WYDAWNICTWO

ATNEL

Magistrala CAN od A do Z. Diagnostyka i programowanie w języku C

Paweł Kardaś

Szczecin 2020

Mojemu Dziadkowi

Autor: Paweł Kardaś
Redakcja techniczna: Mirosław Kardaś
Opracowanie graficzne: Paweł Kardaś
Projekt okładki: Karolina Kardaś

Wydawnictwo Atnel
70-795 Szczecin
ul. Kurza 24
tel. 91 46 35 683
e-mail: biuro@atnel.pl
www.atnel.pl

ISBN 978-83-931797-6-3

Szczecin 2020

Spis treści

2.1. CAN w modelu ISO/OSI

Model OSI (ang. *Open System Interconnection*) został stworzony przez organizację ISO (ang. *International Standard Organization*) jako standard definiujący schemat komunikacji dla urządzeń elektronicznych wysyłających i odbierających dane w sieci. Głównym celem było określenie wspólnych zasad wymiany informacji dla wszystkich technologii komunikacyjnych opartych na tym modelu. Model systemu sieciowego OSI podzielono na siedem ściśle powiązanych ze sobą warstw. Opisuje on drogę poprzez warstwy, jaką musi przebyć informacja wysłana z aplikacji w jednym systemie do aplikacji w drugim systemie, przy czym każda z warstw odpowiada za realizację określonych zadań związanych z całym procesem komunikacji. Opis warstw w modelu ISO/OSI, począwszy od najniższej, obejmuje:

- Warstwa 1 – **fizyczna –** określa fizyczne elementy i parametry interfejsu sieciowego, umożliwia dwustronną konwersję sygnału elektrycznego do postaci strumienia binarnego.
- Warstwa 2 – **łącza danych** – definiuje zasady dostępu do medium komunikacyjnego, umożliwia wykrywanie i zgłaszanie błędów transmisyjnych, tworzy ramki komunikacyjne.
- Warstwa 3 – **sieciowa** – tworzy pakiety danych oraz odpowiada za tzw. routing, czyli utworzenie i zestawianie optymalnych dróg transmisji pomiędzy urządzeniami.
- Warstwa 4 – **transportowa –** zapewnia transmisję zgodną z wymaganymi parametrami, optymalizuje zużycie usług sieciowych.
- Warstwa 5 – **sesji –** odpowiada za mechanizmy pozwalające na synchronizację danych i zarządzanie wymianą informacji.
- Warstwa 6 – **prezentacji –** przekształca otrzymane informacje pod różnymi postaciami do jednolitego formatu – wygodnego do przedstawienia dla użytkownika.
- Warstwa 7 – **aplikacji** – pełni funkcję interfejsu między użytkownikiem a usługami sieciowymi w postaci aplikacji czy programów.

Przedstawione wcześniej etapy w procesie wysyłania i odbierania danych w magistrali CAN, realizowane w warstwie zarówno fizycznej, jak i łącza danych, zaimplementowane są w gotowych układach scalonych (sterowniki i kontrolery CAN), których zadaniem jest realizowanie transmisji. Przedstawiony na rysunku 6 schemat transmisji CAN oparty na modelu ISO/OSI obrazuje jedną z ważniejszych zalet technologii, jaką jest odciążenie użytkownika od kwestii związanych z formatowaniem ramek danych, arbitrażem, obsługą błędów czy filtrowaniem informacji.

2.1.1. Warstwa fizyczna – parametry transmisyjne w CAN

Warstwa fizyczna w CAN, podobnie jak w modelu ISO/OSI, definiuje parametry fizyczne medium transmisyjnego. Określa współzależność między szybkością przesyłania danych, długością przewodów, rodzajem zastosowanego medium oraz impedancją zamykającą. Z praktycznego punktu widzenia znajomość i odpowiednie dobranie wspomnianych parametrów podczas konstrukcji sieci mogą mieć realny wpływ na poprawne albo wadliwe działanie całego systemu. Przede wszystkim należy rozważyć maksymalną prędkość transmisji, biorąc pod uwagę długość całej magistrali, ponieważ im dłuższe jest medium transmisyjne (skrętka), tym bardziej wydłuża się czas transmitowania bitów (zjawisko propagacji sygnału). Brak konkretnych wyliczeń w tym zakresie może przyczynić się do dodatkowych problemów z synchronizacją. Zależności pomiędzy długością magistrali a możliwą do uzyskania prędkością zostały przedstawione na rysunku 7.

Prędkość transmisji (Baud Rate) kbit/s	Rezystancja zamykająca magistralę Ω	Długość magistrali (Bus Lenght) m
1000	124	40
500	127	100
100	150 ~ 300	500
50	150 ~ 300	1000

Rysunek 7. Parametry transmisji CAN – prędkość transmisji, rezystancja zamykająca, długość magistrali

Magistrala CAN domyślnie (przy braku transmisji) utrzymana jest w stanie recesywnym (wysokim po stronie sygnałów TTL). Podczas transmitowania wiadomości węzły odbiorcze oraz nadawcze zmieniają stan magistrali na dominujący (niski) lub recesywny (wysoki) na czas trwania poszczególnych bitów w przesyłanych ramkach. Czas trwania pojedynczego bitu, a także prędkość przesyłanych informacji (ang. *Baud Rate*) wyrażanej w kbit/s (kilobitach na sekundę) zależy od długości przewodów, na podstawie których została zbudowana magistrala CAN. Na rysunku 8 przedstawiono zależności pomiędzy wybranymi prędkościami transmisji, czasem trwania pojedynczego bitu a dopuszczalną dla nich długością magistrali.

Prędkość transmisji (Baud Rate) kbit/s	Czas trwania bitu (Nominal Bit-Time) µs	Długość magistrali (Bus Lenght) m
1000	1	40
500	2	110
250	4	280
125	8	620

Rysunek 8. Parametry transmisji CAN – Prędkość transmisji, czas trwania bitu, długość magistrali

3. Zasady transmisji i protokołu CAN

Wymiana informacji między węzłami w sieci CAN odbywa się według zasad określonych w protokole komunikacyjnym CAN. Do przesyłu informacji między węzłami zastosowano transmisję różnicową zrealizowaną na podstawie skręconej pary przewodów, podczas której sygnały elektryczne docierają do układów wejściowych/wyjściowych sterowników CAN. Sygnały różnicowe zamieniane są przez sterownik (transceiver) na sygnały w standardzie TTL (warstwa fizyczna), które następnie są przekazywane do układu kontrolera CAN. W drugą stronę sygnał TTL z kontrolera CAN przekazywany jest do sterownika CAN, gdzie zamieniany jest na sygnał różnicowy. W dalszej części tego rozdziału wyjaśniono:

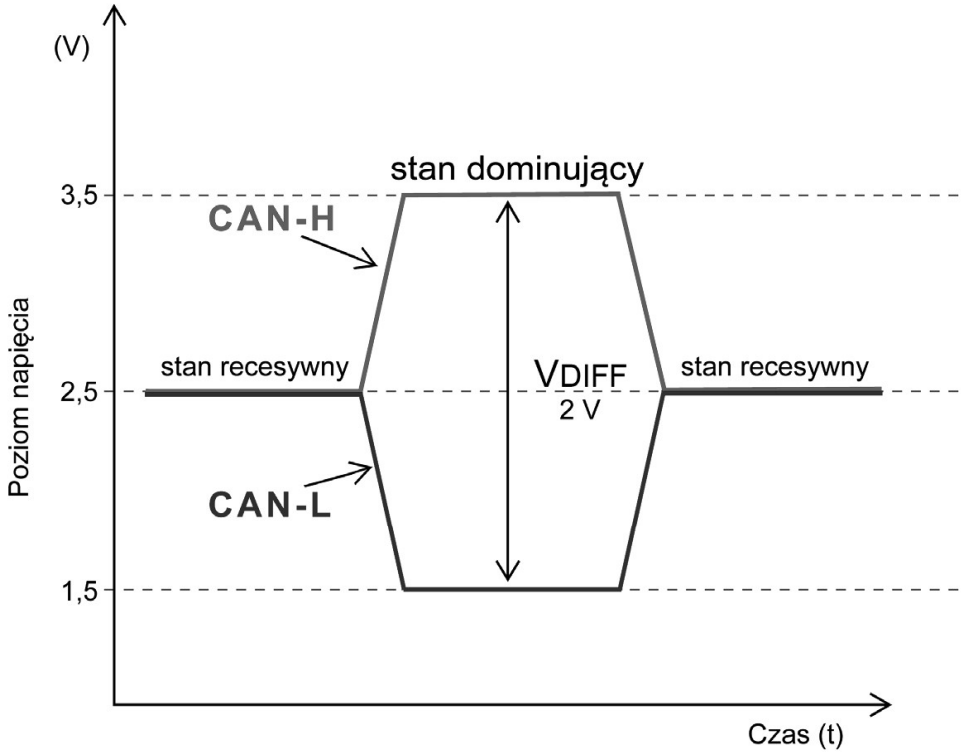
- proces zamiany sygnału różnicowego na sygnał w standardzie TTL,
- rodzaj kodowania danych podczas transmisji cyfrowej,
- zasady budowy ramek CAN,
- proces arbitrażu wiadomości,
- sposoby adresowania i identyfikacji wiadomości,
- mechanizmy wykrywania i usuwania błędów,
- proces filtrowania wiadomości w sieci CAN.

3.1. Transmisja różnicowa

Jak wspomniano wcześniej, medium służącym do realizacji fizycznego połączenia jest tzw. skrętka, para dwóch splecionych ze sobą przewodów. Jest to takie samo podejście jak w przypadku magistrali RS485/422. Skrętka ma przede wszystkim zapewnić znacznie większą

odporność na zakłócenia podczas transmisji danych przy założeniu, że wykorzystuje się transmisję różnicową. W przypadku magistrali CAN stan bitu dominującego reprezentowany jest przez różny poziom napięcia na liniach CAN-H i CAN-L. Rozróżnia się dwa rodzaje bitów.

1. bit dominujący = stan logiczny NISKI (0),
2. bit recesywny = stan logiczny WYSOKI (1).



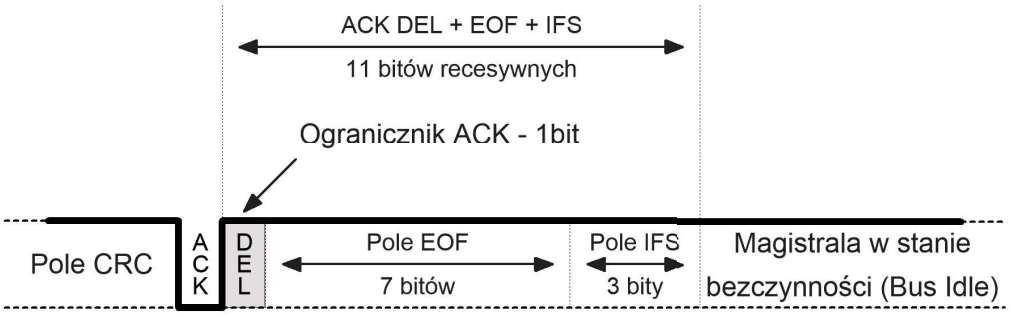
Rysunek 15. Poziomy napięć podczas transmisji różnicowej magistrali CAN

Poziom napięcia bitu recesywnego na obu liniach posiada praktycznie ten sam poziom (2,5 V), po translacji do poziomu TTL jest to zaś stan wysoki. W związku z powyższym bit dominujący na magistrali CAN odpowiada za stan niski po translacji do poziomów TTL.

Na rysunku 15 przedstawiono szczegółową prezentację poziomów napięć dla stanów logicznych na poziomie magistrali różnicowej. Zgodnie z prezentowanym wykresem na linii CAN-H napięcie zależne

Przerwa między ramkami:

IFS (ang. *Intermission*) – stanowi minimalną przerwę przed każdą następną ramką (danych, zdalnego wywołania, błędu i przepełnienia) o czasie trwania trzech bitów recesywnych. Podczas trwania tej przerwy żaden z węzłów nie może rozpocząć transmisji ramek danych i zdalnego wywołania. Wyjątkiem w tym przypadku jest ramka przepełnienia, o której mowa będzie dalej.



Rysunek 26. Przerwa między ramkami – Interframe Space (IFS)

Jak można zauważyć na rysunku 26, po trzech bitach pola IFS magistrala przechodzi w stan bezczynności (ponieważ na magistrali przez czas 11 bitów utrzymywał się stan recesywny – pola ACK DEL, EOF i IFS). Po ostatnim bicie IFS magistrala jest wolna, co oznacza, że węzły mogą rozpocząć kolejną transmisję.

Ramka danych w sieci CAN może mieć długość od 47 do 111 bitów (wchodzących w zakres omawianych pól bitowych). Różnica w liczbie bitów ramki wiąże się z możliwą różną liczbą przesyłanych bajtów danych i bitów dodanych przez mechanizm Bit Stuffing.

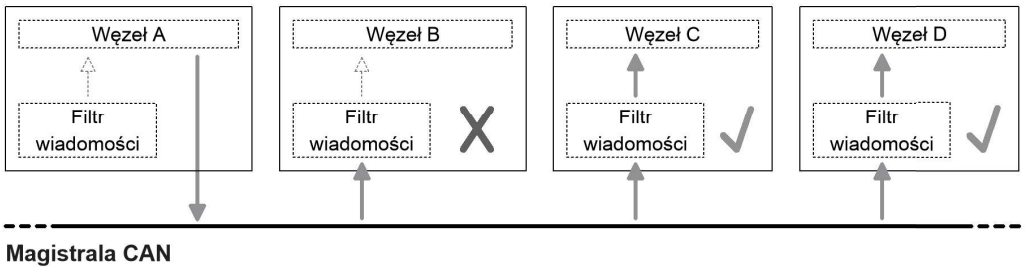
Jedną z różnic pomiędzy ramkami standardowymi w CAN 2.0A i CAN 2.0B jest bit IDE, który w CAN 2.0B występuje na miejscu jednego z bitów opisanych jako r1 (ang. reserved) w CAN 2.0A. Stan dominujący bitu IDE oznacza, że jest to ramka o identyfikatorze 11-bitowym,

sposoby adresowania, proces projektowania protokołów z wyższych warstw aplikacji oraz samo przygotowanie struktury sieci CAN.

3.5. Adresowanie i identyfikacja wiadomości

Sieć CAN można nazwać siecią rozgłoszeniową (*broadcast*), w której wszystkie węzły mają takie same uprawnienia (*multi-master*). Oznacza to, że każdy węzeł w tym samym czasie ma możliwość rozpoczęcia nadawania, pod warunkiem że na magistrali utrzymuje się poziom recesywny, czyli magistrala jest w stanie bezczynności.

Transmitowana wiadomość w sieciach CAN nie zawiera informacji o dostawcy czy odbiorcy, ale jedynie swój własny unikalny numer identyfikacyjny (ID). Mechanizmy filtrowania wiadomości węzłów odbiorczych są skonfigurowane w taki sposób, aby odrzucać lub akceptować wiadomości o zdefiniowanych wartościach bitów w polu ID transmitowanej ramki.



Rysunek 47. Wysyłanie i odbieranie wiadomości w sieci CAN

Jak wspomniano wcześniej, każdy węzeł sprawdza cały czas stan magistrali. W przypadku rozpoczęcia transmisji przez kilka węzłów następuje proces arbitrażu danych. Finalnie tylko jeden węzeł transmituje do końca swoją wiadomość. W tym czasie każdy z węzłów na magistrali przetwarza bit po bicie nadawanej ramki. W momencie pojawienia się

pola ACK wystawia odpowiedni stan w zależności od tego, czy ramka została poprawnie odebrana, czy też został wykryty błąd. W drugim przypadku węzeł, który wykrył błąd, zgłasza go poprzez wyemitowanie ramki błędu i blokuje tym jednocześnie dotychczasowy „ruch" w sieci, czyli transmitowaną ramkę, w której wykryto błąd.

W przypadku poprawnego odebrania wiadomości kontroler CAN za pomocą mechanizmów filtrujących decyduje, czy ramka, która „nadleciała", interesuje go i czy będzie przetwarzał odebrane informacje, czy też zignoruje nową wiadomość. Na rysunku 47 przedstawiono proces wysyłania ramki danych przez jeden (węzeł A) z czterech węzłów podłączonych do przykładowej magistrali CAN. Mechanizmy filtrujące węzłów C oraz D odebrały informacje, nie wykrywając przy tym błędu, i zdecydowały o jej dalszym przetwarzaniu, filtry węzła B zignorowały natomiast poprawnie odebraną ramkę.

Do zadań projektanta sieci CAN należy przygotowanie odpowiedniej struktury numerów ID dla wszystkich ramek, pamiętając o wyższych priorytetach wiadomości z niższym numerem ID. Przemyślana struktura pola ID oraz odpowiednia konfiguracja filtrów we wszystkich węzłach może przyczynić się do zminimalizowania ryzyka występowania błędów podczas transmisji. Istnieje kilka niepisanych reguł, w jaki sposób warto podzielić identyfikator transmitowanej ramki. Podstawowe zasady filtrowania wiadomości na podstawie kontrolera CAN wbudowanego w mikrokontroler AT90CAN128 zostały wyjaśnione w rozdziale 3.7. Zasady filtrowania wiadomości.

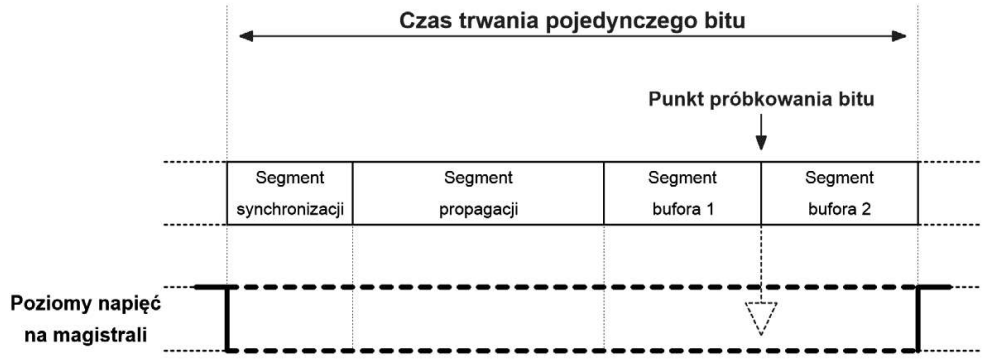
3.6. Mechanizmy wykrywania i usuwania błędów

W ramach standardu CAN protokół transmisji został wyposażony w kilka mechanizmów służących do wykrywania, natychmiastowego

badanego bitu. Zwykle zajmuje dwa kwanty czasu. Pojęcie kwantu czasu zostanie wyjaśnione później.

4.1.2. Podział bitu na fazy

W celu określenia dokładnego momentu próbkowania bitu (ang. *Bit Sample Point*) czas trwania bitu zostaje podzielony na cztery nienachodzące na siebie fazy (dotyczy to każdego transmitowanego bitu), jak zaprezentowano na rysunku 57.



Rysunek 57. Podział bitu na cztery fazy

Segment synchronizacji – Sync_Seg:

Jest używany do synchronizacji wszystkich węzłów w sieci CAN. W ramach tego segmentu zawsze musi wystąpić zbocze opadające, dlatego wykorzystywany jest do tzw. twardej synchronizacji. Dzięki temu resetowane są programowe liczniki służące do badania czasu bitu, gdy w trakcie trwania kilku bitów nie następuje żadna zmiana zbocza.

- Segment czasu propagacji – Prop_Seg:
- Służy do kompensacji opóźnień związanych z propagacją sygnału. Pod uwagę brane są tutaj zarówno opóźnienia związane z czasem rozchodzenia się sygnału w magistrali, jak i opóźnienia związane z zastosowanymi sterownikami CAN.
- Segment bufora 1 i 2 – Phase_Seg1/2:

6. Oprogramowanie transmisji CAN – AVR Message Object

Z poprzednich rozdziałów wiadomo już, na czym polega transmisja danych w protokole CAN, jak wyglądają i z jakich bitów składają się ramki, jak obliczyć czas trwania pojedynczego bitu oraz jak wyznaczyć prędkość transmisji.

Do omówienia pozostała jeszcze kwestia podejścia programistycznego, czyli jak w praktyce zaimplementować przedstawioną w tej publikacji wiedzę teoretyczną we własnym systemie z zastosowaniem mikrokontrolera. W tym rozdziale zostaną omówione następujące zagadnienia:

- implementacja własnego urządzenia jako węzła w sieci CAN,
- przygotowanie układu kontrolera CAN do pracy w sieci: ustawienie prędkości transmisji, wyznaczenie czasu trwania poszczególnych faz pojedynczego bitu i wyznaczania ich czasu w jednostkach Time Quantum (kwantów czasu),
- konstrukcja ramki danych w języku C (warstwa aplikacji),
- wysyłanie wiadomości,
- filtrowanie i odbieranie wiadomości,
- realizacja prototypu przykładowej sieci CAN na podstawie dwóch węzłów wyposażonych w mikrokontrolery AT90CAN128 (dwa zestawy uruchomieniowe ATB 1.05a firmy Atnel oraz moduły ATB CAN SHIELD do tych zestawów).

W kolejnych rozdziałach zostaną omówione najważniejsze rejestry kontrolera CAN wbudowanego w mikrokontroler, których wykorzystanie umożliwi realizację transmisji danych w przykładowej sieci CAN

składającej się z dwóch węzłów. Pierwsze testy programów odpowiedzialnych za wysyłanie oraz filtrowanie i odbieranie wiadomości zostaną zweryfikowane za pomocą narzędzia ATB CAN-LIN OCTOPUS oraz oprogramowania CAN SHARK firmy Atnel.

6.1. Implementacja własnego urządzenia jako węzła w sieci CAN

W celu zaimplementowania własnego prostego węzła w sieci CAN warto przygotować:

- układ kontrolera CAN, np. MCP2515 lub zamiennie mikrokontroler AT90CAN128 albo ATmega64M1 z wbudowanym układem kontrolera (warto zwrócić uwagę na liczbę dostępnych obiektów wiadomość w kontrolerze CAN – im więcej MOb posiada kontroler, tym więcej wiadomości będzie mógł przetwarzać w tym samym czasie),
- układ sterownika CAN, np. MCP2551, który będzie modułem wejściowym/wyjściowym węzła,
- rezonator kwarcowy o wartościach całkowitych częstotliwości, np. 16 MHz, umożliwi i ułatwi dobranie odpowiedniej prędkości Baud Rate,
- rezystory terminujące 120 Ω, podłączone na obu końcach magistrali, linii CAN-H i CAN-L.

Na rysunku 68 przedstawiono przykładowy schemat połączeń elektronicznych, którego zastosowanie we własnym urządzeniu umożliwi podłączenie go do sieci CAN.

7. Biblioteka MK_ATCAN_LIB – praktyczne przykłady zastosowań

Z uwagi na fakt przygotowania zaawansowanej biblioteki do obsługi komunikacji CAN opartej na mikrokontrolerach AVR z wbudowanym kontrolerem niniejszy rozdział poświęcono na prezentację jej wykorzystania w podstawowym zakresie. Biblioteka została przygotowana dla języka AVR GCC. Prezentacja dotyczy zastosowania podstawowych narzędzi dostępnych w ramach biblioteki, przykład oparto zaś na niewielkim projekcie. Zakłada on stworzenie niewielkiej sieci CAN składającej się z czterech węzłów. Dwa węzły będą pełniły funkcję zdalnych czujników temperatury, trzeci węzeł będzie serwerem zbierającym dane z czujników. W ramach czwartego węzła do sieci CAN zostanie podłączone narzędzie diagnostyczne ATB CAN-LIN OCTOPUS. Cała sieć będzie oparta na zestawach uruchomieniowych ATB 1.05a firmy Atnel.

Szczegółowe założenia:

1. Węzły zdalnych czujników temperatury będą oparte na cyfrowych czujnikach 1wire DS18B20. Pomiar będzie dokonywany cyklicznie z określonym interwałem czasowym. Po zebraniu pomiaru wartość temperatury będzie przesyłana do sieci CAN. Każdy węzeł będzie wysyłał własną temperaturę pod określony numer ID na zasadzie rozgłoszeniowej, bez definiowania odbiorcy.
2. Węzeł serwera zaopatrzony zostanie w wyświetlacz alfanumeryczny LCD 2x16, na którym w każdym wierszu będzie prezentowana on-line temperatura z każdego ze zdalnych węzłów. Dodatkowo serwer zostanie podłączony do komputera PC za pomocą łącza

USB/RS232, aby mógł wysyłać odebrane wartości temperatury z każdego ze zdalnych węzłów bezpośrednio do terminala.
3. Prędkość transmisji w sieci (Baud Rate) zostanie ustalona na 125 kbps.
4. Do budowy zdalnych węzłów pomiaru temperatury zastosowano mikrokontrolery ATmega64M1, do serwera zaś – AT90CAN128.
5. W celu weryfikacji poprawności transmisji do magistrali zostanie podłączone narzędzie diagnostyczne ATB CAN-LIN OCTOPUS, aby weryfikować poprawność zaprojektowanych i przesyłanych przez programistę ramek CAN.

W celu lepszej prezentacji możliwości biblioteki wykorzystane zostaną jej podstawowe funkcje związane z obsługą komunikacji za pomocą zdarzeń (Event's): rejestracją funkcji zwrotnych (typu Callback), filtracją wybranych ramek ze ściśle zdefiniowanymi numerami ID (biblioteka przeznaczona jest dla wszystkich procesorów AVR z serii AR90CANxxx oraz ATmegaXXm1).

Sama biblioteka oparta jest przede wszystkim na zaawansowanym cyklicznym buforowaniu zarówno odbieranych, jak i nadawanych wiadomości, z pełną możliwością konfiguracji wszystkich parametrów z tym związanych. Dzięki temu programista może dowolnie optymalizować własne rozwiązania i zajętość pamięci RAM, a także uzyskać możliwie szybki efekt wdrożenia projektu w wersji przeznaczonej dla ostatecznie produkowanych własnych urządzeń.

Na rysunku 102 przedstawiono schemat blokowy układu, który został wykorzystany do przetestowania kodów źródłowych przedstawionych w tym rozdziale.

8. LIN (Local Interconnect Network)

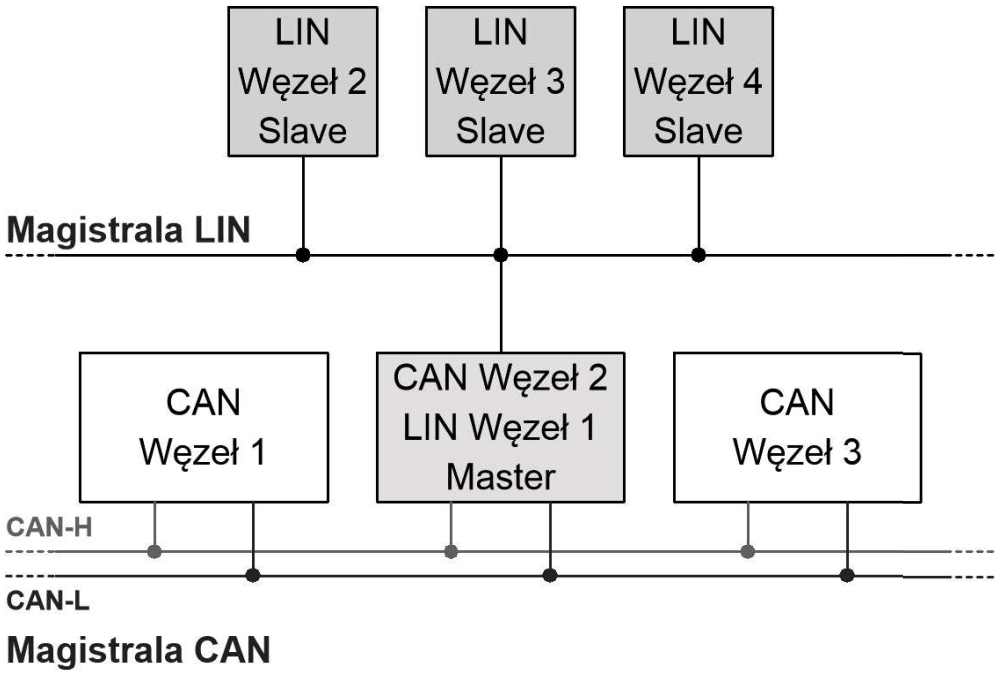
Protokół LIN (ang. *Local Interconnect Network*) został opracowany w celu utworzenia tańszej w implementacji alternatywy dla sieci CAN. Wersja protokołu LIN 1.0 została opracowana w połowie 1999 roku na bazie technologii Volcano-Lite opracowanej przez firmę Volvo Volcano Communications Technology (VCT). Po kilku większych modyfikacjach pod koniec 2002 roku został wydany LIN 1.3. Zaledwie rok później wydano wersję protokołu LIN 2.0. W dzisiejszych czasach niemal każdy pojazd wyposażony jest w kilka magistral LIN, które obsługują różne strefy komfortu, jak np. automatycznie opuszczane szyby, klimatyzacja, wycieraczki.

8.1. Specyfikacja magistrali LIN

Budowa protokołu LIN jest zgodna z modelem ISO/OSI. Podobnie jak w przypadku CAN główną rolę w procesie komunikacji odgrywają warstwy: fizyczna, łącza danych i aplikacji.

Specyfikacja LIN:
- Magistrala LIN ma charakter Master-Slave.
- Jedna sieć LIN może się składać z jednego układu Master i maksymalnie 16 węzłów Slave.
- Napięcie operacyjne – 12 V.
- Warstwa fizyczna bazuje na standardzie ISO 9141 (K-Line).
- Węzły wyposażone w tryb uśpienia i obsługę budzenia.
- Protokół LIN obsługuje proces wykrywania błędów oraz sumę kontrolną.

- Do transmisji danych wykorzystywany jest jeden przewód sygnałowy oraz GND.
- Maksymalna prędkość transmisji wynosi 20 kbps dla maksymalnej długości medium transmisyjnego wynoszącego 40 metrów.
- O wiele mniejszy koszt implementacji w porównaniu z CAN (jeżeli prędkość transmisji i niezawodność nie są elementami kluczowymi w danym systemie).
- Ramki w protokole LIN zawierają 6-bitowe identyfikatory.



Rysunek 106. Przykładowy schemat połączeń magistrali LIN i CAN

Magistrala LIN jest często implementowana w rozbudowanych systemach składających się z kilku sieci CAN. Węzeł Master stanowi łącznik między wybranym podzespołem elektroniki a pozostałą częścią większego systemu (np. w nowoczesnych samochodach, gdzie pojedyncza magistrala LIN odpowiada za funkcje pojedynczego systemu komfortu, np. elektrycznie opuszczane szyby, wycieraczki, klimatyzacja). Na rysunku 106 zaprezentowano przykładowy schemat połączenia magistrali LIN z siecią CAN.