPU2)*5/65472
- GRUPA – numer pytanej grupy
- 0x00 – oznacza wszystkie grupy

0x10C – *Pytanie do procesora o napięcie zasilania*

Jest to pytanie o wartości napięć. W odpowiedzi procesor wyśle wartości napięć na zasilaniu magistrali i zasilaniu samego procesora.

Do portu CAN ⇨	0xAA	0x10C	0x0	NR MOD	NR GRUP	0xXX	0xXX	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x10C	0x1	MODUŁ	GRUPA	VOLBUS1	VOLBUS2	VOLCPU1	VOLCPU2	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

- NR MOD – numer modułu nadającego
- MODUŁ – numer modułu odpowiadającego
- NR GRUP – numer grupy modułu nadającego
- GRUPA – numer grupy modułu odpowiadającego
- 0xXX – dane nieistotne mogą być dowolnej wartości
- VOLBUS1, VOLBUS2 – napięcie magistrali U_{BUS}=(VOLCPU1*256+VOLCPU2)*30,5/65472
- MODUŁ – numer pytanego modułu
- VOLCPU1, VOLCPU2 – napięcie rdzenia procesora U_{CPU}=(VOLCPU1*256+VOLCPU2)*5/65472
- GRUPA – numer grupy pytanego modułu

0x10D – *Pytanie do grupy procesorów o opis*

Jest to pytanie o definiowany przez użytkownika 16-to znakowe opisy procesorów. W odpowiedzi procesory wyślą po dwie ramki, w każdej 8 znaków opisu.

pytanie do wszystkich modułów w jednej grupie

Do portu CAN ⇨	0xAA	0x10D	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x10D	0x1	MODUŁ	GRUPA	abc0	abc1	abc2	abc3	abc4	abc5	abc6	abc7	SUMKTRL	0xA5
	0xAA	0x10D	0x1	MODUŁ	GRUPA	abc8	abc9	abc10	abc11	abc12	abc13	abc14	abc15	SUMKTRL	0xA5

pytanie do wszystkich grup

Do portu CAN ⇨	0xAA	0x10D	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	0x00	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x10D	0x1	MODUŁ	GRUPA	abc0	abc1	abc2	abc3	abc4	abc5	abc6	abc7	SUMKTRL	0xA5
	0xAA	0x10D	0x1	MODUŁ	GRUPA	abc8	abc9	abc10	abc11	abc12	abc13	abc14	abc15	SUMKTRL	0xA5

- NR MOD – numer modułu nadającego
- MODUŁ – numer modułu odpowiadającego
- NR GRUP – numer grupy modułu nadającego
- GRUPA – numer grupy modułu odpowiadającego
- 0xXX – dane nieistotne mogą być dowolnej wartości
- abcx – znaki opisu procesora
- 0x00 – oznacza wszystkie moduły w grupie
- GRUPA – numer pytanej grupy
- 0x00 – oznacza wszystkie grupy

0x10E – *Pytanie do procesora o opis*

Jest to pytanie o definiowany przez użytkownika 16-to znakowy opis procesora. W odpowiedzi procesor wyśle dwie ramki, w każdej 8 znaków opisu.

Do portu CAN ⇨	0xAA	0x10E	0x0	NR MOD	NR GRUP	0xXX	0xXX	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x10E	0x1	MODUŁ	GRUPA	abc0	abc1	abc2	abc3	abc4	abc5	abc6	abc7	SUMKTRL	0xA5
	0xAA	0x10E	0x1	MODUŁ	GRUPA	abc8	abc9	abc10	abc11	abc12	abc13	abc14	abc15	SUMKTRL	0xA5

- NR MOD – numer modułu nadającego
- MODUŁ – numer modułu odpowiadającego
- NR GRUP – numer grupy modułu nadającego
- GRUPA – numer grupy modułu odpowiadającego
- 0xXX – dane nieistotne mogą być dowolnej wartości
- abcx – znaki opisu procesora
- MODUŁ – numer pytanego modułu
- GRUPA – numer grupy pytanego modułu

0x10F – *Pytanie do grupy procesorów o DEV ID*

Jest to pytanie o numery identyfikacyjne procesorów fabrycznie zapisane przez Microchip. W odpowiedzi procesory wyślą dane, w których zawarte są: typy procesorów i wersje ich uaktualnienia. Więcej informacji nt DEV ID znajduje się w dokumentacji procesora PIC18F26K80 firmy Microchip.

pytanie do wszystkich modułów w jednej grupie

Do portu CAN ⇨	0xAA	0x10F	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x10F	0x1	MODUŁ	GRUPA	DEVID1	DEVID2	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

pytanie do wszystkich grup

Do portu CAN ⇨	0xAA	0x10F	0x0	NR MOD	NR GRUP	0xXX	0xXX	0x00	0x00	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x10F	0x1	MODUŁ	GRUPA	DEVID1	DEVID2	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

NR MOD - numer modułu nadającego
MODUŁ - numer modułu odpowiadającego
NR GRUP - numer grupy modułu nadającego
GRUPA - numer grupy modułu odpowiadającego
0xXX - dane nieistotne mogą być dowolnej wartości
DEVID1 DEVID2 - fabryczny numer identyfikacyjny procesora firmy Microchip
0x00 - oznacza wszystkie moduły w grupie
GRUPA - numer pytanej grupy
0x00 - oznacza wszystkie grupy

0x111 – *Pytanie do procesora o DEV ID*

Jest to pytanie o numer identyfikacyjny procesora fabrycznie zapisany przez Microchip. W odpowiedzi procesor wyśle dane, w których zawarte są: typ procesora i wersje jego uaktualnienia. Więcej informacji nt DEV ID znajduje się w dokumentacji procesora PIC18F26K80 firmy Microchip.

Do portu CAN ⇨	0xAA	0x111	0x0	NR MOD	NR GRUP	0xXX	0xXX	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x111	0x1	MODUŁ	GRUPA	DEVID1	DEVID2	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

NR MOD - numer modułu nadającego
MODUŁ - numer modułu odpowiadającego
NR GRUP - numer grupy modułu nadającego
GRUPA - numer grupy modułu odpowiadającego
0xXX - dane nieistotne mogą być dowolnej wartości
DEVID1 DEVID2 - fabryczny numer identyfikacyjny procesora firmy Microchip
MODUŁ - numer pytanego modułu
GRUPA - numer grupy pytanego modułu

5.5. Wiadomości w trybie programowania UART

Wiadomości te umożliwiają wprowadzenie bootloader-a w tryb programowania poprzez port szeregowy UART (RS232) procesora i dokonanie zmian w pamięci programu (pamięci FLASH) i pamięci danych (FLASH i EEPROM). Umożliwiają zatem wgranie oprogramowania firmware i konfigurację procesora przez port UART.

0x020 - pytanie do procesora o wyście z trybu programowania	Wiadomości obsługiwane przez bootloader UART w trybie programowania
0x030 - ramka adresująca	
0x040 - ramka danych	

Tabela 7. Lista wiadomości systemowych używanych w trybie programowania do komunikacji przez port szeregowy UART

0x020 – *Pytanie do procesora o wyjście z trybu programowania*

Jest to pytanie o wyjście z trybu programowania i powrót do pracy normalnej. Po wyjściu z trybu programowania procesor wykonuje restart. Procesor nie wysyła żadnej odpowiedzi.

Do portu UART ⇨	0xAA	0x020	0x0	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	BRAK												

0xXX - dane nieistotne mogą być dowolnej wartości

0x030 – *Ramka adresująca*

W ramce zawarty jest adres komórki pamięci, która ma zostać przeczytana lub zmodyfikowana. Adres musi być wielokrotnością liczby 8 (dla polecenia zapisu FLASH) lub wielokrotnością liczby 64 (dla kasowania pamięci FLASH). Polecenie odczytu i zapisu odczytuje lub zapisuje 8 kolejnych komórek pamięci począwszy od adresu wskazanego w bajtach ADRU:ADRH:ADRL. Przed wykonaniem zapisu do pamięci FLASH musi ona zostać wcześniej skasowana. Polecenie kasowania kasuje 64 komórki począwszy od adresu wskazanego w bajtach ADRU:ADRH:ADRL.

Do portu UART ⇨	0xAA	0x030	0x0	ADRU	ADRH	ADRL	0xXX	0xXX	CMD	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x030	0x1	echo	echo	echo	echo	echo	echo	echo	echo	SUMKTRL	0xA5

ADRU ADRH ADRL - adres komórki pamięci, musi być wielokrotnością 8 dla zapisu FLASH i 64 dla kasowania pamięci FLASH
echo - bajt identyczny jak wysłany
CMD - polecenie odczyt/zapis/kasowanie
0x01 – odczyt pamięci EEPROM lub FLASH
0x02 – zapis pamięci EEPROM lub FLASH
0x03 – kasowanie pamięci FLASH

0x040 – *Ramka danych*
W ramce zawarte są dane, która mają zostać zapisane do komórki pamięci wskazanej w ramce adresującej. Jeśli w ramce adresującej podane zostało polecenie odczytu lub kasowania, wysłanie tej ramki spowoduje odczyt/kasowanie komórek pamięci.

jeśli w ramce adresującej CMD=0x01 (odczyt pamięci EEPROM lub FLASH)

Do portu UART ⇨	0xAA	0x040	0x0	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x040	0x1	DANE R0	DANE R1	DANE R2	DANE R3	DANE R4	DANE R5	DANE R6	DANE R7	SUMKTRL	0xA5

0xXX	- dane nieistotne mogą być dowolnej wartości
DANE Rx	- odebrane odczytane bajty

jeśli w ramce adresującej CMD=0x02 (zapis pamięci EEPROM lub FLASH)

Do portu UART ⇨	0xAA	0x040	0x0	DANE W0	DANE W1	DANE W2	DANE W3	DANE W4	DANE W5	DANE W6	DANE W7	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x040	0x1	DANE R0	DANE R1	DANE R2	DANE R3	DANE R4	DANE R5	DANE R6	DANE R7	SUMKTRL	0xA5

DANE Wx	- bajty do zapisu w pamięci
DANE Rx	- bajty odebrane, potwierdzające poprawność zapisu (powinno być: DANE Rx = DANE Wx)

jeśli w ramce adresującej CMD=0x03 (kasowanie pamięci FLASH)

Do portu UART ⇨	0xAA	0x040	0x0	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x040	0x1	DANE R0	DANE R1	DANE R2	DANE R3	DANE R4	DANE R5	DANE R6	DANE R7	SUMKTRL	0xA5

0xXX	- dane nieistotne mogą być dowolnej wartości
DANE Rx	- odebrane 8 pierwszych bajtów ze skasowanych 64, potwierdzające poprawność skasowania (powinny być równe 0xFF)

0x0F0 – *Ramka błędu*
Ramka błędu zostanie wysłana przez bootloader jeśli zostały podane nieprawidłowe adres lub polecenie w ramce adresowej.

Odpowiedź ⇦	0xAA	0x0F0	0x1	0xFF	0xFF	BVER1	BVER2	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5

BVER1	BVER2	- wersja bootloadera w formacie BVER1.BVER2

5.6. Wiadomości w trybie programowania CAN

Wiadomości te umożliwiają wprowadzenie bootloader-a w tryb programowania poprzez magistralę HAPCAN i dokonanie zmian w pamięci programu (pamięci FLASH) i pamięci danych (FLASH i EEPROM). Umożliwiają zatem wgranie oprogramowania firmware i konfigurację procesora przez magistralę HAPCAN.

0x010 - pytanie do wszystkich procesorów o wyście z trybu programowania	Wiadomości obsługiwane przez bootloader CAN w trybie programowania
0x020 - pytanie do procesora o wyście z trybu programowania	
0x030 - ramka adresująca	
0x040 - ramka danych	

Tabela 8. Lista wiadomości systemowych używanych w trybie programowania do komunikacji przez magistralę HAPCAN

0x010 – *Pytanie do wszystkich procesorów o wyjście z trybu programowania*
Jest to pytanie o wyjście z trybu programowania i powrót do pracy normalnej do wszystkich procesorów na magistrali. Po wyjściu z trybu programowania procesory wykonują restart i nie wysyłają żadnych odpowiedzi.

Do portu CAN ⇨	0xAA	0x010	0x0	0x00	0x00	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	BRAK														

0x00	0x00	- oznacza pytanie skierowane do wszystkich procesorów
	0xXX	- dane nieistotne mogą być dowolnej wartości

0x020 – *Pytanie do procesora o wyjście z trybu programowania*
Jest to pytanie o wyjście z trybu programowania i powrót do pracy normalnej. Po wyjściu z trybu programowania procesor wykonuje restart. Procesor nie wysyła żadnej odpowiedzi.

Do portu CAN ⇨	0xAA	0x020	0x0	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	BRAK														

MODUŁ		- numer modułu programowanego
	GRUPA	- numer grupy modułu programowanego
	0xXX	- dane nieistotne mogą być dowolnej wartości

0x030 – *Ramka adresująca*
W ramce zawarty jest adres komórki pamięci, która ma zostać przeczytana lub zmodyfikowana. Adres musi być wielokrotnością liczby 8 (dla polecenia zapisu FLASH) lub wielokrotnością liczby 64 (dla kasowania pamięci FLASH). Polecenie odczytu i zapisu odczytuje lub zapisuje 8 kolejnych komórek pamięci począwszy od adresu wskazanego w bajtach ADRU:ADRH:ADRL. Przed wykonaniem zapisu do pamięci FLASH musi ona zostać wcześniej skasowana. Polecenie kasowania kasuje 64 komórki począwszy od adresu wskazanego w bajtach ADRU:ADRH:ADRL.

Do portu CAN ⇨	0xAA	0x030	0x0	MODUŁ	GRUPA	ADRU	ADRH	ADRL	0xXX	0xXX	CMD	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x030	0x1	MODUŁ	GRUPA	echo	echo	echo	echo	echo	echo	echo	echo	SUMKTRL	0xA5

MODUŁ - numer modułu programowanego
GRUPA - numer grupy modułu programowanego
ADRU ADRH ADRL - adres komórki pamięci, musi być wielokrotnością 8 dla odczytu i zapisu EEPROM i FLASH lub 64 dla kasowania pamięci FLASH
echo - bajt identyczny jak wysłany
CMD - polecenie odczyt/zapis/kasowanie
0x01 – odczyt pamięci EEPROM lub FLASH
0x02 – zapis pamięci EEPROM lub FLASH
0x03 – kasowanie pamięci FLASH

0x040 – *Ramka danych*

W ramce zawarte są dane, która mają zostać zapisane do komórki pamięci wskazanej w ramce adresującej. Jeśli w ramce adresującej podane zostało polecenie odczytu lub kasowania, wysłanie tej ramki spowoduje odczyt/kasowanie komórek pamięci.

jeśli w ramce adresującej CMD=0x01 (odczyt pamięci EEPROM lub FLASH)

Do portu CAN ⇨	0xAA	0x040	0x0	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x040	0x1	MODUŁ	GRUPA	DANE R0	DANE R1	DANE R2	DANE R3	DANE R4	DANE R5	DANE R6	DANE R7	SUMKTRL	0xA5

MODUŁ - numer modułu programowanego
GRUPA - numer grupy modułu programowanego
0xXX - dane nieistotne mogą być dowolnej wartości
DANE Rx - bajty odebrane

jeśli w ramce adresującej CMD=0x02 (zapis pamięci EEPROM lub FLASH)

Do portu CAN ⇨	0xAA	0x040	0x0	MODUŁ	GRUPA	DANE T0	DANE T1	DANE T2	DANE T3	DANE T4	DANE T5	DANE T6	DANE T7	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x040	0x1	MODUŁ	GRUPA	DANE R0	DANE R1	DANE R2	DANE R3	DANE R4	DANE R5	DANE R6	DANE R7	SUMKTRL	0xA5

MODUŁ - numer modułu programowanego
GRUPA - numer grupy modułu programowanego
DANE Tx - bajty do zapisu w pamięci
DANE Rx - bajty odebrane, potwierdzające poprawność zapisu (powinno być: DANE Rx = DANE Tx)

jeśli w ramce adresującej CMD=0x03 (kasowanie pamięci FLASH)

Do portu CAN ⇨	0xAA	0x040	0x0	MODUŁ	GRUPA	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	SUMKTRL	0xA5
Odpowiedź ⇦	0xAA	0x040	0x1	MODUŁ	GRUPA	DANE R0	DANE R1	DANE R2	DANE R3	DANE R4	DANE R5	DANE R6	DANE R7	SUMKTRL	0xA5

MODUŁ - numer modułu programowanego
GRUPA - numer grupy modułu programowanego
0xXX - dane nieistotne mogą być dowolnej wartości
DANE Rx - odebrane 8 pierwszych bajtów ze skasowanych 64, potwierdzające poprawność skasowania (powinny być równe 0xFF)

0x0F0 – *Ramka błędu*

Ramka błędu zostanie wysłana przez bootloader jeśli zostały podane nieprawidłowe adres lub polecenie w ramce adresowej.

Odpowiedź ⇦	0xAA	0x0F0	0x1	MODUŁ	GRUPA	0xFF	0xFF	BVER1	BVER2	0xFF	0xFF	0xFF	0xFF	SUMKTRL	0xA5
	BRAK														

MODUŁ - numer modułu programowanego
GRUPA - numer grupy modułu programowanego
BVER1 BVER2 - wersja bootloadera w formacie BVER1.BVER2

5.7. Algorytm programowania

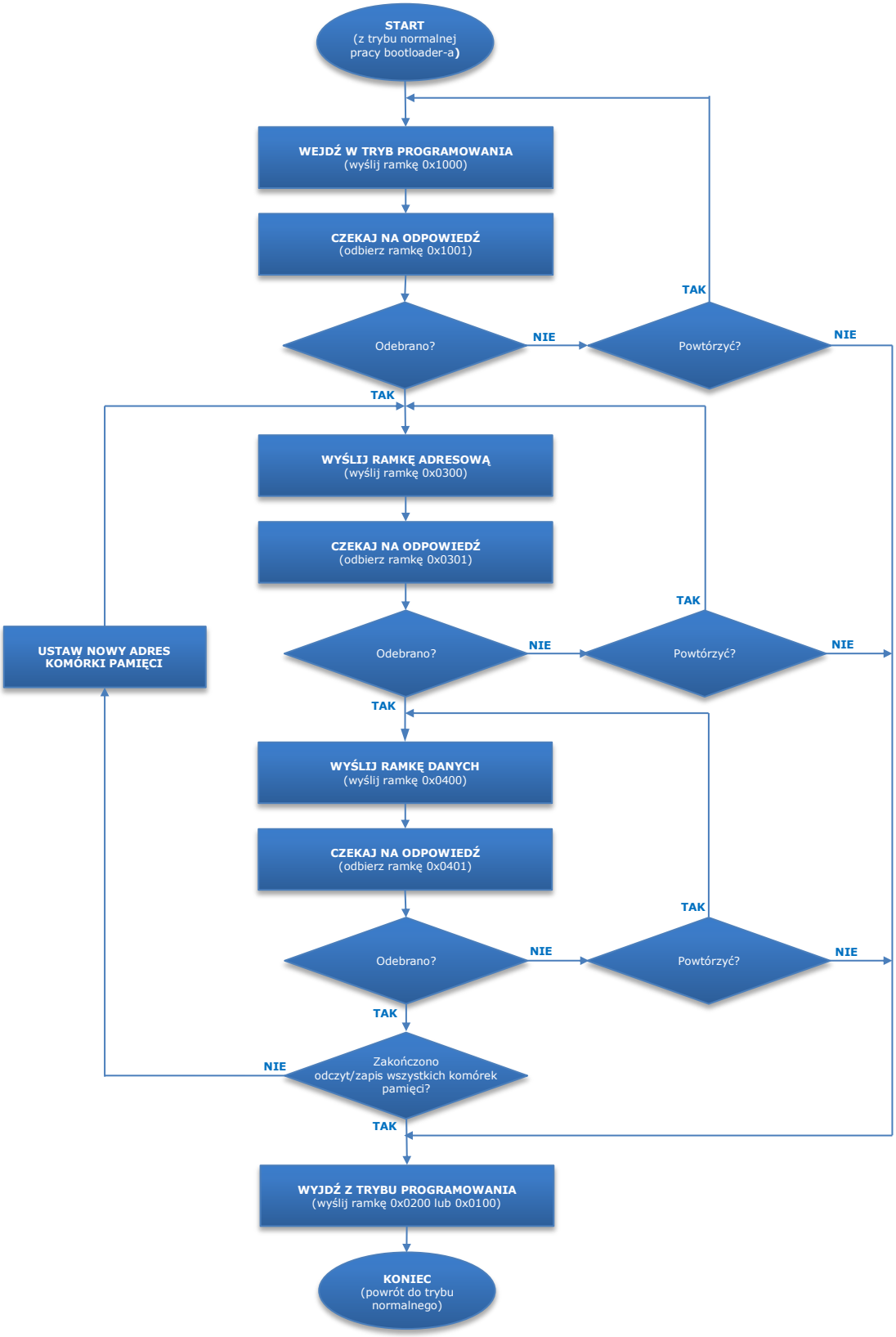
Odczyt, zapis i kasowanie pamięci EEPROM i FLASH procesora w trybie programowania wymaga użycia sekwencji poleceń (Rysunek 13). Cały proces odczytu/zapisu/kasowania pamięci EEPROM I FLASH używa dwóch typów ramek: RAMKI ADRESOWEJ (0x030) i RAMKI DANYCH (0x040). W ramce adresowej musi znaleźć się prawidłowy adres komórki pamięci, który ma zostać odczytany, zapisany lub skasowany oraz polecenie odczytu, zapisu lub skasowania. Ramka danych aktywuje odczyt lub kasowanie komórek pamięci albo musi zawierać dane, które mają zostać zapisane pod wskazanym adresem.

Rozpoczęcie procesu programowania

Aby rozpocząć odczyt lub zapis/kasowanie komórek pamięci procesora, musi on być w trybie programowania. Bootloader procesora wprowadza się w tryb programowania poprzez wysłanie ramki WEJDŹ W TRYB PROGRAMOWANIA (0x100).

Programowanie pamięci programu i danych FLASH

Aby zaprogramować pamięć FLASH należy wybrać adres pomiędzy 0x001000 – 0x00FFF8. Adres musi mieć prawidłową wartość tzn. być wielokrotnością liczby 8 (dla polecenia odczytu i zapisu) lub wielokrotnością liczby 64 (dla polecenia kasowania). Należy też wybrać polecenie 0x01 (aby odczytać pamięć), 0x02 (aby zapisać do pamięci) lub 0x03 (aby skasować pamięć). Zapis do pamięci FLASH musi być poprzedzony skasowaniem tego fragmentu pamięci.



Rysunek 13. Algorytm programowania

Programowanie pamięci danych EEPROM
Aby zaprogramować pamięć EEPROM należy wybrać adres pomiędzy 0xF00000 – 0xF003F8. Należy też wybrać polecenie 0x01 (aby odczytać pamięć), 0x02 (aby zapisać do pamięci). Programowanie pamięci EEPROM nie obsługuje polecenia kasowania. Aby skasować pamięć EEPROM należy użyć polecenia zapisu i jako dane podać wartości „0xFF".

Zakończenie procesu programowania
Bootloader wychodzi z trybu programowania i przechodzi do trybu pracy normalnej po odebraniu ramek 0x010 lub 0x020.

6. Pamięć procesora

Procesor posiada 64kb pamięci programu FLASH, 1kB pamięci danych EEPROM i 3,6kB pamięci operacyjnej RAM. Bootloader rezerwuje część tych pamięci do prawidłowego działania. Pamięć zarezerwowana dla bootloader-a nie może być używana przez oprogramowanie funkcyjne firmware, z wyjątkami opisanymi poniżej.

6.1. Pamięć EEPROM

Bootloader używa kilku komórek pamięci danych EEPROM. Flaga BOOTFL=0xFF umożliwia po restarcie wprowadzenie bootloader-a w tryb programowania. Pozostałe bajty są bajtami konfiguracyjnymi modułu. Są to numery modułu i grupy oraz 16-to znakowy opis. Te komórki pamięci mogą być modyfikowane przez oprogramowanie funkcyjne firmware.

0 1 2 3 4 5 6 7 8 9 A B C D E F		
0xF00000 0xF000F0	256 bajtów pamięci danych EEPROM	0xF0000F 0xF000FF
0xF00100 0xF001F0	256 bajtów pamięci danych EEPROM	0xF0010F 0xF001FF
0xF00200 0xF002F0	256 bajtów pamięci danych EEPROM	0xF0020F 0xF002FF
0xF00300 0xF003F0	256 bajtów pamięci danych EEPROM	0xF0030F 0xF003FF

Tabela 9. Pamięć EEPROM procesora UNIV 3 CPU

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0xF00000	BOOTFL																0xF0000F
0xF00010																	0xF0001F
0xF00020							MODUŁ	GRUPA									0xF0002F
0xF00030	OPIS 0	OPIS 1	OPIS 2	OPIS 3	OPIS 4	OPIS 5	OPIS 6	OPIS 7	OPIS 8	OPIS 9	OPIS 10	OPIS 11	OPIS 12	OPIS 13	OPIS 14	OPIS 15	0xF0003F
0xF00040																	0xF0004F
0xF00050																	0xF0005F
0xF00060																	0xF0006F
0xF00070																	0xF0007F
0xF00080																	0xF0008F
0xF00090																	0xF0009F
0xF000A0																	0xF000AF
0xF000B0																	0xF000BF
0xF000C0																	0xF000CF
0xF000D0																	0xF000DF
0xF000E0																	0xF000EF
0xF000F0																	0xF000FF

BOOTFL - Flaga bootloader-a. Jeśli BOOTFL = 0x00 to po restarcie bootloader będzie w trybie normalnym; Jeśli BOOTFL = 0xFF to bootloader wejdzie w tryb programowania.

MODUŁ - Numer modułu.

GRUPA - Numer grupy modułu

OPIS x - 16 znaków opisu modułu

Tabela 10. Komórki pamięci danych EEPROM używane przez bootloader

6.2. Pamięć FLASH

Kod bootloader-a umieszczony jest na początku pamięci programu FLASH. Zajmuje on obszar 4kB. Ponadto bootloader używa bajtów konfiguracyjnych procesora. Dostęp do tych obszarów jest wyłączony. Obszar oznaczony jako *„28 kB pamięci programu funkcyjnego firmware"* (Tabela 12) jest używany do obliczenia sumy kontrolnej oprogramowania firmware i dlatego nie może zawierać żadnych danych zmienianych w ciągu pracy firmware, a jedynie kod programu. Zmienne dane programu mogą być umieszczone w obszarze oznaczonym jako *"32 kB pamięci danych i programu funkcyjnego firmware"*.

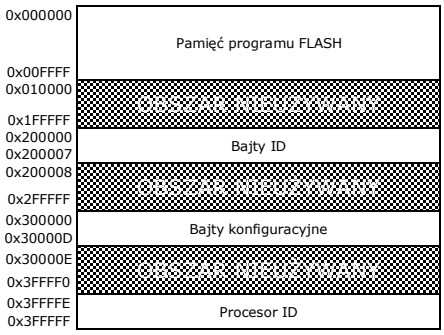


Tabela 11. Pamięć FLASH procesora UNIV 3 CPU

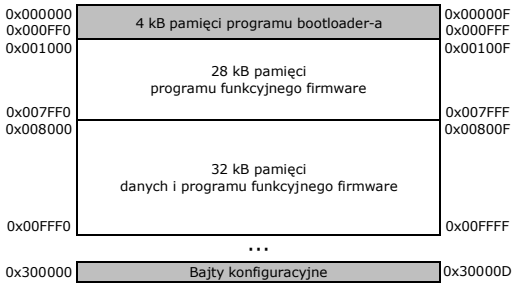


Tabela 12. Obszary pamięci programu i danych FLASH używane przez bootloader

6.3. Pamięć RAM

Bootloader używa większą część banku 1 pamięci operacyjnej RAM procesora UNIV 3. Bajty używane przez bootloader (Tabela 14) mogą być odczytywane przez oprogramowanie funkcyjne firmware i ewentualnie zerowane. Pozostałe obszary pamięci RAM bootloader-a nie powinny być zmieniane, gdyż może to spowodować unieruchomienie, a nawet uszkodzenie modułu.

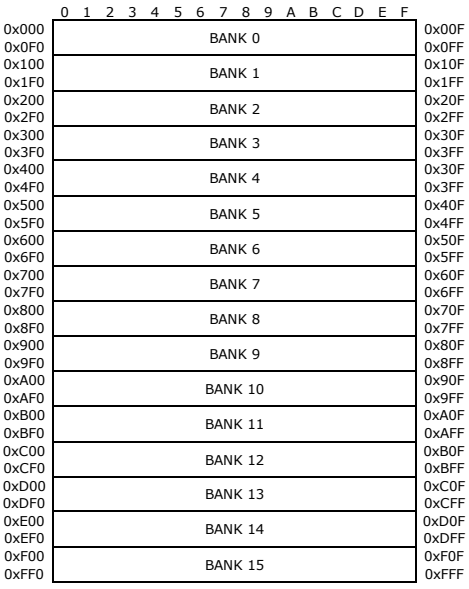


Tabela 13. Pamięć operacyjna RAM procesora UNIV 3 CPU

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0x100	RxBCON	CANFRAME1	CANFRAME2	CANNODE	CANGROUP	CANDLC	CAND0	CAND1	CAND2	CAND3	CAND4	CAND5	CAND6	CAND7	CANFULL	FIRMFLAG	0x10F
0x110	UART0	UART1	UART2	UART3	UART4	UART5	UART6	UART7	UART8	UART9	UART10	UART11	UART12	UART13	UART14	UART15	0x11F
0x120	UART16	UART17	UART18	UART19	UART20	UART21	UART22	UART23	UART24	UART25	UART26	UART27	UART28	UART29	UARTOVF	UARTCNT	0x12F
0x130																	0x13F
0x140																	0x14F
0x150																	0x15F
0x160																UARTON	0x16F
0x170	STATUS_H	WREG_H	BSR_H	PIR1_H	PIR4_H	PIR5_H	CANSTAT_H	CANCON_H	FSR0L_H	FSR0H_H	FSR1L_H	FSR1H_H	TABLAT_H	TBLPTRL_H	TBLPTRH_H	TBLPTRU_H	0x17F
0x180	INTCON_H	EECON1_H	EEDATA_H	EEADRH_H	EEADR_H	TRISA_H	ADCON2_H	ADCON1_H	ADCON0_H	ANCON0_H	ADRESL_H	ADRESH_H	RCSTA1_H	TXSTA1_H	PMD1_H	COMSTAT_H	0x18F
0x190	STATUS_L	WREG_L	BSR_L	PIR1_L	PIR4_L	PIR5_L	CANSTAT_L	CANCON_L	FSR0L_L	FSR0H_L	FSR1L_L	FSR1H_L	TABLAT_L	TBLPTRL_L	TBLPTRH_L	TBLPTRU_L	0x19F
0x1A0	INTCON_L	EECON1_L	EEDATA_L	EEADRH_L	EEADR_L	TRISA_L	ADCON2_L	ADCON1_L	ADCON0_L	ANCON0_L	ADRESL_L	ADRESH_L	RCSTA1_L	TXSTA1_L	PMD1_L	COMSTAT_L	0x1AF
0x1B0																	0x1BF
0x1C0																	0x1CF
0x1D0																	0x1DF
0x1E0																	0x1EF
0x1F0																	0x1FF

RxBCON - CAND7 - bufor odbiorczy CAN

CANFULL - wartość 0xFF sygnalizuje, że odebrana z magistrali CAN wiadomość znajduje się w buforze odbiorczym CAN

FIRMFLAG - flaga zerowana przez bootloader. Ustawiana w firmware funkcjonalnym sygnalizuje zakończenie jego inicjalizacji i gotowość do odbioru wiadomości CAN lub UART.

UART0 - UART29 - bufor odbiorczy UART

UARTOVF - Wartość inna niż 0x00 sygnalizuje przepełnienie bufora

UARTCNT - Podaje liczbę odebranych bajtów przez port szeregowy

UARTON - Wartość 0xFF informuje, że aktywny jest bootloader UART (patrz 4.1. Tryb CAN i UART (32 MHz))

xxxx_H - Kopie rejestrów zapisane przez bootloader w momencie wystąpienia przerwania o wysokim priorytecie

xxxx_L - Kopie rejestrów zapisane przez bootloader w momencie wystąpienia przerwania o niskim priorytecie

Tabela 14. Komórki pamięci operacyjnej RAM używane przez bootloader

6.4. Bajty konfiguracyjne

Bajty konfiguracyjne są podstawową konfiguracją procesora. Znajdują się w obszarze 0x300000 – 0x30000D pamięci FLASH. Zmiana ich wartości nie jest możliwa. Szczegółowy opis znaczenia tych bajtów znajduje się w dokumentacji procesora PIC18F26K80.

	Rejestr	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Wartość	
0x300000	CONFIG1L	-	XINST	-	SOSCSEL1	SOSCSEL0	INTOSCSEL	-	$\overline{\text{RETEN}}$	15h	-0-1 01-1
0x300001	CONFIG1H	IESO	FCMEN	-	PLLCFG	FOSC3	FOSC2	FOSC1	FOSC0	03h	00-0 0011
0x300002	CONFIG2L	-	BORPWR1	BORWPR0	BORV1	BORV0	BOREN1	BOREN0	$\overline{\text{PWRTEN}}$	2Ah	-010 1010
0x300003	CONFIG2H	-	WDTPS4	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN1	WDTEN0	2Eh	-010 1110
0x300005	CONFIG3H	MCLRE	-	-	-	MSSPMSK	-	-	CANMX	89h	1--- 1--1
0x300006	CONFIG4L	$\overline{\text{DEBUG}}$	-	-	BBSIZ0	-	-	-	STVREN	91h	1--1 ---1
0x300008	CONFIG5L	-	-	-	-	CP3	CP2	CP1	CP0	00h	---- 0000
0x300009	CONFIG5H	CPD	CPB	-	-	-	-	-	-	00h	00-- ----
0x30000A	CONFIG6L	-	-	-	-	WRT3	WRT2	WRT1	WRT0	0Fh	---- 1111
0x30000B	CONFIG6H	WRTD	WRTB	WRTC	-	-	-	-	-	80h	100- ----
0x30000C	CONFIG7L	-	-	-	-	EBTR3	EBTR2	EBTR1	EBTR0	0Fh	---- 1111
0x30000D	CONFIG7H	-	EBTRB	-	-	-	-	-	-	00h	-0-- ----

Tabela 15. Wartości bajtów konfiguracyjnych procesora UNIV 3 CPU

7. Oprogramowanie funkcyjne firmware

Urządzenie zbudowane w oparciu o procesor UNIV 3 CPU wymaga oprogramowania, które będzie realizowało funkcje tego urządzenia. Oprogramowania funkcyjne dla procesora są dostępne na stronach poszczególnych urządzeń Projektu HAPCAN hapcan.com.

7.1. Własne oprogramowanie funkcyjne

Istnieje możliwość dopasowania funkcjonalności oprogramowania do własnych potrzeb poprzez modyfikację gotowych programów lub stworzenie własnego oprogramowania od podstaw.
Do tworzenia lub modyfikacji programu potrzebne są:

1. Oprogramowanie MPLAB firmy Microchip do pisania i kompilowania kodu. Program MPLAB dostępny jest bezpłatnie na stronie microchip.com.
2. Szczegółowe informacje o procesorze PIC18F26K80, na którym zbudowany jest UNIV 3 CPU. Te również dostępne są na stronie microchip.com.

3. Konwerter kodu wynikowego (plik .hex) otrzymanego z programu MPLAB na plik .haf (hapcan automation firmware). Konwerter dostępny jest na stronie hapcan.com.
4. Program HAPCAN Programator do wgrania do procesora nowego firmware. HAPCAN Programator dostępny jest na stronie hapcan.com.

7.2. Pamięć programu funkcyjnego

Program funkcyjny może zajmować obszar pamięci FLASH od adresu 0x001000 do 0x00FFFF (Tabela 16). W obszarze 0x001000 – 0x007FFF nie mogą znaleźć się żadne zmienne dane, gdyż obszar ten jest analizowany do obliczenia sumy kontrolnej firmware.

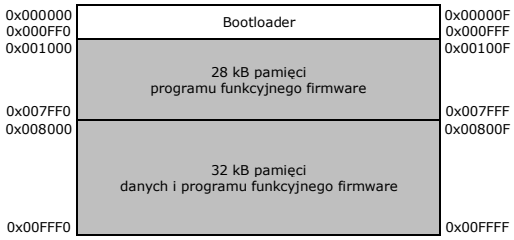
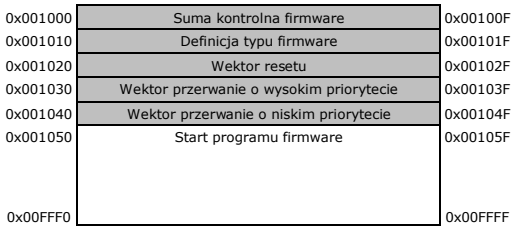


Tabela 16. Obszar pamięci dla programu funkcyjnego

W celu zapewnienia prawidłowej komunikacji bootloadera z oprogramowaniem funkcjonalnym, w programie funkcyjnym muszą być zadeklarowane pewne wartości (suma kontrolna, typ firmware) i przestrzegane obszary (wektory resetu i przerwań) (Tabela 17).



Suma kontrolna firmware	– wartość ta jest obliczana i zapisywana automatycznie w firmware podczas konwersji kodu pliku .hex otrzymanego z programu MPLAB do pliku .haf (hapcan automation firmware).
Definicja typu firmware	– dane dotyczące programu funkcyjnego: typ, wersja, uaktualnienie. Dokładny opis tych wartości znajduje się w szablonie kodu, w sekcji *7.6. Szablon kodu oprogramowania funkcjonalnego*.
Wektor resetu	– miejsce od którego będzie wykonywany program po resecie procesora.
Wektor przerwania o wysokim priorytecie	– miejsce od którego będzie wykonywany program, gdy procesor zgłosi przerwanie o wysokim priorytecie.
Wektor przerwania o niskim priorytecie	– miejsce od którego będzie wykonywany program, gdy procesor zgłosi przerwanie o niskim priorytecie.
Start programu firmware	– miejsce od którego powinien zaczynać się program funkcyjny.

Tabela 17. Obszary specjalne w pamięci programu funkcyjnego

7.3. Pamięć danych oprogramowania funkcyjnego

Dane oprogramowania funkcyjnego mogą być umieszczone w pamięci RAM, EEPROM i FLASH, ale poza obszarem używanym przez bootloader. Ponadto w obszarze 0x001000 – 0x007FFF nie mogą znaleźć się żadne zmienne dane, gdyż obszar ten jest analizowany do obliczenia sumy kontrolnej firmware.

7.4. Odbiór wiadomości UART

Bootloader obsługuje port szeregowy UART1 na wyjściach PIN17 (TX1) i PIN18 (RX1) procesora UNIV 3 CPU. Odbiór wiadomości z portu szeregowego obsługiwany jest przez przerwanie o wysokim priorytecie. Domyślnie port szeregowy ustawiony jest na odbiór transmisji z następującymi parametrami: prędkość transmisji 115200 bps, 8 bitów danych, 1 bit stopu.

Port rozpoczyna odbiór w momencie pojawienia się pierwszego bajta. Zostaje on zapisany do bufora i port oczekuje na kolejny bajt. Maksymalny czas oczekiwania na kolejny bajt równy jest czasowi transmisji jednego bajta powiększonemu o 20%. Dla transmisji o prędkości 115200 bps czas ten wynosi on około 90us. Kolejne bajty zapisywane są do bufora, którego maksymalna pojemność to 30 bajtów. Odbiór z portu szeregowego kończy się

kiedy przerwa po odebraniu ostatniego bajta jest większa niż 90us. Odebrane dane zostają zapisane w buforze odbiorczym UART (bajty od UART0 do UART29 – adres 0x110-0x12D pamięci RAM) (Tabela 14). Ilość odebranych i zapisanych bajtów w buforze sygnalizowana jest wartością w rejestrze UARTCNT (adres 0x12F) (Tabela 14). Jeśli port szeregowy odbierze więcej niż 30 bajtów, wtedy 31-wszy zapisywany jest ponownie na początku bufora odbiorczego (tj. od bajta UART0) i każde takie przepełnienie sygnalizowane jest zwiększeniem o 1 wartości rejestru UARTOVF (adres 0x12E) (Tabela 14).
Po zakończeniu odbioru bootloader sprawdza czy odebrana wiadomość jest wiadomością systemową (*5.3. Wiadomości systemowe UART).* Jeśli tak, wysyła odpowiedź i następnie przekazuje przerwanie o wysokim priorytecie do oprogramowania funkcyjnego (adres 0x001030 pamięci programu, patrz Tabela 17). Jeśli odebrana wiadomość nie jest systemową, przerwanie jest przekazywane natychmiast pod adres 0x001030.
Oprogramowanie funkcyjne firmware powinno sprawdzić bajt UARTCNT i jeśli jest różny od zera, odczytać odpowiednią ilość danych z bufora odbiorczego (UART0-UART29). Odczyt danych z bufora musi zakończyć się wyzerowaniem rejestru UARTCNT.

7.5. Odbiór wiadomości CAN

Bootloader obsługuje port CAN na wyjściach PIN23 (CANTX) i PIN24(CANRX) procesora UNIV 3 CPU. Odbiór wiadomości z portu CAN obsługiwany jest przez przerwanie o niskim priorytecie. Domyślne parametry portu to: prędkość transmisji 125kbps, format CAN 2.0B (tylko ramki z rozszerzonym, 29-bitowym identyfikatorem), stała ilość 8 bajtów danych.

Prawidłowo odebrana wiadomość CAN zostaje zapisana do bufora odbiorczego CAN (bajty od CANFRAME1 do CAND7 – adres 0x101-0x10D pamięci RAM) (Tabela 14). Odebranie wiadomości sygnalizowane jest też wartością 0xFF bajta CANFULL (adres 0x10E) (Tabela 14).

Po zakończeniu odbioru bootloader sprawdza czy odebrana wiadomość jest wiadomością systemową (*5.4. Wiadomości systemowe CAN)*. Jeśli tak, wysyła odpowiedź i następnie przekazuje przerwanie o niskim priorytecie do oprogramowania funkcyjnego (adres 0x001040 pamięci programu, patrz Tabela 17). Jeśli odebrana wiadomość nie jest systemową, przerwanie jest przekazywane natychmiast pod adres 0x001040.

Oprogramowanie funkcyjne firmware powinno sprawdzić bajt CANFULL i jeśli jest równy 0xFF, odczytać bufor odbiorczy CAN. Odczyt danych z bufora musi zakończyć się wyzerowaniem rejestru CANFULL.

7.6. Szablon kodu oprogramowania funkcjonalnego

WZÓR KODU OPROGRAMOWANIA FUNKCYJNEGO FIRMWARE

```
;=====================================================================
;   HAPCAN - Home Automation Project Firmware (http://hapcan.com)
;   Copyright (C) 2013 hapcan.com
;
;   This program is free software: you can redistribute it and/or modify
;   it under the terms of the GNU General Public License as published by
;   the Free Software Foundation, either version 3 of the License, or
;   (at your option) any later version.
;
;   This program is distributed in the hope that it will be useful,
;   but WITHOUT ANY WARRANTY; without even the implied warranty of
;   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;   GNU General Public License for more details.
;
;   You should have received a copy of the GNU General Public License
;   along with this program.  If not, see <http://www.gnu.org/licenses/>.
;=====================================================================
;   Filename:               univ3-template-rev2.asm
;   Associated diagram:     none
;   Author:                 Jacek Siwilo
;   Note:                   This is a template file for UNIV 3 CPU processor
;=====================================================================
;   Revision History
;   Rev:  Date:     Details:
;   0     01.2013   Original version
;   1     03.2013   Minor changes
;   2     09.2013   Minor changes
;=====================================================================
;===  FIRMWARE DEFINITIONS  ==========================================  ;Definicje programu (w tej sekcji należy określić typ, numer wersji i numer
;=====================================================================  ;uaktualnienia oprogramowania
    #define    ATYPE    .255                    ;application type [0-255]  ;Typ aplikacji (urządzenia) - wartość 255 dla aplikacji użytkownika
    #define    AVERS    .0                    ;application version [0-255]  ;Wersja urządzenia (możliwe wartości 0 - 255)
    #define    FVERS    .0                      ;firmware version [0-255]  ;Wersja oprogramowania dla powyższego urządzenia (możliwe wartości 0 - 255)

    #define    FREV     .0                    ;firmware revision [0-65536]  ;Uaktualnienie oprogramowania (możliwe wartości 0 - 65536)
;=====================================================================
;===  NEEDED FILES  ==================================================  ;Deklaracje plików wchodzących w skład projektu
;=====================================================================
    LIST P=18F26K80                          ;directive to define processor  ;Deklaracja procesora
    #include <P18F26K80.INC>          ;processor specific variable definitions  ;Plik ze zmiennymi procesora
    #include "univ3-template-rev2.inc"                    ;project variables  ;Plik ze zmiennymi oprogramowania
INCLUDEDFILES   code
    #include "univ3-fake_bootloader-rev2.inc"        ;fake bootloader only for  ;Plik fałszywego bootloadera (do celów debugowania oprogramowania)
;                                                             debugging
;=====================================================================
;===  FIRMWARE CHECKSUM  =============================================  ;Suma kontrolna oprogramowania umieszczona w programie
;=====================================================================  ;Wartość ta zostanie automatycznie dodana podczas konwersji pliku .hex do
FIRMCHKSM   code    0x001000                                             ;do pliku .haf (nie jest wymagana żadna zmiana)
    DB      0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF

;=====================================================================
;===  FIRMWARE ID  ===================================================  ;Dane firmware umieszczone w programie
;=====================================================================
FIRMID      code    0x001010
    DB      0x30, 0x00, 0x03,ATYPE,AVERS,FVERS,FREV>>8,FREV
;            |     |     |     |     |     |     |_____|_____ firmware revision
;            |     |     |     |     |     |_________________ firmware version
;            |     |     |     |     |_____________________ application version
;            |     |     |     |___________________________ application type
;            |     |     |_____________________________ hardware version '3'
;            |_____|_____|_____|_____|_____|_____|_____|_____ hardware type 'UNIV'
```

```
;=========================================================================
;===  MOVED VECTORS  =====================================================   ;wektory
;=========================================================================
;PROGRAM RESET VECTOR                                                        ;Wektor resetu - miejsce od którego będzie wykonywany program po resecie
FIRMRESET   code    0x1020                                                  ;procesora
        goto    Main
;PROGRAM HIGH PRIORITY INTERRUPT VECTOR                                      ;Wektor przerwania o wysokim priorytecie - miejsce od którego będzie
FIRMHIGHINT code    0x1030                                                  ;wykonywany program, gdy procesor zgłosi przerwanie o wysokim priorytecie
        call    HighInterrupt
        retfie
;PROGRAM LOW PRIORITY INTERRUPT VECTOR                                       ;Wektor przerwania o niskim priorytecie - miejsce od którego będzie
FIRMLOWINT  code    0x1040                                                  ;wykonywany program, gdy procesor zgłosi  przerwanie o niskim priorytecie
        call    LowInterrupt
        retfie

;=========================================================================
;===  FIRMWARE STARTS  ===================================================   ;Miejsce od którego zaczyna się program funkcyjny
;=========================================================================
FIRMSTART   code    0x001050
;-------------------------------------------------------------------------
;---  LOW PRIORITY INTERRUPT  --------------------------------------------   ;Procedura obsługi przerwań o niskim priorytecie
;-------------------------------------------------------------------------
LowInterrupt
        movff   STATUS,STATUS_LOW       ;save STATUS register               ;Zachowanie rejestrów używanych podczas przerwania
        movff   WREG,WREG_LOW           ;save working register
        movff   BSR,BSR_LOW             ;save BSR register
        movff   FSR0L,FSR0L_LOW         ;save other registers used in high int
        movff   FSR0H,FSR0H_LOW
        movff   FSR1L,FSR1L_LOW
        movff   FSR1H,FSR1H_LOW

    ;main firmware ready flag                                               ;Sprawdzenie czy inicjalizacja oprogramowanie została zakończona
        banksel FIRMREADY
        btfss   FIRMREADY,0
        bra     ExitLowInterrupt        ;main firmware is not ready yet
    ;CAN buffer                                                             ;Sprawdzenie czy została odebrana wiadomość z magistrali CAN
        banksel CANFULL
        btfsc   CANFULL,0
        call    WriteToCanRxFIFO        ;receive and save message in RAM

ExitLowInterrupt
        movff   BSR_LOW,BSR             ;restore BSR register               ;Powrót rejestrów używanych podczas przerwania do stanu przed przerwaniem
        movff   WREG_LOW,WREG           ;restore working register
        movff   STATUS_LOW,STATUS       ;restore STATUS register
        movff   FSR0L_LOW,FSR0L         ;restore other registers used in high int
        movff   FSR0H_LOW,FSR0H
        movff   FSR1L_LOW,FSR1L
        movff   FSR1H_LOW,FSR1H
    return

;-------------------------------------------------------------------------
;---  HIGH PRIORITY INTERRUPT  -------------------------------------------   ;Procedura obsługi przerwań o wysokim priorytecie
;-------------------------------------------------------------------------
HighInterrupt
        movff   STATUS,STATUS_HIGH      ;save STATUS register               ;Zachowanie rejestrów używanych podczas przerwania
        movff   WREG,WREG_HIGH          ;save working register
        movff   BSR,BSR_HIGH            ;save BSR register
        movff   FSR0L,FSR0L_HIGH        ;save other registers used in high int
        movff   FSR0H,FSR0H_HIGH
        movff   FSR1L,FSR1L_HIGH
        movff   FSR1H,FSR1H_HIGH

    ;main firmware ready flag                                               ;Sprawdzenie czy inicjalizacja oprogramowanie została zakończona
        banksel FIRMREADY
        btfss   FIRMREADY,0
        bra     ExitHighInterrupt       ;main firmware is not ready yet
    ;uart                                                                   ;Sprawdzenie czy została odebrana wiadomość z portu szeregowego UART
        banksel UARTCNT
        tstfsz  UARTCNT                 ;check if UART received anything
        call    WriteToUartRxFIFO       ;receive and save message in RAM

ExitHighInterrupt
        movff   BSR_HIGH,BSR            ;restore BSR register               ;Powrót rejestrów używanych podczas przerwania do stanu przed przerwaniem
        movff   WREG_HIGH,WREG          ;restore working register
        movff   STATUS_HIGH,STATUS      ;restore STATUS register
        movff   FSR0L_HIGH,FSR0L        ;restore other registers used in high int
        movff   FSR0H_HIGH,FSR0H
        movff   FSR1L_HIGH,FSR1L
        movff   FSR1H_HIGH,FSR1H
    return

;=========================================================================
;===  MAIN PROGRAM  ======================================================   ;Program główny
;=========================================================================
Main:
    ;disable global interrupt for startup                                  ;Wyłączenie przerwań na czas inicjalizacji programu funkcyjnego
        bcf     INTCON,GIEH             ;disable high interrupt
        bcf     INTCON,GIEL             ;disable low interrupt
    ;firmware initialization                                               ;Miejsce na procedury inicjalizacyjne
        ;
    ;firmware started
        banksel FIRMFLAG                                                    ;Flaga sygnalizująca zakończenie inicjalizacji programu funkcyjnego
        bsf     FIRMFLAG,0              ;set flag "firmware started and ready for interrupts"
    ;enable global interrupt                                               ;Ponowne włączenie przerwań
        bsf     INTCON,GIEH             ;enable high interrupt
        bsf     INTCON,GIEL             ;enable low interrupt
Loop:                                                                       ;Pętla główna programu
        clrwdt                                                              ;Zerowanie timera Watchdog-a
        bra      Loop

;=========================================================================
;===  FIRMWARE ROUTINES  =================================================
;=========================================================================
CANInterrupt                                                                ; Procedura zapisu do bufora FIFO odebranej wiadomości CAN
        ;put your code here
    return
UartInterrupt                                                               ; Procedura zapisu do bufora FIFO odebranej wiadomości UART
        ;put your code here
    return
;=========================================================================
;===  END  OF  MAIN PROGRAM  =============================================
;=========================================================================
    END
```

7.7. Nieprawidłowy firmware

Modyfikacje oprogramowania firmware mogą spowodować niewłaściwe funkcjonowanie procesora. Na przykład ingerencja firmware funkcyjnego w zastrzeżony dla bootloader-a obszar pamięci może spowodować, że dostęp do bootloadera stanie się niemożliwy. W takim wypadku należy odłączyć zasilanie procesora UNIV 3 CPU. Po ponownym podłączeniu zasilania bootloader wystartuje prawidłowo (co umożliwi z nim komunikację) i dopiero po 3 sekundach przystąpi do uruchomienia firmware. W ciągu 3 sekund od podłączenia zasilania procesora należy wprowadzić bootloader w tryb programowania (wysyłając wiadomość 0x100) i następnie skasować nieprawidłowy firmware.

8. Zmiany w wersjach bootloader-a

Wersja bootloader-a	Zmiany	Pełna kompatybilność z wersją wcześniejszą
3.0	Wersja oryginalna.	-
3.1	Usunięto problem z odbiorem UART przy prędkościach innych niż 115200bps	-
3.2	Poprawiono zarządzanie energią.	√
3.3	Dodano tryb błędu bootloader-a	√
3.4	Zmiany w obsłudze przerwania CAN	√

9. Wersja dokumentu

Plik	Opis	Data
boot_3-4a_pl.pdf	Wersja oryginalna	Marzec 2013
boot_3-4b_pl.pdf	Dodano kilka rysunków, zaktualizowano wzór kodu oprogramowania i poprawiono opis ramki 0x105 i 0x106	Wrzesień 2013