mgr Marian MENDEL
Wojskowy Instytut Techniczny Uzbrojenia

REJESTRY MIKROKONTROLERA DO WSPÓŁPRACY Z MAGISTRALĄ CAN (CONTROLLER AREA NETWORK)

Streszczenie: W artykule opisano struktury rejestrów mikrokontrolera używane do sterowania magistralą CAN (Controller Area Network). Zamieszczono przykłady użycia tych rejestrów dla wybranego wariantu pracy CAN. Informacje zawarte w artykule wystarczają do napisania aplikacji dla mikroprocesora obsługującej CAN w zakresie konfiguracji i transmisji danych.

Słowa kluczowe: CAN, mikrokontroler, mikroprocesor, magistrala, rejestr, transmisja.

REGISTERS OF MICROCONTROLLER USED FOR INTERACTION WITH CAN (CONTROLLER AREA NETWORK) BUS

Abstract: In the paper the structures of microcontroller registers used for control the CAN (Control Area Network) bus were presented. The examples of usage of the registers for selected CAN operating mode were shown. The information presented in the paper are sufficient to write an application for microprocessor that provides the configuration and the transmission of data for CAN.

Keywords: CAN, microcontroller, microprocessor, bus connection, register, transmission.

1. Wstęp

W artykule [1] przedstawiono ogólne zasady komunikacji mikrokontrolerów poprzez magistralę CAN, w tym wyznaczanie parametrów związanych z prędkością transmisji. Niniejszy artykuł jest kontynuacją ww. ze szczególnym uwzględnieniem rejestrów mikrokontrolera związanych z przesyłaniem danych poprzez CAN. Nazwy rejestrów i pól zostały przyjęte tak, jak w karcie katalogowej mikrokontrolera PIC18F2480/2580/4480/4580. Przykłady zostały opracowane dla mikrokontrolera PIC18F2580. Opis rejestrów i ich pól dotyczył będzie tylko udoskonalonego podstawowego wariantu transmisji danych (wariant 1), a pola związane z innymi wariantami pracy nie zostaną opisane (pozostaną puste).

2. Rejestry związane z konfiguracją CAN

Do opisu zawartości rejestrów użyto poniższego schematu, w którym:

1) M - oznacza uprawnienie do modyfikacji bitu (gdy uprawnień jest więcej, to oddzielone są znakiem „/”) i przyjmuje znaczenia:

- R - dozwolony odczyt bitu,
- W - dozwolony zapis (zmiana) bitu,
- S - zmiana bitu dostępna tylko sprzętowo dla mikrokontrolera,
- U - bit niezaimplementowany (nie używany),

2) D - oznacza wartość bitu przyjmowaną po starcie mikrokontrolera (default) i przyjmuje wartości:

- 1 - bit ustawiony,
- 0 - bit zgaszony,
- x - przypadkowa wartość bitu,

3) Pole - oznacza nazwę pola rejestru przyjętą w karcie katalogowej mikrokontrolera.

M/M-D	M/M-D	M/M-D	M/M-D	M/M-D	M/M-D	M/M-D	M/M-D
Pole	Pole	Pole	Pole	Pole	Pole	Pole	Pole
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Konfiguracja pracy CAN może być zmieniana poprzez zmianę zawartości odpowiednich rejestrów z zachowaniem warunków, w których ta zmiana może nastąpić. Zawsze taką konfigurację wykonuje się przed włączeniem CAN i w nielicznych przypadkach w czasie pracy CAN. Niektóre pola rejestrów konfiguracyjnych mogą być zmienione tylko wtedy, gdy CAN jest przełączony w tryb konfiguracji. Konfiguracja obejmuje ustalenie: wariantu pracy CAN, prędkości transmisji i zdefiniowanie filtrów i masek identyfikatorów akceptowanych wiadomości.

2.1. Konfiguracja wariantu pracy CAN

CAN może pracować w jednym z trzech wariantów, które różnią się między sobą liczbą dostępnych rejestrów odbiorczych i nadawczych, sposobem użycia rejestrów komunikacyjnych, liczbą dostępnych filtrów i kolejkowaniem wiadomości (p. [2]. str. 331).
Zmianę wariantu można zrealizować poprzez zmianę wartości rejestru ECANCON (ENHANCED CAN CONTROL REGISTER) ale tylko wtedy, gdy CAN jest w trybie konfiguracji. Tryby pracy CAN ustawia się poprzez zmianę rejestru CANCON (CAN CONTROL REGISTER).

Rejestr CANCON:

R/W-1	R/W-0	R/W-0					
REQOP2	REQOP1	REQOP0					
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie: CANCON.REQOP (bit 7-5) wymagany tryb pracy CAN:

- CANCON.REQOP=0b1xx - żądanie przełączenia w tryb konfiguracji,
- CANCON.REQOP=0b011 - żądanie przełączenia w tryb nasłuchu bez nadawania,
- CANCON.REQOP=0b010 - żądanie przełączenia w tryb odbierania i przekazywania do mikrokontrolera nadawanych przez siebie wiadomości,
- CANCON.REQOP=0b001 - żądanie wyłączenia CAN,
- CANCON.REQOP=0b000 - żądanie przełączenia w normalny tryb pracy,

Zmiana CANCON.REQOP powoduje zapisanie odpowiedniego żądania. Wykonanie zapisanego żądania zostaje zapisane w rejestrze CANSTAT (CAN STATUS REGISTER) i dopiero wtedy CAN jest w określonym trybie pracy.

Rejestr CANSTAT:

R-1	R-0	R-0					
OPMODE2	OPMODE1	OPMODE0					
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie: CANSTAT.OPMODE (bit 7-5) aktualny tryb pracy CAN:

- CANSTAT.OPMODE=0b100 - tryb konfiguracji,

- CANSTAT.OPMODE=0b011 - tryb nasłuchu bez nadawania,
- CANSTAT.OPMODE=0b010 - tryb odbierania i przekazywania do mikrokontrolera nadawanych przez siebie wiadomości,
- CANSTAT.OPMODE=0b001 - CAN wyłączony lub w uśpieniu,
- CANSTAT.OPMODE=0b000 - normalny tryb pracy.

Rejestr ECANCON:

R/W-0	R/W-0						
MDSEL1	MDSEL0						
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie: ECANCON.MDSEL (bit 7-6) wybór wariantu pracy CAN:
- ECANCON.MDSEL=0b00 - podstawowy wariant transmisji danych (wariant 0),
- ECANCON.MDSEL=0b01 - udoskonalony podstawowy wariant transmisji danych (wariant 1),
- ECANCON.MDSEL=0b10 - udoskonalony wariant transmisji danych typu FIFO (wariant 2),

Przykład 1

```
CANCON.REQOP = 0b100;                    //żądanie przełączenia w tryb konfiguracji,
while (CANSTAT.OPMODE != 0b100);         //czekamy na przełączenie CAN w tryb konfiguracji,
ECANCON.MDSEL = 0b01;                    //ustawienie wariantu 1 transmisji danych
CANCON.REQOP = 0b000;                    //żądanie przełączenia w normalny tryb pracy CAN,
while (CANSTAT.OPMODE);                  //czekamy na przełączenie CAN w tryb normalnej pracy
```

2.2. Konfiguracja prędkości transmisji

Do ustawienia prędkości transmisji CAN przeznaczone są rejestry: BRGCON1 (BAUD RATE CONTROL REGISTER 1), BRGCON2 (BAUD RATE CONTROL REGISTER 2) i BRGCON3 (BAUD RATE CONTROL REGISTER 3). Zmiana prędkości transmisji CAN może być dokonana tylko w trybie konfiguracji CAN.

Rejestr BRGCON1:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie:
- BRGCON1.SJW (bit 7-6) - szerokość przedziału dosynchronizowania bitu SJW (Synchronized Jump Width) przyjmuje wartości od 0 do 3, co implikuje SJW = (BRGCON1.SJW + 1) * t_Q, [1],
- BRGCON1.BRP (bit 5-0) - wartość współczynnika skalującego częstotliwość oscylatora PRESC [1] i przyjmuje wartości całkowite z przedziału domkniętego [0,63], co implikuje PRESC = BRGCON1.BRP + 1.

Rejestr BRGCON2:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEG2PHTS	SAM	SEG1PH2	SEG1PH1	SEG1PH0	PRSEG2	PRSEG1	PRSEG0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie:

- BRGCON2.SEG2PHTS (bit 7) - sposób wyznaczania czasu trwania fazy SEG2 [1]:
- BRGCON2.SEG2PHTS=1 - z oprogramowania mikrokontrolera,
- BRGCON2.SEG2PHTS=0 - SEG2=Max(IPT,SEG1), gdzie: IPT - Information Processin Time,
- IPT = 2* t_Q lub IPT = 3* t_Q, gdy PRESC = 1 lub gdy zastosowano potrójne próbkowanie odebranego bitu: BRGCON2.SAM=1,
- BRGCON2.SAM (bit 6) - sposób próbkowania odebranego bitu:
- BRGCON2.SAM=1 - potrójne próbkowanie odebranego bitu,
- BRGCON2.SAM=0 - pojedyncze próbkowanie odebranego bitu,
- BRGCON2.SEG1PH (bit 5-3) - czas trwania fazy SEG1 [1], przyjmuje wartości z przedziału domkniętego [0,7], co implikuje
SEG1 = (BRGCON2.SEG1PH+1)*t_Q,
- BRGCON2.PRSEG (bit 2-0) - czas trwania propagacji bitu PROP [1], przyjmuje wartości z przedziału domkniętego [0,7], co implikuje
PROP = (BRGCON2.PRSEG + 1) * t_Q.

Rejestr BRGCON3:

R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
WAKDIS	WAKFIL	-	-	-	SEG2PH2	SEG2PH1	SEG2PH0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie:
- BRGCON3.WAKDIS (bit 7) - możliwość budzenia CAN ze stanu uśpienia:
- BRGCON3.WAKDIS =1 - wyłączona możliwość budzenia CAN ze stanu uśpienia,
- BRGCON3.WAKDIS =0 - włączona możliwość budzenia CAN ze stanu uśpienia,
- BRGCON3.WAKFIL (bit 6) - użycie linii do budzenia CAN:
- BRGCON3.WAKFIL =1 - użyta,
- BRGCON3.WAKFIL =0 - nie użyta,
- BRGCON3.SEG2PH (bit 2-0) - czas trwania fazy SEG2 [1] (ma znaczenie tylko wtedy, gdy BRGCON2.SEG2PHTS=1), przyjmuje wartości z przedziału domkniętego [0,7], co implikuje
SEG2 = (BRGCON3.SEG2PH+1)*t_Q,

Przykład 2

```
CANCON.REQOP = 0b100;     //żądanie przełączenia w tryb konfiguracji,
while (CANSTAT.OPMODE != 0b100);          //czekamy na przełączenie CAN w tryb konfiguracji,
ECANCON.MDSEL = 1;        //ustawienie wariantu 1 transmisji danych
BRGCON1.SJW = 0;   //SJW=1
BRGCON1.BRP = 1;   //PRESC = 2
BRGCON2.SEG2PHTS=1;       /*długość SEG2 wyznaczona z oprogramowania mikrokontrolera,*/
BRGCON2.SAM=0;    //pojedyncze próbkowanie odebranego bitu,
BRGCON2.SEG1PH = 1;       //SEG1=2*tQ,
BRGCON2.PRSEG = 2;        //PROP=3*tQ,
BRGCON3.WAKDIS =1         //wyłączona możliwość budzenia CAN ze stanu uśpienia,
BRGCON3.WAKFIL =0;        //nie użyta linia do budzenia CAN
BRGCON3.SEG2PH = 1;       //SEG2=2*tQ,
CANCON.REQOP = 0;//żądanie przełączenia w normalny tryb pracy CAN,
while (CANSTAT.OPMODE); //czekamy na przełączenie CAN w tryb normalnej pracy.
```

2.3. Definicje filtrów i masek wiadomości

W celu zarządzania odbieranymi wiadomościami przez CAN pod kątem przekazywania ich do mikrokontrolera należy zdefiniować odpowiednie filtry i skorelowane z nimi maski,

których działanie opisano w [1]. Przykładowo w mikrokontrolerze PIC18F2580 mamy do dyspozycji 16 filtrów i 2 maski. W maskach zapisujemy identyfikatory wiadomości, które mają być przyjmowane. Maski przeznaczone są do zasłonięcia wybranych bitów identyfikatora w celu przyjęcia wiadomości mimo to, że bit ten jest niezgodny z odpowiednim bitem filtra. Zasadę akceptacji bitu identyfikatora wiadomości lub braku akceptacji tego bitu, co implikuje odrzucenie odebranej wiadomości, przedstawia poniższa tabela.

Maska bit n	Filtr bit n	Ideat. wiad. bit n	Akceptacja
0	x	x	Tak
1	0	0	Tak
1	0	1	Nie
1	1	0	Nie
1	1	1	Tak

x - dowolna wartość bitu

W rejestrach RXFCON0 i RXFCON1 zaznaczamy, które filtry są zdefiniowane w celu użycia przez układ CAN.

Rejestr RXFCON0:

R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RXF7EN	RXF6EN	RXF5EN	RXF4EN	RXF3EN	RXF2EN	RXF1EN	RXF0EN
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Rejestr RXFCON1:

R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
RXF15EN	RXF14EN	RXF13EN	RXF12EN	RXF11EN	RXF10EN	RXF9EN	RXF8EN
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie:
- pole RXFnEN = 1 oznacza, że n-ty filtr jest zdefiniowany (dla $0 \leq n \leq 15$), a w przeciwnym wypadku n-tego filtru brak.

Do zapisu filtrów wiadomości przeznaczone są rejestry RXFnSIDH, RXFnSIDL, RXFnEIDH i RXFnEIDL, gdzie $0 \leq n \leq 15$.

Rejestr RXFnSIDH: 8 najstarszych bitów filtru n ($0 \leq n \leq 15$)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Rejestr RXFnSIDL: część filtru n ($0 \leq n \leq 15$)

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	-	EXIDEN	-	EID17	ESID16
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie:
- RXFnSIDL.EXIDEN - typ identyfikatora wiadomości:
- RXFnSIDL.EXIDEN=0 - identyfikator standardowy (11 bitowy) określony przez bity SID10:SID0,
- RXFnSIDL.EXIDEN=1-identyfikator rozszerzony (29 bitowy), gdzie bity SID10:SID0 są najstarszymi bitami identyfikatora.

Rejestr RXFnEIDH: starsza część filtru n (0 ≤ n≤ 15) w trybie rozszerzonym

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Rejestr RXFnEIDL: najmłodsza część filtru n (0≤ n≤ 15) w trybie rozszerzonym

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie bity EID17:EID0 stanowią najmłodsze bity identyfikatora wiadomości w trybie rozszerzonym, a bity SID10:SID0 są najstarszymi bitami tego identyfikatora.
Do zapisu masek wiadomości przeznaczone są rejestry RXMnSIDH, RXMnSIDL, RXMnEIDH i RXMnEIDL, gdzie 0≤ n ≤ 1. Struktury tych rejestrów są identyczne ze strukturami odpowiednich rejestrów RXFnSIDH, RXFnSIDL, RXFnEIDH i RXFnEIDL. Kojarzenie filtrów z maskami wykonuje się poprzez odpowiednie wpisy do rejestrów MSEL0, MSEL1, MSEL2 i MSEL3.

Rejestr MSEL0: kojarzenie filtrów 0, 1, 2 i 3 z maskami

R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
FIL3_1	FIL3_0	FIL2_1	FIL2_0	FIL1_1	FIL1_0	FIL0_1	FIL0_0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie:
pola FILn_ zawierają numer maski skojarzonej z filtrem n wg zasad:
- FILn_=0 - z filtrem n skojarzona jest maska 0,
- FILn_=1 - z filtrem n skojarzona jest maska 1,
- FILn_=2 - z filtrem n skojarzony jest filtr 15 jako maska,
- FILn_=3 - z filtrem n nie skojarzona jest żadna maska.

Identyczne znaczenie mają rejestry:
- MSEL1 - kojarzenie filtrów 4, 5, 6 i 7 z maskami,
- MSEL2 - kojarzenie filtrów 8, 9, 10 i 11 z maskami,
- MSEL3 - kojarzenie filtrów 12, 13, 14 i 15 z maskami.

Przykład 2

```
RXFCON0 = 0b00000011;       //Aktywne są tylko filtry 0 i 1
RXFCON1 = 0;
RXF0EIDL=0x7B;      // Filtr 0: przyjmować wiadomości o identyfikatorze 0x7B
RXF0EIDH=0; //starsze bity filtru 0
RXF0SIDL=0;  //starsze bity filtru 0
RXF0SIDH=0; //starsze bity filtru 0
RXF0SIDL.EXIDEN=1;          //Identyfikator rozszerzony
RXF1EIDL=0xA0;      //Filtr 1: przyjmować wiadomości o identyfikatorze 0xA0
RXF1EIDH=0; //starsze bity filtru 1
RXF1SIDL=0;  //starsze bity filtru 1
RXF1SIDH=0; //starsze bity filtru 1
RXF1SIDL.EXIDEN=1;          //Identyfikator rozszerzony
RXM0EIDL=0xF0;      /*Maska 0: przyjmować wszystkie wiadomości niezależnie od 4 najmłodszych
bitów w identyfikatorze. Pozostałe bity identyfikatora muszą być zgodne z odpowiednim filtrem*/
RXM0EIDH=0xFF;      //starsze bity maski 0, wymagana zgodność bitów z filtrem
RXM0SIDL=0xFF;      //starsze bity maski 0, wymagana zgodność bitów z filtrem
RXM0SIDH=0xFF;      //starsze bity maski 0, wymagana zgodność bitów z filtrem
RXM0SIDL.EXIDEN=1;      //Identyfikator rozszerzony
```

MSEL0=0b00000011; /*Dla filtra 0 brak maski, a dla filtra 1 skojarzono maskę 0; przyjmowane będą wiadomości o identyfikatorze 0x7B i identyfikatorach 0xA0 ÷ 0xAF*/.

2.4. Podział buforów programowalnych

W CAN standardowo wydzielono 2 bufory odbiorcze (RXB0, RXB1) i 3 bufory nadawcze (TXB0, TXB1, TXB2). Ponadto do dyspozycji użytkownika oddano 6 buforów programowalnych (B0÷B5), które można użyć wedle uznania jako nadawcze lub odbiorcze. Sposób użycia tych buforów definiuje się w rejestrze BSEL0.

Rejestr BSEL0: określenie typów buforów Bn (dla 0≤ n ≤5)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
B5TXEN	B4TXEN	B3TXEN	B2TXEN	B1TXEN	B0TXEN	-	-
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie pola BnTXEN oznaczają typ buforu Bn następująco:
- BnTXEN = 0 - bufor Bn jest buforem odbiorczym,
- BnTXEN = 1 - bufor Bn jest buforem nadawczym.

3. Rejestry sterujące transmisją CAN

Komunikacja pomiędzy mikrokontrolerem i CAN może odbywać się w przerwaniach od CAN lub poprzez podgląd rejestrów związanych z CAN. Sposób realizacji poszczególnych funkcji określają rejestry TXBIE, BIE0, PIE3 i IPR3, a flagi zdarzeń znajdują się w rejestrze PIR3.

Rejestr PIE3: flagi włączające/wyłączające przerwania od CAN

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIE	WAKIE	ERRIE	TXBnIE	TXB1IE	TXB0IE	RXBnIE	FIFOWMIE
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie wartość 1 flagi oznacza włączenie zgłaszania przerwania, a wartość 0 wyłączenie:
- IRXIE - błąd odbioru wiadomości,
- WAKIE - budzenie ze stanu czuwania,
- ERRIE - błąd na CAN,
- TXBnIE - obsługa flag zawartych w rejestrach TXBIE i BIE0 dotyczących nadania wiadomości,
- TXB1IE=TXB0IE=0 - flagi te występują tylko w wariancie 0 pracy CAN,
- RXBnIE - obsługa flag zawartych w rejestrze BIE0 dotyczących odbioru wiadomości,
- FIFOWMIE - obsługa kolejki FIFO odebranych wiadomości (działa w wariancie 2 pracy CAN).

Rejestr TXBIE: flagi włączające/wyłączające przerwania od CAN wywołane odebraniem wiadomości

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	U-0	U-0
-	-	-	TXB2IE	TXB1IE	TXB0IE	-	-
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie wartość 1 flagi oznacza włączenie zgłaszania przerwania, a wartość 0 wyłączenie:
TXBnIE - odebrana wiadomość w buforze TXBn (0 ≤ n≤ 2).

Rejestr BIE0: flagi włączające/wyłączające przerwania od CAN wywołane odebraniem/nadaniem wiadomości

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
B5IE	B4IE	B3IE	B2IE	B1IE	B0IE	RXB1IE	RXB0IE
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie wartość 1 flagi oznacza włączenie zgłaszania przerwania, a wartość 0 wyłączenie:
- BnIE - odebrana/wysłana wiadomość w buforze Bn ($0 \leq n \leq 5$),
- RXBnIE - odebrana wiadomość w buforze RXBn ($0 \leq n \leq 1$).

Rejestr IPR3: flagi priorytetów przerwań ustawionych w rejestrze PIE3

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIP	WAKIP	ERRIP	TXBnIP	TXB1IP	TXB0IP	RXBnIP	FIFOWMIP
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie wartość 1 flagi oznacza wysoki priorytet przerwania, a wartość 0 niski:
- IRXIP - priorytet przerwania IRXIE,
- WAKIP - priorytet przerwania WAKIE,
- ERRIP - priorytet przerwania ERRIE,
- TXBnIP - priorytet przerwania TXBnIE,
- TXB1IP=TXB0IP=0 - flagi te występują tylko w wariancie 0 pracy CAN,
- RXBnIP - priorytet przerwania RXBnIE,
- FIFOWMIP - priorytet przerwania FIFOWMIE.

Rejestr PIR3: flagi zdarzeń CAN

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIF	WAKIF	ERRIF	TXBnIF	TXB1IF	TXB0IF	RXBnIF	FIFOWMIF
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie wartość 1 flagi oznacza wystąpienie zdarzenia, a wartość 0 brak tego zdarzenia:
- IRXIF - błąd odbioru wiadomości,
- WAKIF - budzenie ze stanu czuwania,
- ERRIF - błąd na CAN,
- TXBnIF - zdarzenie związane z rejestrami TXB0, TXB1, TXB3 lub B0÷B5 dotyczące nadania wiadomości,
- TXB1IF=TXB0IF=0 - flagi te występują tylko w wariancie 0 pracy CAN,
- RXBnIF - zdarzenie związane z rejestrami B0÷B5 dotyczące odbioru wiadomości,
- FIFOWMIF - w kolejce FIFO nowa wiadomość (działa w wariancie 2 pracy CAN).

Uwaga.
Po obsłudze zdarzenia związanego z przypisaną mu flagą, flagę tę należy obowiązkowo programowo zgasić.

3.1. Struktura buforów nadawczych i odbiorczych

Struktury buforów nadawczego i odbiorczego są identyczne. W ich skład wchodzą następujące rejestry:
- konfiguracyjny CON,
- identyfikatora wiadomości: SIDH, SIDL, EIDH, EIDL,
- liczby bajtów danych DLC,
- danych: D0, D1,…,D7.

Struktura rejestru konfiguracyjnego CON dla buforu nadawczego jest inna niż dla buforu odbiorczego.

Rejestr konfiguracyjny CON dla buforu odbiorczego

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
RXFUL	RXM1	RTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie pola oznaczają:
- RXFUL - odebrana wiadomość: 1- jest; 0 - brak (bit jest gaszony, gdy,
- RXM1 - typ odbieranych wiadomości: 1 - wszystkie, włącznie z błędnymi; 0 - o identyfikatorach zgodnych z filtrami,
- FILHIT - nr filtra, z którym jest zgodny identyfikator odebranej wiadomości.

Rejestr konfiguracyjny CON dla buforu nadawczego

R/W-0	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
TXBIF	TXABT	TXLARB	TXERR	TXREQ	RTREN	TXPRI1	TXPRI0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

gdzie pola oznaczają:
- TXBIF - stan nadawania wiadomości: 1- została nadana poprawnie; 0 - w trakcie nadawania,
- TXABT - przerwano transmisję wiadomości: 1 - tak, 0 - nie,
- TXLARB - utracono wiadomość w procesie arbitrażu na magistrali CAN: 1 - tak, 0 - nie,
- TXERR - wykryto błąd nadawania na magistrali CAN: 1 - tak, 0 - nie,
- TXREQ - żądanie nadania:
  1 - tak (automatycznie wykonywane są:TXABT=0, TXLARB=0 i TXERR=0),
  0 - nie (automatycznie zerowane po nadaniu),
- RTREN - wiadomość zawiera zdalne żądanie automatycznej odpowiedzi przez stację odbierającą: 1 - tak, 0 - nie,
- TXPRI - priorytet nadania wiadomości (pobrania z buforów nadawczych): od 0 - najniższego do 3 - najwyższego.

3.2. Nadawanie

Aby nadać wiadomość należy wpisać do rejestrów bufora nadawczego wymagane wartości. Oprócz standardowych buforów nadawczych TXB0, TXB1 i TXB2 dostępne są bufory programowalne B0÷B5, które w zależności od potrzeb mogą być użyte jako nadawcze lub odbiorcze. Poniższy przykład ilustruje wysłanie wiadomości.

Przykład 3

```
BSEL0 = 0;      //Wszystkie bufory programowalne B0÷B5 będą użyte jako odbiorcze
//Poszukiwanie pustego buforu nadania
if (TXB0CON.TXREQ == 0)    //Czy dostępny jest bufor TXB0?
{
        ECANCON.EWIN=3;  //Grupa rejestrów specjalnych przeznaczonych dla buforu TXB0
                //(wybrana z listy dla struktury rejestru ECANCON)
        //11-bitowy (standardowy) identyfikator wiadomości = 7 = 0b111
        //Wpisanie danych do buforu TXB0
        TXB0SIDH=0; //Starsze 8 bitów identyfikatora
        TXB0SIDL=0b11100000;                //Młodsze 3 bity identyfikatora (0b111) i EXIDEN=0
        TXB0DLC=8;  //Liczba bajtów danych
```

```
        TXB0D0=1;…; TXB0D7=8;              //Bajty danych o wartościach: 1, 2, 3, 4, 5, 6, 7, 8
        TXB0CON.RTREN=0; //Brak zdalnego żądania wiadomości
        TXB0CON.PRI=0;      //Najniższy priorytet pobrania przez CAN wiadomości do nadania
        TXB0CON.TXREQ=1; //Nadać wiadomość (GOTOWE!)
}
else if (TXB1CON.TXREQ == 0)        //Czy dostępny jest bufor TXB1?
{
        ECANCON.EWIN=4;  //Grupa rejestrów specjalnych przeznaczonych dla buforu TXB1
        //Wpisanie danych do buforu TXB1
}
else if (TXB2CON.TXREQ == 0)        //Czy dostępny jest bufor TXB2?
{
        ECANCON.EWIN=5;  //Grupa rejestrów specjalnych przeznaczonych dla buforu TXB2
        // Wpisanie danych do buforu TXB2
}
```

3.3. Odbiór

Standardowo do odbioru wiadomości przeznaczone są bufory RXB0 i RXB1. Ponadto można wykorzystać bufory programowalne B0÷B5, które w zależności od potrzeb mogą być użyte jako nadawcze lub odbiorcze. Proces odbioru może być realizowany poprzez podgląd buforów odbiorczych lub metodą obsługi przerwań od buforów odbiorczych. Poniższy przykład ilustruje odebranie wiadomości.

Przykład 4

```
BSEL0 = 0b11111100;  //Wszystkie bufory programowalne B0÷B5 będą użyte jako nadawcze
unsigned int8 i,Dane[8],DLC,SIDH,SIDL,EIDH,EIDL,*DANE_RX;
Identyfikator;
#if defined (PRZERWANIA_BUFOROW_ODBIORU_CAN)
ECANCON.EWIN=CANSTAT.EICODE;        /* Grupa rejestrów specjalnych dla buforu
odbiorczego zgłaszającego przerwanie. Ta instrukcja ma sens i jest obowiązkowa tylko w obsłudze
przerwania*/
#endif
if (RXB0CON.RXFUL)// Czy jest wiadomość w buforze RXB0
{
//Odczyt z buforu odbiorczego RBX0
        #ifndef PRZERWANIA_BUFOROW_ODBIORU_CAN
        ECANCON.EWIN=0b10000;   //Grupa rejestrów specjalnych dla buforu odbiorczego RXB0
        #endif
        SIDH=RXB0SIDH;
        SIDL= RXB0SIDL;
        if (RXB0SIDL. EXIDEN)
        {
                EIDH= RXB0EIDH;
                EIDL= RXB0EIDL;
        }
        DLC=RXB0DLC;
        DANE_RX=&RXB0D0;
        for (i=0; i<DLC;i++)
        {
                Dane[i]=* DANE_RX;
```

```
                DANE_RX++;
        }
        RXB0CON.RXFUL=0;
}
else if (RXB1CON.RXFUL)// Czy jest wiadomość w buforze RXB1
{
//Odczyt z buforu odbiorczego RBX1 analogicznie jak dla RBX0
        #ifndef PRZERWANIA_BUFOROW_ODBIORU_CAN
        ECANCON.EWIN=0b10001;   //Grupa rejestrów specjalnych dla buforu odbiorczego RXB0
        #endif
}
#if defined (PRZERWANIA_BUFOROW_ODBIORU_CAN)
PIR3.RXBnIF=0;
#endif
```

4. Podsumowanie

Treść niniejszego artykułu w połączeniu z [1] pozwala na napisanie prostej aplikacji dla mikrokontrolera PIC18F2580 korzystającej z CAN w celu transmisji danych. Inne mikrokontrolery mogą mieć inną strukturę rejestrów związanych z CAN ale ich pola są analogiczne do tych, które zostały opisane w tym artykule. Aby w pełni wykorzystać możliwości magistrali CAN należy dokładnie zapoznać się z kartą katalogową określonego mikrokontrolera.

Literatura

[1] Marian Mendel, Magistrala CAN (CONTROLLER AREA NETWORK). Konfiguracja i transmisja danych, *Problemy Techniki Uzbrojenia*, nr 2, str. 49, 2013r.

[2] *PIC18F2480/2580/4480/4580 sheet*, Microchip Technology Inc., 2009r.