Metody cyfrowej regeneracji transmisji w sieciach polowych stosujących protokół Modbus RTU

Łukasz Herb

Streszczenie: W niniejszym artykule przedstawiono wybrane sposoby regeneracji transmisji w sieciach polowych oraz przyczyny jej stosowania. Scharakteryzowano protokół Modbus RTU oraz format znaku występujący w nadajnikach UART. Metody regeneracji podzielono, ze względu na wielkość jednocześnie odtwarzanej informacji – bit, znak oraz ramka.

Słowa kluczowe: Modbus, RS485, regeneracja, UART

THE METHODS OF THE DIGITAL TRANSMISSION REGENERATION IN FIELDBUS WITH THE MODBUS RTU PROTOCOL

Abstract: *This paper presents reasons and methods of transmission regeneration in fieldbus. Describes Modbus RTU protocol and character format in UART transmitters. Methods were divided into three types of regeneration: a bit, a character and a frame.*

Keywords: Modbus, RS485, regeneration, UART

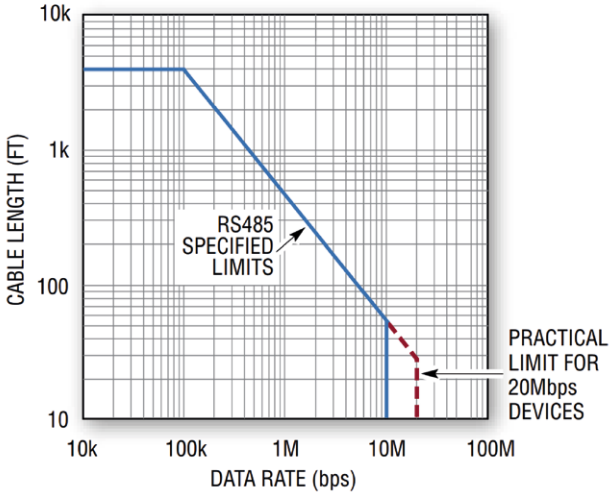
1. Wprowadzenie

Sieci polowe stosowane w przemyśle narażone są na zakłócenia generowane przez środowisko, w którym pracują. Pracujące w pobliżu przewodów komunikacyjnych urządzenia elektryczne generują przewodzone lub promieniowane zaburzenia transmisji, wpływające bezpośrednio na możliwy do uzyskania zasięg oraz prędkość transmisji. Często stosowany w przemyśle standard RS485, transmitujący dane parą skręconych przewodów, pomimo że wykazuje dużą odporność na zakłócenia [4], uzależnia maksymalną możliwą do uzyskania prędkość od odległości między węzłami (rys. 1). Ze standardu wynika maksymalna możliwa do uzyskania prędkość transmisji równa 35 Mb/s na odległości 10 m oraz maksymalny możliwy do uzyskania dystans 1200 m, jednak z prędkością 100 kb/s.

Aby dostosować prędkość oraz zasięg komunikacji, stosuje się urządzenia (repeatery) pośredniczące w wymianie danych. Zwykle posiadają dwa porty nadawczo-odbiorcze, pomiędzy którymi przekazują dane. W najprostszym przypadku są wyposażone w dwa scalone sterowniki magistrali, które przesyłają odebrane dane bez ich interpretacji oraz sprawdzania poprawności. Podobną rolę pełnią konwertery magistrali, mogące przekazywać dane poprzez różne media (na przykład konwertując transmisję symetryczną parą przewodów na komunikację światłowodową), ale zachowując format transmisji. O ile proste urządzenia pośredniczące pozwalają ominąć ograniczenia standardu, nie rozwiązują problemów powstałych w wyniku zaburzeń, a często same pogarszają jakość sygnału (poprzez nieprawidłowe sterowanie linią kierunku transmisji) oraz wprowadzają zakłócenia.

W roli regeneratorów sygnałów mogą wystąpić także urządzenia sieciowe przeznaczone do pracy w topologii magistrali. Często nie są one podłączane równolegle do magistrali, ale transmitują dane przez własne układy elektroniczne, będąc podłączonymi szeregowo w magistralę.

W tym artykule przedstawiony zostanie format znaku transmitowanego w sieciach polowych wykorzystujących UART, a także działanie protokołu Modbus RTU. Na przykładzie opisanego sposobu transmisji przedstawione zostaną trzy sposoby regeneracji z wykorzystaniem układów cyfrowych, ich wpływ na jakość transmisji oraz zależności czasowe w sieci.



Rys. 1. Zależność pomiędzy prędkością transmisji danych a długością kabla [3]

2. Uniwersalny asynchroniczny odbiornik i nadajnik

UART (ang. *Universal asynchronous receiver/transmitter*) to urządzenie peryferyjne konwertujące dane z magistrali równoległej na szeregową. Jest często stosowane w komputerach PC (w postaci interfejsu RS232) lub może zostać dołączone do rdzenia mikroprocesora (8251 produkcji Intela lub Z80-SIO

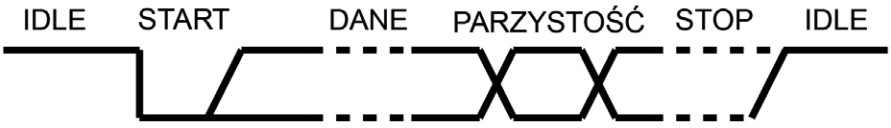
dostarczanego przez Zilog). Wiele mikrokontrolerów (rdzeń procesora wraz z peryferiami) jest wyposażonych w kilka kontrolerów UART (w rodzinie STM32F103 typowo występuje 5 zintegrowanych kontrolerów [5]). Ponadto prosta budowa pozwala na implementację UART-u w strukturach mikroprogramowalnych CPLD oraz FPGA. Opublikowany przez społeczność OpenCores.org blok Minimal UART Core zajmuje 64 elementy *slice* struktury Xilinx Spartan 3E co w przypadku układu XC3S400 pozwala na uruchomienie 56 kontrolerów [5].

Format pojedynczego znaku przesyłanego przez UART przedstawiono na rys. 2. Początek transmisji sygnalizowany jest bitem startu, czyli wyjściem ze stanu jałowego (*idle*), za który przyjęto stan wysoki. Po bicie startu następuje transmisja właściwych danych; może zostać przesłanych od 5 do 9 bitów. Ostatni bit danych może zostać zastąpiony bitem parzystości, uzupełniającym utworzoną liczbę tak, aby suma jej jedynek w reprezentacji binarnej utworzyła liczbę nieparzystą lub parzystą. Następnie pojawia się jeden lub więcej bitów *stop* transmitowanych jako stan wysoki. W skróconej formie format znaku oznacza się liczbą informującą o liczbie bitów danych, literą oznaczającą sposób obliczania bitu parzystości (N – brak, E – do parzystych, O – do nieparzystych, M – zawsze ustawiony stan wysoki) oraz liczbą bitów *stop*. (np. 8N1 – 8 bitów danych, brak parzystości, jeden bit *stop*).

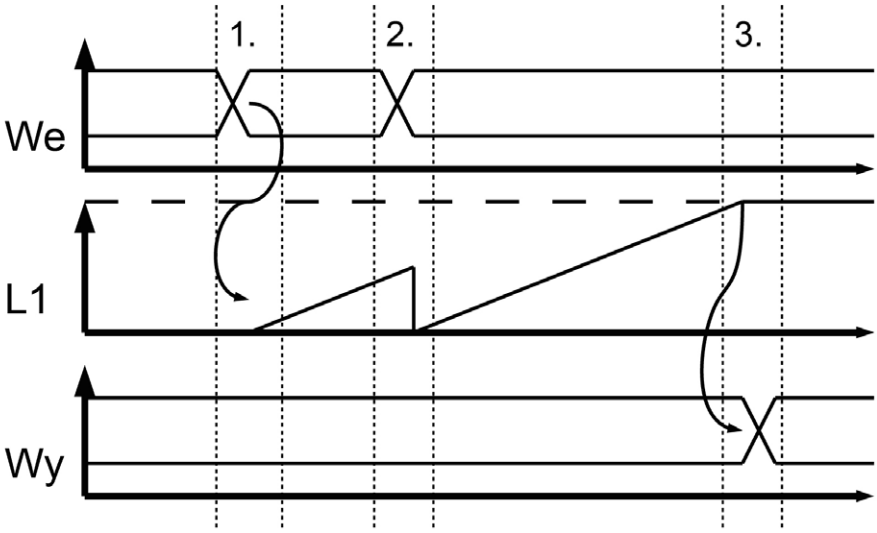
Prędkość transmisji portu szeregowego (prędkość bodowa, *baudrate*) wyrażana jest w bodach lub równych co do wartości bitach na sekundę (b/s). Prędkość transmisji w znakach na sekundę obliczana jest natomiast ze wzoru (1).

$$zbr = \frac{br}{1 + b + p + s} \qquad (1)$$

gdzie:
zbr – prędkość transmisji znaku;
br - prędkość bodowa;
b – liczba bitów danych w znaku;
p – liczba bitów parzystości;
s – liczba bitów stopu.



Rys. 2. Format znaku portu szeregowego UART



Rys. 3. Przebiegi czasowe układu regeneracji jednego bitu

Tabela 1. Ramka protokołu Modbus RTU

START	ADRES	FUNKCJA	DANE	CRC	KONIEC
T1-T2-T3-T4	8b	8b	N * 8b	16b	T1-T2-T3-T4

3. Protokół Modbus RTU

Protokół Modbus opublikowany został przez firmę Modicon w 1979 roku [1]. Ze względu na otwarty standard i łatwość w implementacji jest on nadal szeroko stosowany w sieciach przemysłowych [6]. Opiera się na architekturze klient – serwer. Urządzenie nadrzędne (*Master*) jest klientem pobierającym dane z urządzenia podrzędnego – serwera (*Slave*). Urządzenie podrzędne, zgodnie ze standardem udostępnia przede wszystkim następujące dane [1]:

1. *Discretes Input* – możliwy tylko odczyt - udostępnia stan odczytanych wejść dyskretnych.
2. *Coils* – zapis lub odczyt – działanie w zależności od oprogramowania urządzenia.
3. *Input Registers* – tylko odczyt – udostępnia wartości odczytane z wejść dostępnych w urządzeniu.
4. *Holding Registers* – odczyt lub zapis – działanie w zależności od oprogramowania urządzenia.

Każda wymiana danych odbywa się poprzez wysłanie ramki zapytania przez urządzenie *Master* oraz ramki odpowiedzi przez urządzenie *Slave* [2]. Format ramki został przedstawiony w tabeli 1 i jest w obu przypadkach taki sam. Zakończenie transmisji sygnalizowane jest stanem jałowym, trwającym co najmniej czas nadawania 3,5 znaków (T1-T2-T3-T4).

4. Regeneracja pojedynczego bitu

Ponieważ prędkość transmisji nie ulega zmianie, znany jest czas potrzebny na transmisję jednego znaku oraz czas trwania pojedynczego bitu. Cyfrowy układ regeneracji bitu przedstawiony został w postaci przebiegów czasowych na rys. 3. Posiada on wejście danych z ma-

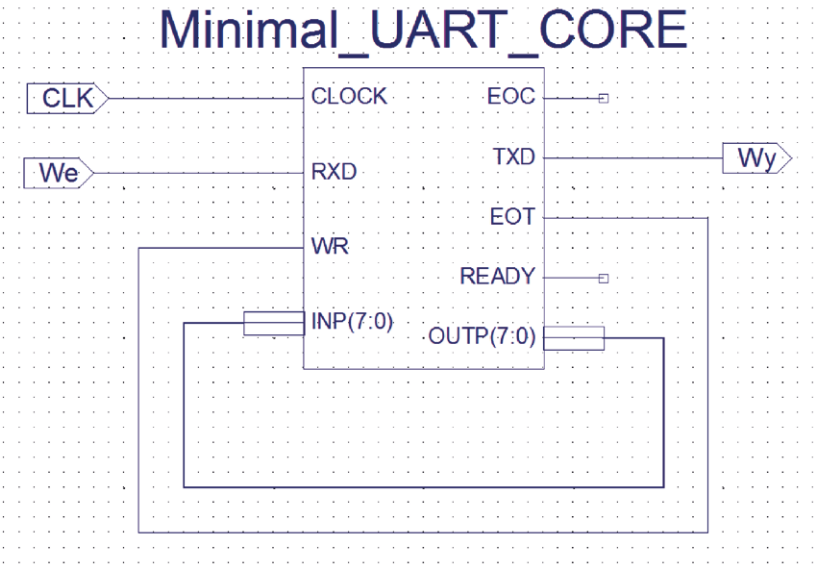
gistrali (We) wyjście zregenerowanego sygnału (Wy) oraz nieuwzględnione na wykresie wejście zegarowe. Pojawienie się zbocza na magistrali (faza 1) sygnalizuje rozpoczęcie transmisji nowego bitu, co rozpoczyna zliczanie impulsów zegarowych (licznik L1). Jeżeli stan magistrali zmienił się przed upływem czasu trwania bitu (faza 2), licznik jest zerowany i następuje ponowne zliczanie. Osiągnięcie przez licznik wartości odpowiadającej liczbie impulsów zegarowych wygenerowanych podczas trwania transmisji jednego bitu oznacza, że bit dotarł niezakłócony i następuje jego przepisanie na wyjście (Wy). Stan wyjścia oraz licznika jest utrzymywany do momentu wystąpienia kolejnego zbocza.

Ten sposób regeneracji jest prosty w implementacji w logikach programowalnych, a także w logice szytej. Pozwala na odfiltrowanie chwilowych zmian stanu na magistrali w stanie jałowym, a także odtwarza zbocza sygnału danych. Metoda ta nie interpretuje jednak całych znaków, przez co zakłócenie zmieniające stan magistrali podczas trwania transmisji powoduje utratę danych.

Modyfikacja rozwiązania, stosująca dodatkowy licznik odmierzający minimalny czas trwania stanu na wyjściu (Wy), pozwala dodatkowo uniewrażliwić regenerator na różnice w czasach opadania i narastania zbocza na magistrali (a zatem czasu trwania jednego bitu). Powstałe w ten sposób przesunięcie fazy sygnału odbieranego i zregenerowanego zniwelowane zostanie podczas stanu jałowego. Niebezpieczeństwem jest wystąpienie takiej różnicy czasu zmiany stanu, która spowoduje przesunięcie w fazie o ponad połowę okresu – wtedy znak zostanie uszkodzony.

5. Regeneracja całego znaku

Dzięki łatwemu dostępowi do układów UART oraz ich implementacji w strukturach programowalnych możliwe jest wykonanie układu regeneracji całego znaku za pomocą sprzężonego nadajnika i odbiornika. W szczególnym przypadku zastosowany może zostać jeden nadajnik/odbiornik (*transceiver*) rys. 4. Magistrala równoległa danych odebranych (OUTP) połączona została z równoległym wejściem danych (INP) do nadania przez nadajnik. Sygnał odebrania



Rys. 4. Schemat bloku regeneracji znaku w środowisku Xilinx ISE

znaku (EOT) jest jednocześnie sygnałem rozpoczęcia transmisji (WR) przez nadajnik. Do układu należy doprowadzić sygnał zegarowy o częstotliwości wymaganej przez zastosowany typ odbiornika i nadajnika, odpowiedni do prędkości transmisji danych.

Odbiór i nadanie tego samego znaku pozwala na eliminację wpływu szumów (zaawansowane układy UART pozwalają na kilkukrotne próbkowanie stanu magistrali podczas odbioru jednego bitu), ale przede wszystkim zachowuje właściwy czas trwania poszczególnych bitów. Ponieważ dane odbierane i nadawane są z taką samą prędkością, a nadajnik i odbiornik taktowane tym samym sygnałem zegarowym, nie występuje niebezpieczeństwo odebrania znaku przed zakończeniem jego nadawania. Jeżeli czas nadawania znaku odbieranego jest krótszy niż wysyłanego, uznaje się go za błędny.

Przedstawiony sposób regeneracji interpretuje jedynie pojedyncze znaki; w przypadku, gdy źródło danych nie zachowuje maksymalnego czasu między znakami, problem zostanie przeniesiony na wyjście.

Jeżeli oprócz prostej logiki programowalnej w urządzeniu pośredniczącym zastosowany zostanie mikrokontroler, możliwe jest zbuforowanie całej przesyłanej ramki. Przed przekazaniem ramki dalej, może zostać sprawdzona jej poprawność, co pozwoli uniknąć generowania niepotrzebnego ruchu w sieci. Dodatkowo przesłanie całej ramki zachowa odpowiedne przerwy międzyznakowe wymagane do prawidłowej interpretacji ramki przez odbiorcę.

Regeneracja całej ramki wprowadza znaczne opóźnienie transmisji, jednak w przypadku dotarcia do regeneratora błędnej ramki odpowiedź szczególna, zawierająca informację o błędzie, jest wysyłana do węzła nadrzędnego natychmiastowo.

6. Regeneracja ramki danych

W systemach czasu rzeczywistego wymieniane dane tracą ważność po przekroczeniu określonego, obliczonego dla danego systemu, czasu. Dlatego należy zauważyć, że wszystkie wymienione wcześniej metody regeneracji wprowadzają do transmisji opóźnienie równe czasowi transmisji odtwarzanych danych. Czas ten wydłuża się wraz ze wzrostem liczby regeneratorów lub abonentów na magistrali, posiadających możliwość regeneracji.

W przypadku regeneracji znaku oraz bitu wyznaczenie czasu opóźnienia jest łatwe do policzenia i opisane zależnością (2).

$$T = i\frac{1}{br} \qquad (2)$$

gdzie:
T – czas wprowadzonego opóźnienia;
i – liczba węzłów regenerujących;
br – prędkość bodowa w przypadku regeneracji bitu lub w przypadku regeneracji znaku prędkość transmisji znaku obliczona według zależności (1).

Obliczone opóźnienie widoczne jest dla urządzenia podrzędnego. Ponieważ odpowiedź także transmitowana jest poprzez regeneratory dla *Mastera* czas ten będzie zawsze dwa razy dłuższy.

W przypadku regeneracji całej ramki protokołu Modbus należy obliczyć liczbę znaków przypadających na pytanie oraz odpowiedź a następnie wynik pomnożyć przez opóźnienie dla jednego znaku z zależności (2).

Podsumowanie

Przedstawione metody są stosowane w urządzeniach sieciowych pracujących w systemach czasu rzeczywistego. Odpowiednio dobrana metoda regeneracji oraz liczba i miejsce umieszczenia regeneratorów pozwalają osiągnąć kompromis pomiędzy wprowadzanym do sieci opóźnieniem a odpornością na zakłócenia i jakością transmitowanego sygnału.

Ze względu na łatwość implementacji przedstawionych rozwiązań mogą one zostać zastosowane w już istniejących urządzeniach wykonanych w oparciu o mikroprocesory i układy programowalne. Ponadto takie węzły, niebędące regeneratorami, mogą w ten sposób, bez ingerencji w obwód elektroniczny, zyskać taką funkcjonalność.

Literatura

[1] Modbus application protocol specification v1.1b, Modbus-IDA – wersja grudzień 2006r. http://www.modbus.org/

[2] Modicon Modbus Protocol Reference Guide. PI–MBUS–300 Rev. J, Modicon – wersja czerwiec 1996r. http://web.eecs.umich.edu/~modbus/

[3] Linear Technology: RS485 Quick Guide. Linear Technology – dostęp październik 2014r. http://www.linear.com/

[4] Mielczarek W.: *Tłumienie zakłóceń i ochrona informacji w systemach pomiarowych*. Politechnika Śląska, skrypt nr 1921, Gliwice 1995.

[5] Herb Ł.: *Zależność czasu trwania cyklu sieci od budowy wewnętrznej koncentratora danych protokołu Modbus*. „Napędy i Sterowanie" 11/2013, s. 84–89

[6] Kluk P.: *Wykorzystanie protokołu komunikacyjnego MODBUS RTU w urządzeniach EA Z bazujących na platformie ARM*. „Elektronika" 5/2007.

Łukasz Herb – Politechnika Śląska, Instytut Informatyki,
e-mail: lukasz.herb@polsl.pl.

artykuł recenzowany