mgr Marian MENDEL
Wojskowy Instytut Techniczny Uzbrojenia

MAGISTRALA CAN (CONTROLLER AREA NETWORK) KONFIGURACJA I TRANSMISJA DANYCH

Streszczenie:W artykule przedstawiono ogólne zasady komunikacji mikrokontrolerów poprzez do magistralę CAN (Controller Area Network) ze szczególnym uwzględnieniem struktury ramki danych, identyfikacji wiadomości, arbitrażu dostępu do magistrali, unikania konfliktów, wykrywania i korekcji błędów występujących na magistrali i w stacjach uczestniczących w transmisji danych, faz przesyłania pojedynczego bitu i filtracji wiadomości u odbiorcy. Został zamieszczony algorytm określania parametrów związanych konfiguracją CAN dla założonej prędkości danych i długości magistrali.

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

CAN BUS (CONTROLLER AREA NETWORK) CONFIGURATION AND DATA TRANSMISSION

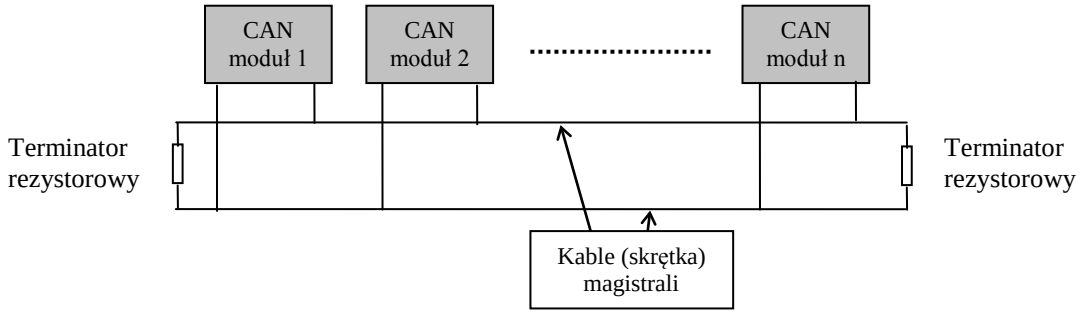
Abstract: In the paper the general rules of communication of microcontrollers through CAN bus (Controller Area Network) were presented. The particular stress was laid on the structure of data frame, the identification of messages, the arbitrage in the access to the bus, the avoiding of conflicts, the detecting and correction of errors occurring on the bus and in the stations taking part in data transmission, the single bit transmission phases and the receiver message filtering. The algorithm of determination of parameters bounded with CAN configuration for the assumed data speed and the bus length was also shown.

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

1. Wstęp

W ostatnich latach bardzo dynamicznie rozwija się przemysł samochodowy pod kątem podnoszenia bezpieczeństwa i komfortu użytkownika, co wymusza stosowanie coraz większej liczby mikrokontrolerów współpracujących ze sobą. Z tym związane jest zwiększanie liczby i długości wiązek kabli. Np. w 2005r. samochód średniej klasy zawierał ok. 100 mikrokontrolerów połączonych kablami o łącznej długości ok. 2 km. Komunikacja pomiędzy mikrokontrolerami stała się zaporą w dalszym rozszerzaniu automatyzacji w kolejnych wersjach samochodów. Z identycznymi problemami stykają się producenci innego sprzętu powszechnego użytku.

Z pomocą przyszedł przemysł komputerowy, gdzie opracowano Controller Area Network (CAN). Idea CAN polega na komunikacji pomiędzy modułami CAN podłączonymi do zwykłej skrętki przewodów (Rys. 1) z prędkością do 1Mb/s (Tab. 1).



Rys. 1. Topologia magistrali CAN

Tab. 1. Współzależność pomiędzy szybkością przesyłania danych, długością magistrali, medium magistrali i impedancją zamykającą magistrali.

Długość magistrali	Kabel magistrali		Rezystancja zamykająca magistrali	Maksymalna prędkość przesyłania danych
	Rezystancja	Przekrój poprzeczny kabla		
0÷40m	70mΩ/m	0,56÷0,66mm	124Ω	1Mb/s przy 40m
40÷300m	<60mΩ/m	0,66÷0,8mm	127Ω	500kb/s przy 100m
300÷600m	<40mΩ/m	0,8÷0,87mm	150Ω÷300Ω	100kb/s przy 500m
600÷1 000m	<26mΩ/m	0,98÷1,01mm	150Ω÷300Ω	50kb/s przy 1 000m

Jak stąd widać, jedna skrętka kabli zastępuje plątaninę różnych wiązek kabli. Wygląda to pięknie ale czy równie dobrze realizowana jest wymiana danych pomiędzy mikrokontrolerami? O tym dalej.

2. Ogólne zasady wymiany danych na magistrali CAN

Układy CAN są to zintegrowane, inteligentne, peryferyjne bloczki składowe w istniejących systemach mikrokontrolerowych lub tych, które dopiero zostaną zbudowane. Protokół CAN zaszyty jest w chipie tak, że użytkownik nie musi już koncentrować się na szczegółach technologii komunikacyjnej.

CAN wykorzystuje topologię magistrali polegającą na tym, że wszystkie elementy są połączone z pojedynczą skrętką pary przewodów (ekranowaną lub nie), zakończoną na obydwu końcach odpowiednimi impedancjami zakończenia magistrali (patrz rys. 1) oraz na tym, że każda stacja może komunikować się z każdą w sieci bez żadnych ograniczeń.

Przepływ wiadomości między indywidualnymi stacjami dołączonymi do magistrali CAN jest realizowany przez nadawanie w warunkach kontrolowanej rywalizacji opatrzonych odpowiednim priorytetem wiadomości lub ramek. Dostęp do magistrali zostanie omówiony w części dotyczącej unikania konfliktów.

W specyfikacji CAN występują struktury i operacje na nich, które zapewniają sprawną transmisję wiadomości na magistrali. Poniżej przedstawię ogólny opis podstawowych pojęć używanych w specyfikacji CAN.

Dominujące i recesywne stany magistrali lub bitów

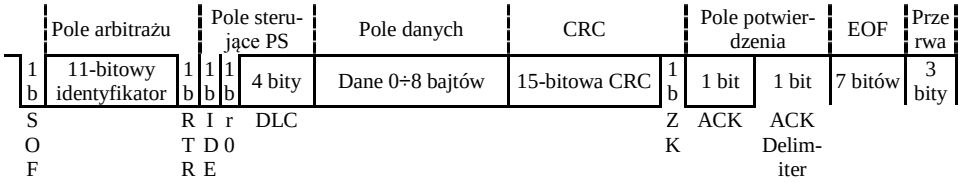
Rzeczywista transmisja danych przez medium transmisyjne danych nie odbywa się jak zwykle w postaci jedynek i zer, ale przez bity dominujące i recesywne (ustępujące, dominowane). Stan recesywny jest rodzajem stanu magistrali, który może być przykrywany (nadpisywany) przez stan dominujący magistrali. Zasada jest taka, że gdy stacja dołączona do magistrali wysyła recesywny bit, a inna stacja w tym samym czasie wysyła bit dominujący, to bit dominujący ma pierwszeństwo przed bitem recesywnym, co oznacza, że stan dominujący jest akceptowany przez całą magistralę. Przyporządkowanie stanów logicznych na magistrali jest generalnie takie, że wartość logiczna zero reprezentuje stan dominujący, a wartość logiczna 1 stan recesywny. To ustalenie tworzy podstawy specyfikacji CAN i będzie dokładniej opisane dalej.

Pakiety danych

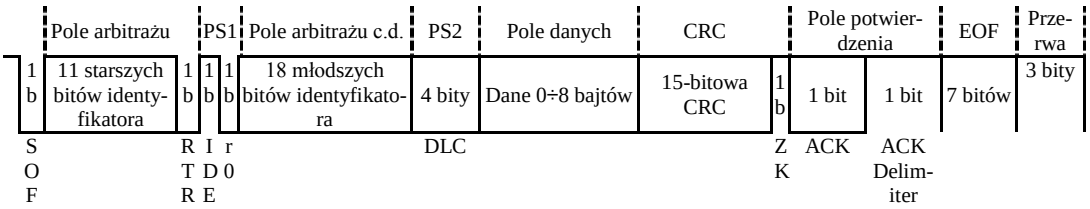
Do wymiany danych przez magistralę, w sieci używane są cztery rodzaje pakietów danych nazywanych zazwyczaj ramkami: ramka danych, ramka zdalnego wywołania, ramka sygnalizacji błędu i ramka przepełnienia.

Ramka danych

Ramka danych jest stosowana przez stacje aby zgodnie z ich oprogramowaniem przesłać dane w linię. Format typowej ramki, która składa się z pojedynczych pól pokazano na rys. 2 i rys. 3.



Rys. 2. Struktura ramki danych (format ramki standardowej – CAN 2.0A)



Rys. 3. Struktura ramki danych (format ramki rozszerzonej – CAN 2.0B)

Pola ramki danych

- **SOF**

Jest to bit startowy ramki, który jest zawsze bitem dominującym (0). Wszystkie stacje dołączone do magistrali synchronizują swoje wewnętrzne stopnie odbiorcze z narastającym zboczem tego bitu (impulsu).

- **Pole arbitrażu (decyzyjne)**

To pole, o długości 12 bitów, zawiera dane określające priorytet dostępu do magistrali.

- **Identyfikator 11-bitowy**

To pole zawiera identyfikator (ID) transmitowanych ramek. Słowo 11-bitowe umożliwia utworzenie aż do 2^{11} = 2048 różnych identyfikatorów, z których dostępnych jest tylko 2032: pozostałych 16 jest zarezerwowanych dla specjalnych funkcji. To oznacza, że pojedynczy sterownik sieci może przetworzyć 2032 różnych wiadomości (wartości zmierzone, pozycja

przełączników, funkcje sygnalizacyjne itp.). Chociaż wydaje się, że jest to duża liczba, to w wielu zastosowaniach nie jest wystarczająca. Dlatego tez został opracowany format ramki rozszerzonej EFF (ang. Extended Frame Format) z identyfikatorem 29-bitowym (CAN 2.0B). W tym standardzie może być przetworzonych 2^{29} = 536 870 912 ramek.

- **Bit zdalnego żądania transmisji RTR (ang. Remote Transmission Request)**

Ten bit, który jest zwykle dominującym (0), umożliwia stacji zaadresowanie i wysłanie wiadomości do innej określonej stacji. Jest to bardzo ważne, gdy jakieś dane są pilnie potrzebne do przetworzenia.

- **Pole sterujące**

Jest to 6-bitowe pole zawierające informację jak zbudowana jest ramka danych.

– **Bit rozszerzenia identyfikatora IDE (ang. Identifier Extension)**

Wartość tego bitu wskazuje czy jest transmitowana ramka w standardowym formacie z identyfikatorem 11-bitowym (bit IDE dominujący = 0), czy ramka w formacie rozszerzonym z identyfikatorem 29-bitowym (bit IDE recesywny = 1).

– **Bit r0 (bit rezerwowy 0)**

Ten dominujący bit został przewidziany jako zapasowy dla ewentualnego rozszerzenia specyfikacji systemu.

– **DLC (ang. Data Length Code)**

To 4-bitowe pole wskazuje ile bajtów danych jest kolejno transmitowanych w polu danych. Specyfikacja systemu CAN określa długość pola danych na 0 ÷ 8 bajtów, to oznacza, że w pojedynczej ramce danych może być transmitowanych nie więcej niż 8 bajtów danych.

- **Pole danych**

To 8-bajtowe pole zawiera bajty transmitowanych danych (0 ÷ 8).

- **Pole CRC**

Pole CRC o długości 15 bitów zawiera dodatkowe informacje wprowadzone w celu zabezpieczenia transmitowanych danych przed błędami. W tym celu stacja nadająca tworzy, zgodnie z określonymi zasadami, 15-bitową sumę kontrolną CRC na podstawie wysyłanych danych i wysyła ją razem z ramką danych. Stacja odbierająca oblicza, zgodnie z tymi samymi zasadami, na podstawie odebranych danych podobną sumę kontrolną i porównuje ją z odebraną. Jeżeli wartości tych dwóch sum są identyczne (zwykły przypadek), to transmisję danych można kontynuować. Jeżeli sumy nie są identyczne, to uruchamiana jest procedura korekcji błędu. Pole CRC jest kończone bitem ogranicznika (Znacznik Końca - ZK), który jest zwykle transmitowany w postaci recesywnej.

- **Pole potwierdzenia ACK**

To 2-bitowe pole potwierdzenia służy do wysłania potwierdzenia poprawności odebrania ramek danych.

– **Przerwa ACK**

To 1-bitowe pole jest transmitowane w postaci recesywnej i dlatego może być przykryte (nadpisane) bitem dominującym transmitowanym przez inną stację dołączoną do magistrali. Umożliwia to stacjom odbierającym wysłanie potwierdzenia odebrania poprawnej ramki danych. Bit potwierdzenia jest bitem dominującym i jest transmitowany

przez stację, zawsze po odebraniu wiadomości wolnych od błędów. Ponieważ jest to bit dominujący, to przykrywa bit recesywny wysyłany przez stację nadającą. A zatem, jeżeli stacja nadająca odbiera bit dominujący podczas okienka przerwy ACK, zamiast swojego własnego, wysłanego wcześniej bitu recesywnego, to jest informowana, że przynajmniej jedna stacja odebrała wiadomość.

– **ACK Delimiter**

Okienko przerwy ACK jest zakończone transmitowanym również recesywnie bitem ogranicznika ACK (ang. ACK Delimiter).

- **Pole zakończenia ramki EOF (ang. End of Frame)**

To pole składa się z siedmiu recesywnych bitów i kończy ramkę danych. Przed następną ramką danych, która może być transmitowana, stacje odbierające potrzebują krótkiej przerwy, która umożliwia im przetworzenie lub przynajmniej zapamiętanie odebranych danych.

- **Przerwa (ang. Intermission)**

Jest określona przez trzy recesywne bity pola przerwy kończącego ramkę danych.

Unikanie konfliktów

Ponieważ wszystkie stacje są dołączone do jednej magistrali CAN, to pojawiają się dwa problemy, które należy rozwiązać:
– Co stanie się, gdy kilka stacji zechce wysłać wiadomość w tym samym czasie?
– Jak jest podejmowana decyzja, która stacja może rozpocząć nadawanie, a która stacja musi poczekać ze swoją transmisję?

W celu ich uniknięcia konfliktów stosowana jest specjalna procedura dostępu do magistrali, która musi być przestrzegana przez wszystkie stacje, gdy chcą wysyłać wiadomości. W tej procedurze ważną rolę odgrywają właśnie bity dominujące i recesywne w polu arbitrażowym (ang. Arbitration Field). Zasadniczo każda stacja nadająca słyszy swoje własne przesłanie na magistrali: wysyła jakiś bit, odbiera go z powrotem i porównuje z wysłanym. Jeżeli te dwa bity są identyczne, to transmisja wiadomości jest dozwolona. Jeżeli jednak te dwa bity nie są jednakowe, to transmisja jest zabroniona (stacja odpada). Wspomniany wcześniej bit recesywny (o wartości 1) może być przykryty przez bit dominujący (o wartości 0). W tab. 2 pokazano rozstrzygnięcie dostępu do magistrali 3 stacji, które jednocześnie chcą nadawać. Jeżeli jakaś stacja jest w trybie nadawania, to pozostałe o tym wiedzą i są w stanie odbioru. Taki sposób rozstrzygania konfliktów wymusza żeby identyfikatory wiadomości nadawanych przez określoną stacje były różne (unikatowe na magistrali) od identyfikatorów wiadomości nadawanych przez pozostałe stacje podłączone do magistrali.

Tab. 2. Przykład rozstrzygnięcia konfliktu dostępu do magistrali CAN

Stacja	Identyfikator	Arbitraż
Nr 1	00~~1101010011~~	Odpada jako pierwsza
Nr 2	00001101001	Wygrywa
Nr 3	00001~~111001~~	Odpada jako druga

Detekcja błędów i ich korekcja

Jedną z najbardziej rzucających się w oczy cech magistrali CAN jest jej nadzwyczajna zdolność wykrywania wielu błędów podczas transmisji danych i odpowiedniego reagowania na nie. Błędy mogą występować i oczywiście występują, ale skoro są rozpoznawane, to moż-

na je skorygować. Tylko nierozpoznane błędy mogą powodować, że fałszywe dane będą przetwarzane jako poprawne.

Wykrywanie błędów transmisji

W CAN zastosowano równocześnie kilka sposobów wykrywania błędów.

- **Detekcja błędnego bitu**

Każda stacja zawsze odbiera zwrotnie swoją własną transmisję. Dlatego, jeżeli po fazie arbitrażu jest tylko jedna stacja, która wysyła wiadomość do magistrali i odbiera zwrotnie inny, różniący się od wysłanego bit (stan magistrali), to jest oczywiste, że na magistrali wystąpił błąd. W takiej sytuacji stacja przełącza się na procedurę korekcji błędu. (patrz dalej).

- **Wykrywanie błędnych bitów dodatkowych**

Specyfikacja CAN określa wyraźnie, że gdy w ramce danych jest transmitowanych kolejno więcej niż pięć bitów o tej samej wartości (na przykład siedem razy wartość zero w jakimś polu), to każda grupa pięciu bitów jest poprzedzana przez bit komplementarny (tutaj oczywiście 1). Ten wprowadzony bit, który oczywiście nie zawiera żadnej informacji, jest nazywany bitem dodatkowym. Po zakończeniu odbioru, te bity są usuwane ze strumienia danych, tak że tylko pierwotna wiadomość jest przetwarzana. Dodatkowe bity mogą być z łatwością użyte do kontroli błędów. Jeżeli odbiornik wykryje w ramce więcej niż pięć kolejnych bitów o tej samej wartości (lecz nie w polu EOF), to jest oczywiste, że to nie jest poprawny odczyt, i że wystąpił błąd podczas transmisji danych (wystąpił dodatkowy bit lub został odwrócony jeden lub więcej bitów). Wtedy odbiornik zawiesza działanie i uruchamia procedurę poprawiania błędów.

- **Detekcja błędu CRC**

Proces ten polega na oszacowaniu sumy kontrolnej CRC w odbiorniku. Gdy sumy kontrolne: odebrana i obliczona różnią się, to odbiornik zmienia swoje działanie uruchamiając procedurę korekcji błędów.

- **Detekcja błędu potwierdzenia**

W opisie formatu ramki (p. rys. 2 i rys. 3) wspomniano o bicie ACK (bit przerwy na potwierdzenie), który jest wysyłany przez stację jako bit recesywny. Wszystkie stacje, które poprawnie odebrały poprzednią ramkę nadpisują (przykrywają) ten bit bitem dominującym. Stacja nadająca wykrywa to i wie, że przynajmniej jedna stacja odebrała jej dane poprawnie. Jeżeli stacja nadająca stwierdzi, że jej bit w przerwie na potwierdzenie (ACK) nie został nadpisany, to wie, że żadna stacja nie odebrała jej wiadomości poprawnie. W takim razie, zawiesza swoje dotychczasowe działanie i wywołuje procedurę korekcji błędów.

- **Detekcja błędu formatu**

W tym procesie wykorzystywany jest fakt, że w ramce jest kilka pól, które muszą zawsze mieć ustaloną zawartość: w polu ogranicznika CRC (bit końca CRC), w polu ogranicznika potwierdzenia i w polu EOF są zawsze bity recesywne. Jeżeli w tych polach zostanie wykryty bit dominujący, to taki stan może być tylko spowodowany przez błąd transmisji danych. Wówczas stacja nadająca uruchamia także procedurę korekcji błędu.

Korekcja błędów

Procedura korekcji błędów w wypadku błędu transmisji danych jest realizowana w dwóch wariantach. Po pierwsze, ramki w których został stwierdzony jakiś błąd są natychmiast odrzucane przez odpowiednią stację i nie przetwarzane. Po drugie, jeżeli któraś ze stacji systemu wykryje jakiś błąd, to wysyła natychmiast ramkę informującą o błędzie, która składa

się z sześciu dominujących bitów (sygnalizacja błędu) i ogranicznika ramki błędu zawierającego osiem bitów recesywnych. Wskutek tego wszystkie bity recesywne na magistrali są nadpisywane, tak że występuje na niej tylko sześć dominujących bitów. Jest to jednak naruszenie przyjętej zasady, że nie więcej niż pięć kolejnych bitów może mieć tą samą wartość. Wszystkie pozostałe stacje dołączone do magistrali wykrywają ten stan i uznają dopiero co odebraną ramkę jako błędną (wadliwą), odrzucają ją i także wysyłają ramkę sygnalizującą o błędzie. Inaczej mówiąc, stacja która wykryła błąd celowo uszkadza całą transmitowaną ramkę, tak że wszystkie stacje dołączone do magistrali odbierają ją jako błędną. To oznacza, że o jakimś błędzie lokalnym w jednej stacji są natychmiast poinformowane wszystkie pozostałe stacje. Głównym założeniem sieci jest, żeby wszystkie stacje odbierały poprawne dane, które mogą być dalej przetwarzane lub żeby wszystkie stacje odbierały błędne dane, które będą odrzucane. Pierwotna stacja nadająca stwierdza oczywiście, że ramka którą wysłała jest z błędem, poprawia tą wiadomość i natychmiast wysyła ponownie.

Błąd wewnątrz stacji

Co się stanie, gdy stacja sama stwierdzi, że jest uszkodzona, wysyła dane z nieodpowiednią szybkością lub jest stacją, która odbiera tylko błędne dane? Taka stacja mogłaby stale wysyłać ramkę sygnalizująca błąd i tym samym zablokować całą sieć. CAN jest oczywiście odpowiednio zabezpieczony przed takim zdarzeniem.

3. Konfiguracja prędkości transmisji CAN

Prędkość transmisji danych bezpośrednio zależy od czasu potrzebnego na transmisję pojedynczego bitu.

Rys. 4. Fazy transmisji pojedynczego bitu

W czasie transmisji pojedynczego bitu wykonywane są określone procesy w kolejnych fazach:

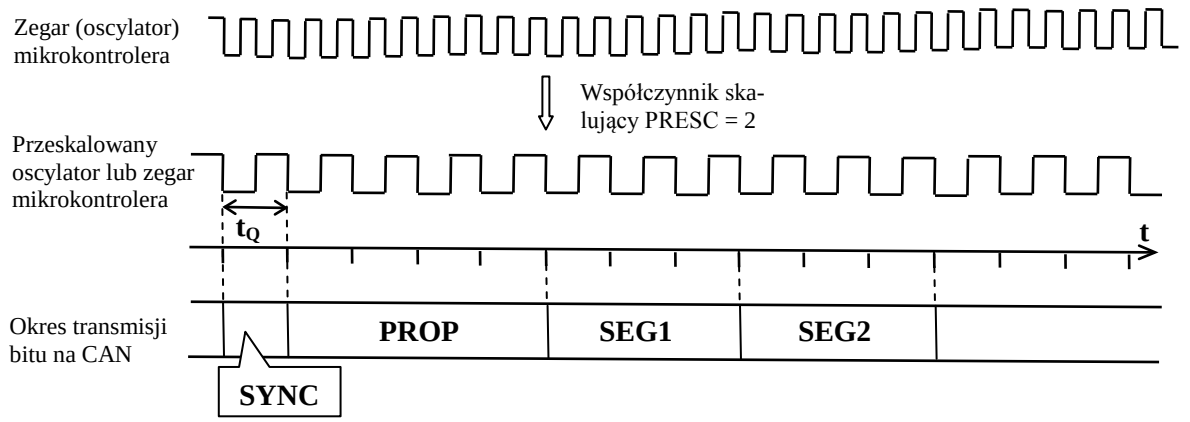
- SYNC - faza synchronizacji stacji na magistrali przed przesłaniem bitu,
- PROP - faza propagacji bitu, tzn. ustawienia stanu magistrali (dominujący lub recesywny) przez nadawcę i odczytanie tego stanu przez odbiorców,
- SEG1 i SEG2 - faza optymalizacji położenia punktu próbkowania bitu (dosynchronizowania, resynchronizacji) poprzez zmianę długości SEG1 i SEG2 tak, aby na ich granicy był optymalny punkt próbkowania bitu.

Czas potrzebny na transmisję bitu t_{BIT} jest sumą czasów realizacji poszczególnych faz:

$$t_{BIT} = t_{SYNC} + t_{PROP} + t_{SEG1} + t_{SEG2}$$

Dla określonych mikrokontrolera, oscylatora i żądanej prędkości transmisji czasy te należy oszacować i zapisać w odpowiednich rejestrach. Ponieważ w rejestrach na zapis każdego czasu przeznaczono 3 bity, konieczne jest zastosowanie współczynnika skalującego PRESC w

celu wyznaczenia nowej jednostki czasu t_Q tzw. kwant czasu, który jest PRESC razy dłuższy od cyklu oscylatora (rys. 5).



Rys. 5. Zależność pomiędzy zegarem (oscylatorem) mikrokontrolera, kwantem czasu t_Q i fazami transmisji pojedynczego bitu

ALGORYTM WYZNACZANIA PARAMETRÓW CZASOWYCH DLA TRANSMISJI POJEDYNCZEGO BITU NA MAGISTRALI CAN

KROK 1: Obliczenie czasu propagacji bitu (t_{PROP}).

t_{PROP} jest czasem zwłoki dojścia sygnału od nadajnika do odbiornika i z powrotem obliczany wg wzoru:

$$t_{PROP} = 2*(t_{Bus} * l_{Bus} + t_{Tx} + t_{Rx})[s]$$

gdzie:

t_{Bus} czas zwłoki sygnału na 1m magistrali CAN,
l_{Bus} długość magistrali CAN,
t_{Tx} czas zwłoki nadajnika Tx CAN,
t_{Rx} czas zwłoki odbiornika Rx CAN.

KROK 2: Wyznaczenie współczynnika skalującego częstotliwość oscylatora.

Dla zadanej prędkości transmisji na magistrali CAN f_{CAN} [bit/s] obliczyć czas potrzebny na transmisję pojedynczego bitu t_{BIT} wg wzoru:

$$t_{BIT} = 1/f_{CAN} [s]$$

Dla częstotliwości oscylatora f_{OSC} dobrać całkowitą liczbę PRESC = 2, 4, 6,…,128 (współczynnik skalujący) i obliczyć kwant czasu t_Q:

$$t_Q = PRESC/f_{OSC} [s]$$

tak, aby dla liczby całkowitej N_{tQ} takiej, że $7 < N_Q < 26$, spełniona była równość:

$$N_{tQ} * t_Q = t_{BIT}.[s]$$

KROK 3: Obliczenie czasu propagacji bitu PROP wyrażonego w jednostkach t_Q.

$$PROP = ROUND\_UP(t_{PROP}/t_Q),$$

gdzie: ROUND_UP zwraca najmniejszą liczbę całkowitą nie mniejszą niż argument.
Jeżeli PROP > 8, to powrócić do kroku 2 i wyznaczyć większy t_Q.

KROK 4: Wyznaczenie zwłok czasowych (etapów) SEG1 i SEG2 wyrażonych w t_Q.

Obliczyć:

$$R = N_{tQ} - PROP - SYNC,$$

gdzie: SYNC=t_Q jest zwłoką na synchronizację bitu.
Jeżeli R < 3, to powrócić do kroku 2 i wyznaczyć mniejszy t_Q.
Jeżeli R = 3, to SEG1 = 1, SEG2 = 2 i ustawić pojedyncze próbkowanie bitu.
Jeżeli R > 3 i jest nieparzyste, to do PROP dodać 1 i R zmniejszyć o 1, a następnie SEG1 = SEG2 = R/2.
Jeżeli R > 3 i jest parzyste, to SEG1 = SEG2 = R/2.

KROK 5: Wyznaczenie szerokości przedziału dosynchronizowania SJW bitu (Synchronized Jump Width).

Musi być spełniony warunek: SJW <= 4 i SJW <= SEG1

SJW jest tym większy im mniej precyzyjny i stabilny jest oscylator (np. ceramiczny).

Zwykle wystarcza SJW = 1[t_Q].

KROK 6: Obliczenie błędu częstotliwości CAN Δf_{CAN} wynikającego z błędu częstotliwości oscylatora Δf_{OSC}.

Δf_{OSC} jest zapisany w karcie katalogowej mikrokontrolera lub oscylatora i oznacza względne odchylenia rzeczywistej częstotliwości oscylatora f_R od częstotliwości wzorcowej f_{OSC}:

$$\Delta f_{OSC} = |(f_R - f_{OSC})/f_{OSC}|$$

Obliczyć wahania czasu $\Delta t_{\Delta f}$ wynikające z błędu Δf_{OSC} wg wzoru:

$$\Delta t_{\Delta f} = \Delta f_{OSC}/f_{OSC}[s]$$

Ostatecznie obliczyć:

$$\Delta f_{CAN} = (2 * \Delta t_{\Delta f})/(10 * t_{BIT})$$

KROK 7: Obliczenie dopuszczalnego błędu częstotliwości Δf_{CAN}.

Obliczyć:

$\Delta f_1 < SJW/(20*N_{tQ})$
$\Delta f_2 < MIN(SEG1, SEG2)/(2*(13* N_{tQ} - SEG2))$

Przyjąć:

$$\Delta f = MIN(\Delta f_1, \Delta f_2).$$

Jeżeli wynik jest niezadowalający, to powrócić do kroku 2 i tam zmienić t_Q oraz N_{tQ}.
Jeżeli mimo zmian t_Q i N_{tQ} błąd Δf jest zawsze mniejszy od Δf_{CAN}, to należy wymienić oscylator na inny (o większej częstotliwości).

PRZYKŁAD

Wyznaczyć parametry czasowe dla transmisji pojedynczego bitu na magistrali CAN dla danych:

- prędkość transmisji CAN: f_{CAN} = 1Mb/s,
- długość magistrali CAN: l_{Bus}=1m,
- czas zwłoki sygnału na magistrali CAN: t_{Bus} = 5ns/m,
- suma zwłok nadajnika Tx i odbiornika Rx: $t_{Tx} + t_{Rx}$=130ns,
- częstotliwość oscylatora: f_{OSC} = 16MHz,
- błąd częstotliwości oscylatora: Δf_{OSC} = 2%.

KROK 1

$$t_{PROP} = 2(t_{Bus} * l_{Bus} + t_{Tx} + t_{Rx}) = 2(5*1+130)ns = 270ns$$

KROK 2

$$t_{BIT} = 1/f_{CAN}\ [s] = 10^{-6}s = 1000ns,$$

Dla częstotliwości oscylatora f_{OSC} = 16MHz dobrać całkowitą liczbę PRESC (współczynnik skalujący) i obliczyć kwant czasu t_Q:

$$t_Q = PRESC/f_{OSC}\ [s]$$

tak, aby dla liczby całkowitej N_{tQ} takiej, że $7 < N_Q < 26$, spełniona była równość:

$$N_{tQ} * t_Q = t_{BIT}.$$

Dla PRESC = 2 mamy:
$t_Q = (2/(16*10^6))s = 0{,}125*10^{-6}s = 125ns$,
$N_{tQ} = t_{BIT}/t_Q = 1000/125 = 8$.
$N_Q = 8$, więc PRESC = 2 jest dobry.
Dla PRESC = 4 mamy:
$t_Q = (4/(16*10^6))s = 0{,}25*10^{-6}s = 250ns$,
$N_{tQ} = t_{BIT}/t_Q = 1000/250 = 4$.
$N_Q < 8$, więc PRESC = 4 jest niedopuszczalny.

KROK 3

$$PROP = ROUND\_UP(t_{PROP}/t_Q) = ROUND\_UP(270/125) = 3,$$

KROK 4

$$R = N_{tQ} - PROP - SYNC = 8 - 3 - 1 = 4,$$

stąd SEG1 = SEG2 = R/2 = 4/2 = 2.

KROK 5

Zgodnie z warunkiem SJW <= 4 i SJW <= SEG1
możemy więc przyjąć SJW = 1 lub SJW = 2;

KROK 6

$\Delta t_{\Delta f} = \Delta f_{OSC}/f_{OSC}[s] = 0.02/(16 * 10^6)[s] = 1.25ns$
$\Delta f_{CAN} = (2 * \Delta t_{\Delta f})/(10 * t_{BIT}) = (2 * 1.25ns)/(10 * 1000ns) = 0.00025 = 0.025\%$

KROK 7

Dla SJW = 1 mamy:
$\Delta f_1 < SJW/(20*N_{tQ}) = 1/160 = 0.00625 \Rightarrow 0.625\%$.

Dla SJW = 2 mamy:
$\Delta f_1 < SJW/(20*N_{tQ}) = 2/160 = 0.0125 \Rightarrow 1.25\%$.

$\Delta f_2 < MIN(SEG1, SEG2)/(2*(13*N_{tQ} - SEG2)) = 1/(13*8-2)=2/102 = 0{,}01961 \Rightarrow 1.961\%$

Stąd:

dla SJW = 1

$$\Delta f = MIN(\Delta f_1, \Delta f_2) = MIN(0.625, 1.961)\% = 0.625\%,$$

dla SJW = 2

$$\Delta f = MIN(\Delta f_1, \Delta f_2) = MIN(1.25, 1.961)\% = 1.25\%,$$

W związku z tym, że $\Delta f_{CAN} = 0.025\% < \Delta f = 0.625\%$, przyjmujemy SJW = 1.

4. Filtracja wiadomości w odbiorniku CAN

Wszystkie stacje podłączone do magistrali SAN, łącznie z nadajnikiem, odbierają każdą wiadomość, która pojawi się na magistrali. Układ CAN może filtrować odebrane wiadomości i przekazywać do mikrokontrolera tylko te, które dany mikrokontroler może przetworzyć. Jest to bardzo duże odciążenie mikrokontrolera wobec możliwości pojawienia się na magistrali CAN tysięcy wiadomości o różnych identyfikatorach, z których tylko nieliczne przeznaczone są dla określonej stacji.

Tab. 3. Akceptacja lub odrzucenie identyfikatora wiadomości w zależności od filtra i maski

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

Wiadomość jest przekazywana do mikrokontrolera, gdy wszystkie bity identyfikatora wiadomości są akceptowane przez odpowiednią maskę i skojarzony z nią filtr CAN ustawione w odpowiednich rejestrach.

Z powyższej tabeli widać, że:

1. Jeżeli bit maski na określonej pozycji jest zgaszony, to niezależnie od wartości bitu filtra na tej pozycji akceptowany jest odpowiedni bit identyfikatora wiadomości.
2. Jeżeli bit maski na określonej pozycji jest ustawiony, to bit identyfikatora wiadomości jest akceptowany tylko wtedy, gdy jest zgodny z bitem filtra na tej pozycji.

Powyższe zależności można zapisać postaci zdania logicznego:

$$\sim \mathrm{bit}_M \vee (\mathrm{bit}_F \wedge \mathrm{bit}_{ID}) \vee (\sim \mathrm{bit}_F \wedge \sim \mathrm{bit}_{ID})$$

gdzie:

- bit ustawiony oznacza wartość logiczną prawda, a zgaszony oznacza fałsz,
- bit_M oznacza bit maski,
- bit_F oznacza bit filtru,
- bit_{ID} oznacza bit identyfikatora wiadomości,
- wszystkie bity są na tej samej pozycji w swoich grupach.

Jeżeli zdanie jest prawdziwe, to bit identyfikatora wiadomości jest akceptowany, a w przeciwnym wypadku nieakceptowany.

5. Podsumowanie

Interfejs CAN powstał na potrzeby przemysłu samochodowego, ale jego zalety takie, jak: bardzo duża prędkość transmisji, odporność na zakłócenia, sprawne procedury obsługi błędów, zabezpieczenia przed zablokowaniem magistrali, podłączenie do magistrali bardzo dużej liczby stacji, niezawodny arbitraż dostępu do magistrali (nadawania) bez wyznaczania stacji głównej i możliwość utworzenia 2^{29} różnych identyfikatorów wiadomości sprawiają, że magistrala CAN może być z powodzeniem stosowana w urządzeniach militarnych. Pewnym ograniczeniem jest mała liczba, bo maksymalnie 8 bajtów danych. Dla znacznej większości czujników i pomiarów ta liczba jest wystarczająca.

Literatura

[1] *PIC18F2480/2580/4480/4580 sheet*, Microchip Technology Inc., 2009r.
[2] www.freescale.com, Stuart Robb, AN1798 CAN Bit Timing Requirements, 2012r.
[3] Magistrala CAN, cz.1, *Elektronika Praktyczna*, nr 1, str. 21 ÷ 24, 2000r.
[4] Magistrala CAN, cz.2, *Elektronika Praktyczna*, nr 2, str. 21 ÷ 24, 2000r.
[5] Magistrala CAN, cz.3, *Elektronika Praktyczna*, nr 3, str. 13 ÷ 16, 2000r.