Redakcja Elektroniki Praktycznej dziękuje firmie NDN z Warszawy, autoryzowanemu dystrybutorowi i serwisowi firmy Rigol, za udostępnienie oscyloskopu Rigol DS2202 dla potrzeb tego artykułu.

Analiza protokołów (1)

Termin „analiza protokołów" znany jest elektronikom nie od dziś. Specjaliści różnych dziedzin interpretują go jednak zgoła odmiennie. Inżynier telekomunikacji zajmujący się systemami łączności radiowej czy telefonii komórkowej analizę protokołów będzie rozumiał inaczej niż informatyk, a jeszcze inaczej konstruktor projektujący układy elektroniczne. W artykule zajmiemy się protokołami wykorzystywanymi w popularnych interfejsach komunikacyjnych.

Do redakcji naszego pisma przychodzi wiele pytań dotyczących analizy protokołów. Czytelnicy coraz częściej spotykają się z tym terminem w artykułach drukowanych i umieszczanych w Internecie. Temat ten jest jednak poruszany tylko przy okazji recenzji oscyloskopów i analizatorów stanów logicznych. Postanowiliśmy więc pomóc w rozwikłaniu ewentualnych wątpliwości publikując krótki cykl artykułów na ten temat.

Definicja

Już z informacji podanych we wstępie do artykułu wynika, że termin „analiza protokołów" ma wiele znaczeń. Nas będą interesowały protokoły komunikacyjne stosowane na najniższym poziomie systemów elektronicznych, a więc wykorzystywane do łączności między poszczególnymi blokami funkcjonalnymi urządzenia lub bezpośrednio współpracujących ze sobą urządzeń. Bloki te muszą oczywiście charakteryzować się odpowiednią inteligencją, którą zapewniają mikroprocesory, mikrokontrolery, układy programowalne itp.

Szereg protokołów, o których mowa opracowano z uwzględnieniem charakterystycznych cech urządzeń, dla których są dedykowane, inne zoptymalizowano ze względu na media wykorzystywane do łączności lub charakterystyczną zawartość przesyłanych informacji (danych).

Liczba obecnie stosowanych protokołów jest dość duża, ale trzeba pamiętać, że większość z nich powstała w ostatnich 30 latach, a nawet później. Przykładem niech będzie interfejs CAN, którego początki sięgają lat osiemdziesiątych XX wieku. Interfejs ten powstał w wyniku zapotrzebowania na rozwiązania elektroniczne w nowoczesnych samochodach tamtego okresu, a było to wówczas możliwe ze względu na dostateczny rozwój technologii (przede wszystkim podzespołów elektronicznych takich jak mikroprocesory, miniaturowe czujniki wielkości nieelektrycznych itp.). Innym przykładem może być interfejs (i protokół) ARINC 429 wykorzystywany dla odmiany w lotnictwie. W obu przypadkach stosowane są inne rozwiązania układowe, inaczej są zbudowane interfejsy i w odmienny sposób zorganizowano transmisję danych, czyli przyjęto różne protokoły. Jest tak, mimo zbliżonych funkcji realizowanych przez oba interfejsy. Podstawowym zadaniem jest przecież zbieranie danych z licznych czujników rozmieszczonych w wielu punktach samochodu lub samolotu i przesyłanie ich do komputera pokładowego. Ten z kolei na podstawie odebranych informacji wypracowuje sygnały dla elementów wykonawczych (poduszki powietrzne, systemy wspomagania, systemy kontroli trakcji w samochodach czy wyświetlacz komputera pokładowego w samolocie). Można wymienić jeszcze szereg podobnych przykładów, ale w dalszej części artykułu skupimy się na czterech najczęściej stosowanych protokołach stosowanych w interfejsach komunikacyjnych o tej samej nazwie. Będą to:

- **RS-232** – historycznie jeden z pierwszych protokołów komunikacyjnych. Był on wykorzystywany przez szereg lat niemal we wszystkich urządzeniach elektronicznych, niezależnie od ich funkcji i zastosowań (urządzenia przemysłowe, sprzęt wojskowy, medyczny itp.). Jego popularność gwałtownie spadła po opublikowaniu specyfikacji interfejsu USB. Dzisiaj USB zastąpił niemal całkowicie wysłużonego RS-232, ale ze względu na dużą liczbę sprawdzonych i nadal działających aplikacji, RS-232 jest nadal instalowany w wielu urządzeniach.
- **I²C** – protokół opracowany przede wszystkim do komunikacji między mikrokontrolerem a rozmaitymi czujnikami, np. wielkości nieelektrycznych, pamięciami EEPROM itp. Częściowo rozwiązywał on problem ograniczonej szybkości transmisji w RS-232. Inną zaletą protokołu I²C jest wbudowany mechanizm adresowania urządzeń, co znacznie ułatwia selektywną komunikację z wieloma nadajnikami i odbiornikami. Ze względu na prawa licencyjne, spotykane są inne nazwy interfejsu I²C (np. TWI, IIC), jednak w każdym przypadku stosowany jest identyczny protokół.
- **SPI –** obecnie jeden z częściej stosowanych interfejsów komunikacyjnych. Pełni on funkcje zbliżone do omawianych wcześniej RS-232, i I²C. W interfejsie SPI, podobnie jak w I²C, zastosowano transmisję synchroniczną, co wymagało dołożenia dodatkowej linii dla przebiegu taktującego przesyłanie danych.
- **Parallel** – w tym przypadku protokół należy utożsamiać w zasadzie bezpośrednio z samym interfejsem. Dane są przesyłane równolegle przez wieloprzewodowe szyny danych/adresowych. Mimo oczekiwanych dużych szybkości transferu – w jednym takcie przesyłane są całe słowa, interfejs ten nie jest jednak stosowany zbyt często, głównie ze względu na znaczną komplikację połączeń i generowanie sporych zakłóceń.

Pod pojęciem „analiza protokołu” będziemy rozumieli badanie transmisji realizowanej przez jeden z wymienionych interfejsów. Wynikiem takiego pomiaru będzie graficzna prezentacja wysyłanych danych (w postaci oscylogramów) uzupełniona o wyświetlenie zdekodowanych danych. W zależności od charakteru przesłanych informacji można wybrać formę prezentacji danych, np. ASCII, liczby binarne, dziesiętne lub szesnastkowe. Obok tych informacji mogą być ponadto wyświetlane informacje dodatkowe, np. znacznik początku i końca pakietu, znaczniki błędów, adresy urządzeń itp.

Narzędzia diagnostyczne

W naszym kursie ograniczyliśmy się do protokołów wymienionych wcześniej. Przyjęliśmy, że wszystkie pomiary będą wykonywane oscyloskopem Rigol DS2202 z zainstalowaną opcją analizy protokołów. Zastosowana w nim funkcja pomiarowa „Decode” uwzględnia właśnie wymienione protokoły. Niestety, w zestawie brakuje np. USB czy 1-wire. Należy ponadto liczyć się z różnymi rozwiązaniami, które można spotkać u poszczególnych producentów. W oscyloskopach wyższej klasy mamy prawo spodziewać się szerszej gamy obsługiwanych protokołów. Warto również pamiętać, że funkcja analizy protokołów jest też na ogół dostępna w analizatorach stanów logicznych. W każdym przypadku należy upewniać się, czy jest ona instalowana domyślnie czy tylko opcjonalnie za dodatkową opłatą. Pojawiają się już wprawdzie pierwsze przypadki standardowego implementowania analizy przynajmniej najbardziej popularnych protokołów, ale proces ten przebiega z dziwnie dużymi oporami. Miejmy nadzieję, że wkrótce, kupując niemal każdy oscyloskop cyfrowy, od razu będziemy mieli możliwość analizy przynajmniej kilku, np. RS-232 (UART), SPI, I²C, Parallel. Podobną sytuację mieliśmy z analizą widmową FFT. Kiedyś był to rarytas, dzisiaj funkcję tę spotykamy w każdym oscyloskopie cyfrowym.

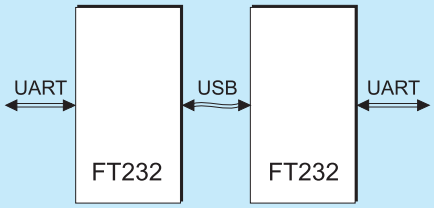
Wirtualne wcielenie analizatora protokołów

Jak mogliśmy się już przekonać, urządzenie o nazwie analizator protokołów w zasadzie nie istnieje. Używając tego terminu zawsze więc będziemy mieli na myśli określoną funkcjonalność na przykład oscyloskopu cyfrowego.

Protokół RS-232 (UART)

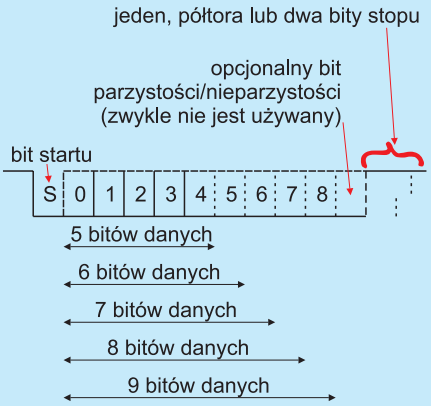
Po niezbędnym wstępie przystępujemy do zajęć praktycznych. Zaczynamy od protokołu RS-232, najczęściej chodzi o jego specyfikację RS-232C. W dostępnych materiałach Czytelnicy mogą spotkać się również z nazwą UART. W obu przypadkach chodzi o ten sam protokół (w sensie ramki transmisyjnej), jednak w uarcie rozpatrywanym jako interfejs mamy do czynienia z inną logiką sygnałów. Poziomy logiczne interfejsu UART odpowiadają technologii zastosowanych układów cyfrowych: TTL, TTL-LS, CMOS, TTL-LVC (3,3 V) itp. Na przykład w technologii TTL-LVC „1” logicznej odpowiada napięcie 3,3 V, a „0” odpowiada 0 V. Jest to bardzo istotne, gdyż w specyfikacji RS-232 przyjęto, że stanowi „1” odpowiada napięcie z zakresu od –3...–15 V, a stanowi „0” napięcie 3...15 V.

Mimo popularności wspomnianego wcześniej interfejsu USB, elektronicy chętnie omijają go stosując popularne mostki USB-UART, np. produkcji FTDI. Pozwala to wykorzystywać procedury transmisyjne identyczne jak dla interfejsu RS-232 (UART). Badając urządzenia z takimi rozwiązaniami zwykle nie ma potrzeby wnikania w transmisję od trony USB, wystarczy analiza protokołu po stronie UART (**rysunek 1**). Oscyloskop Rigol DS2202 jest wówczas wystarczającym narzędziem diagnostycznym.



Rysunek 1. Transmisja danych pomiędzy dwoma urządzeniami kablem USB z zastosowaniem protokołu UART

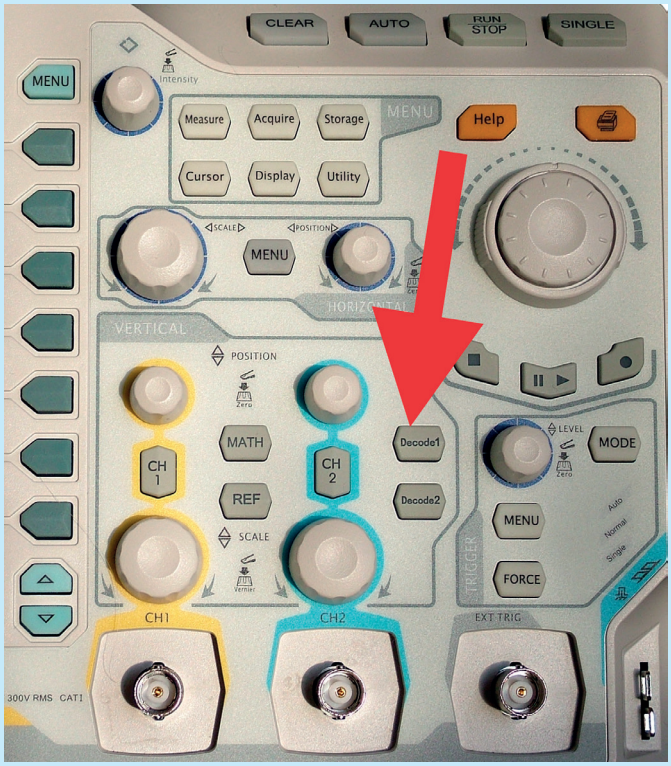
Teraz już na pewno możemy rozpocząć pomiary. Ramkę protokołu RS-232 przedstawiono na **rysunku 2**. Jak widać, mamy do czynienia z transmisją asynchroniczną, w której każdy wysyłany znak rozpoczyna się bitem startu. Następnie wysyłane są bity danych. Protokół przewiduje ramkę z 5, 6, 7, 8 lub 9 bitami. Najczęściej stosowana jest ramka 8-bitowa. W protokole przewidziano bit parzystości/nieparzystości, który jest wysyłany po bitach danych. Najczęściej jednak urządzenia z niego nie korzystają, w ramce po prostu go nie ma. Obowiązkowo natomiast muszą być wysłane bity stopu. Może być 1 bit, półtora bitu lub 2 bity. Zwykle korzystamy z jednego bitu stopu. Zaraz po zakończeniu wysyłania znaku (jego ostatniego bitu stopu) można rozpocząć wysyłanie następnego znaku. W nadajniku i w odbiorniku muszą być zastosowane wewnętrzne generatory taktujące o jednakowej częstotliwości. Ewentualna rozbieżność (w pewnym dopuszczalnym zakresie) jest niwelowana przez dosynchronizowanie się odbiornika na bitach startu. Warunkiem jest jednak prawidłowe rozpoznanie pierwszego bitu w transmisji, gdyż później bity startu niczym nie różnią się od pozostałych bitów i może nastąpić zerwanie komunikacji. Ważnym parametrem takiej komunikacji jest prędkość transmisji. Musi jej przestrzegać zarówno nadajnik, jak i odbiornik.



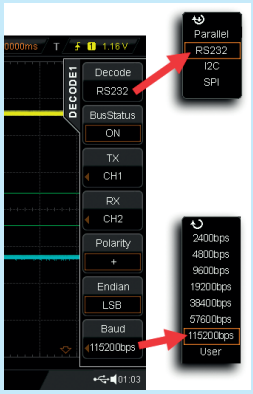
Rysunek 2. Ramka protokołu RS-232

Każde urządzenie dysponuje nadajnikiem (linia TxD) i odbiornikiem (linia RxD), co umożliwia transmisję dwukierunkową, więc obie linie muszą być skrzyżowane: TxD urządzenia 1 na RxD urządzenia 2 i analogicznie linie RxD na TxD.

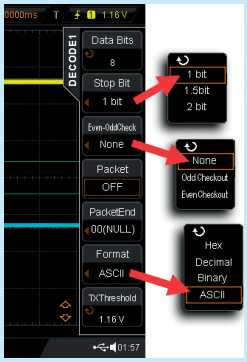
Powyższe informacje są wystarczające do odpowiedniego skonfigurowania obu urządzeń, będą też potrzebne do wprowadzenia prawidłowych nastaw analizatora protokołu. Przyjmijmy więc, że nasza ramka jest typu 115200,8,n,1, co oznacza, że dane są wysyłane z szybkością 115200 bitów na sekundę (to nie jest szybkość przesyłania znaków!), dana ma długość 8 bitów, nie korzystamy z bitu parzystości (gdyby taki bit był zastosowany zamiast „n” byłaby literka „o” – dla nieparzystości lub „e” – dla parzystości). Bit parzystości był wykorzystywany do kontroli prawidłowości transmisji, bardzo zawodnej zresztą. Specjalizowane układy odbiorników miały zain-



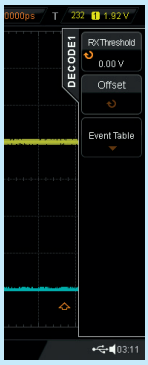
Fotografia 3. Lokalizacja przycisku *Decode1* na płycie czołowej oscyloskopu DS2202



Rysunek 4. Menu DECODE1 (pierwsza strona)



Rysunek 5. Menu DECODE1 (druga strona)



Rysunek 6. Menu DECODE1 (trzecia strona)

stalowany sprzętowy układ zliczający modulo 2 jedynki w przesyłanym znaku. Jeśli tak otrzymany wynik nie zgadzał się z bitem parzystości, ustawiana była odpowiednia flaga błędu, mogło być również generowane przerwanie. Dzisiaj opcja ta jest w zasadzie już tylko historyczną pozostałością i nikt z niej nie korzysta.

Nasz analizator protokołów będzie wykorzystywał oba kanały oscyloskopu, które dołączymy do linii TxD i RxD. *De facto* będziemy mierzyć interfejs UART z 3,3-woltową logiką. Pomiar rozpoczynamy od włączenia funkcji analizatora protokołów (przycisk *Decode1* (**fotografia 3**), – zauważmy, że możliwa jest niezależna analiza dwóch urządzeń). Na ekranie zostaje wyświetlone menu „DECODE1", w którym z łatwością odnajdujemy pola, odpowiadające parametrom transmisji (**rysunek 4**). Idąc od góry są to:

- typ protokołu –> wybieramy *RS232*,
- *BusStatus* – wybieramy *ON*, co powoduje włączenie dekodera odpowiedniego protokołu. Odczytane informację są wyświetlane na ekranie z zaznaczeniem linii, których dotyczą (np. TxD lub RxD),
- *TX* – wybieramy *CH1*, co oznacza, że kanał 1 oscyloskopu będzie dołączony do linii TxD,
- *RX* – wybieramy *CH2*, analogicznie dla linii *Polarity* – wybieramy +, gdyż badamy interfejs UART, pracujący w logice dodatniej. Gdyby to był RS-232, należałoby wybrać –,
- *Endian* – wybieramy *LSB*. W przypadku protokołu RS-232 w zasadzie nie mamy tu manewru, gdyż w ramce danych zawsze jako pierwsze są wysyłane bity najmniej znaczące,
- *Baud* – wybieramy *115200bps*, gdyż z taką właśnie szybkością pracuje badane urządzenie. Oprócz kilku wartości standardowych użytkownik może tu wprowadzać własną szybkość transmisji,

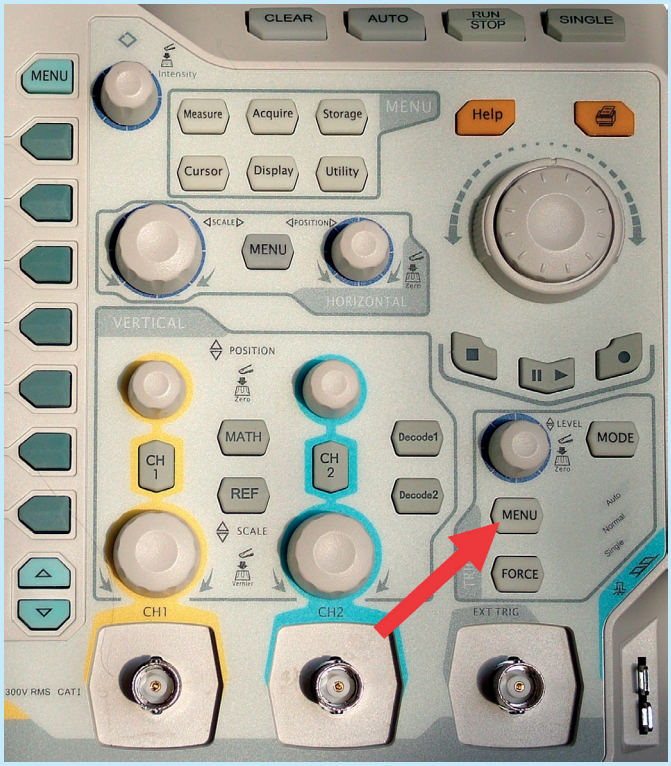
Teraz należy nacisnąć przycisk przejścia do następnej strony menu, czyli strzałkę w dół umieszczoną pod przyciskami funkcyjnymi obok ekranu. Rozwijają się kolejne opcje menu (**rysunek 5**).

- *Data Bits* – wybieramy *8*, gdyż pracujemy z ramką 8-bitową,
- *Stop Bit* – wybieramy *1*, bo w ramce przyjęliśmy jeden bit stopu,
- *Even-OddCheck* – wybieramy *None*, gdyż nie korzystamy z kontroli parzystości,
- *Packet* – na razie pomijamy tę opcję, wybieramy *OFF*,
- *PackeEND* – również pomijamy, nastawa nie ma znaczenia, jeśli wcześniej wybrano OFF. Opcje te będą omówione w dalszej części artykułu,
- *Format* – ponieważ w transmisji będą przesyłane informacje tekstowe wybieramy *ASCII*,
- *TXTherhold* – ten parametr określa poziom progowy dla zastosowanej logiki układów cyfrowych. Zwykle jest on ustawiany w połowie zakresu napięcia wyjściowego. Naciśnięcie przycisku umieszczonego przy opcji *TXTherhold* powoduje wyświetlenie poziomej linii na przebiegu związanym z linią TxD przedstawiającej aktualny poziom progowy. Wyświetlona jest także liczbowa wartość tego napięcia. Parametr *TXTherhold* jest zmieniany wielofunkcyjnym pokrętłem znajdującym się w górnym lewym rogu panelu zawierającego elementy regulacyjne.

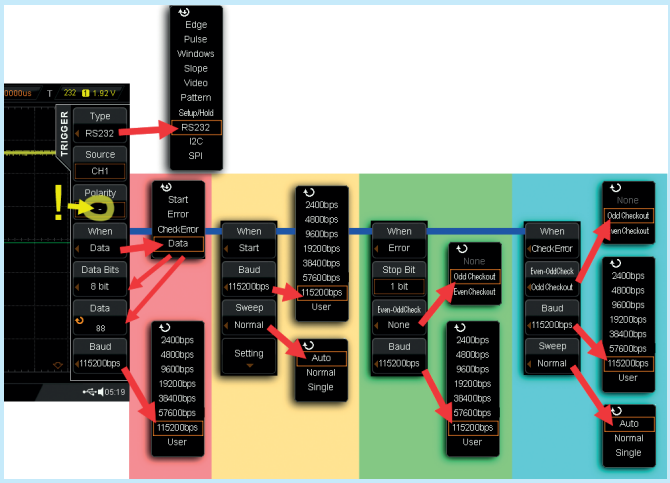
Przechodzimy do kolejnego menu, tak jak robiliśmy to wcześniej (**rysunek 6**).

- *RXTherhold* – wykonujemy analogiczne operacje dla linii RxD, tak jak robiliśmy to dla TxD,
- *Offset* – opcja ta pozwala dogodnie spozycjonować zdekodowane informacje. Korzystamy z pokrętła wielofunkcyjnego,
- *Event Table* – na razie pomijamy tę opcję.

Po wykonaniu wszystkich czynności, na ekranie powinien pojawić się oscylogram zawierający przebiegi na liniach TxD i RxD wraz ze zdekodowanymi danymi. Przebiegi będą jednak prawdopodobnie dość przypadkowo przesuwały się poziomo po ekranie, gdyż nie wprowadziliśmy jeszcze odpowiedniego warunku wyzwalającego. Procedura jest bardzo podobna jak podczas ustawianiu parametrów transmisji, z tym że teraz należy wybrać menu ukryte pod przyciskiem *Menu* w sekcji wyzwalania (**fotografia 7, rysunek 8**). Widzimy znajome opcje, którym powinny być nadane wartości zgodne z wprowadzonymi wcześniej parametrami. Jest jednak pewien niejasny wyjątek. Otóż w przypadku opcji *Polari-*



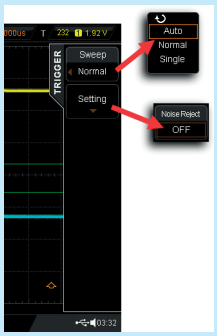
Fotografia 7. Lokalizacja przycisku *Menu* wybierającego opcje wyzwalania



Rysunek 8. Menu TRIGGER zawierające opcje wyzwalania (pierwsza strona)

ty definiowanej w menu warunku wyzwalania, trzeba wprowadzić wartość „–” (poprzednio był +). Prawdopodobnie jest to jakiś błąd w oprogramowaniu oscyloskopu DS2202. W omawianym menu pojawiła się dodatkowa pozycja:

- *When* – zawierająca opcje *Start*, *Error*, *CheckError* i *Data*. Ich znaczenie jest dość oczywiste. Po wybraniu opcji *Start*, układ wyzwalania współpracując z dekoderem transmisji będzie poszukiwał bitu startu i na nim nastąpi wyzwolenie. Problem polega na tym, że takich bitów zwykle jest dużo, rzadko kiedy wysyłamy jeden znak. Przy takiej nastawie trudno więc będzie uzyskać stabilny oscylogram. Oscyloskop będzie się synchronizował do każdego kolejnego napotykanego bitu startu. Opcja ta jest przydatna przy transmisji krótki paczek danych.



Rysunek 9. Menu TRIGGER (druga strona) z opcją *Sweep* ustalającą tryb pracy układu wyzwalania

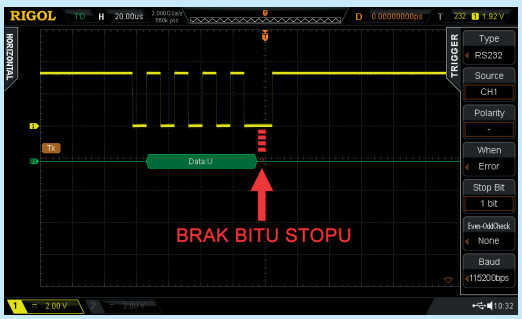
Przydatnym warunkiem wyzwalania jest natomiast ustawianie pułapki na określoną daną. W ten sposób łatwo można sprawdzać, czy dana taka pojawia się w ogóle w transmisji. Poszukiwana dana jest wprowadzania w postaci dziesiętnej w dolnym polu *Data* menu wyzwalania. Niestety, w oscyloskopie DS2202 przewidziano tylko taką postać danej, jeśli poszukujemy na przykład znaku ASCII „R”, to należy dokonać ręcznego przeliczenia, korzystając z tablicy kodów ASCII. W naszym przypadku znak „R” ma wartość dziesiętną 82, i taką liczbę należy wprowadzić, korzystając z pokrętła wielofunkcyjnego. Wybierając taki warunek wyzwalania, szczególnie wtedy, gdy poszukujemy specyficznego, rzadko występującego w transmisji zdarzenia, korzystne jest włączenie trybu wyzwalania „Normal”. Opcja ta jest dostępna po przejściu do kolejnego okna menu wyzwalania (**rysunek 9**). W trybie „Normal” wychwycone zdarzenie zostaje stabilnie zatrzaśnięte na ekranie. Wprawdzie w trybie „Auto” następowałoby także wyzwalanie na określonym zdarzeniu, ale przy dłuższym braku kolejnego takiego zdarzenia oscyloskop byłby wyzwalany bez spełnienia warunku, skutkujące zerwaniem synchronizacji.

Zastosowanie opcji *Error* powoduje wyzwalanie oscyloskopu po wykryciu błędu ramki, czyli braku bitu stopu. Sytuację taką przedstawiono na **rysunku 10.** Jednocześnie czerwony znacznik wyświetlony na ekranie sygnalizuje miejsce, w którym wystąpił błąd. Na oscylogramie widzimy, że tam gdzie powinien być znak stopu o wartości „1”, jest „0”.

Analogicznie działa opcja *CheckError*. Jest ona jednak stosowana tylko wtedy, gdy transmisja przebiega z włączoną kontrolą parzystości. Należy pamiętać o odpowiednim ustawieniu parametrów pracy analizatora protokołów.

Zakończyliśmy konfigurowanie naszego oscyloskopu do pracy jako analizatora protokołu RS-232. Możemy przystąpić do pomiarów, ale o tym już w następnym odcinku.

Jarosław Doliński, EP



Rysunek 10. Wyzwalanie po wykryciu błędu ramki

Redakcja Elektroniki Praktycznej dziękuje firmie NDN z Warszawy, autoryzowanemu dystrybutorowi i serwisowi firmy Rigol, za udostępnienie oscyloskopu Rigol DS2202 dla potrzeb tego artykułu.

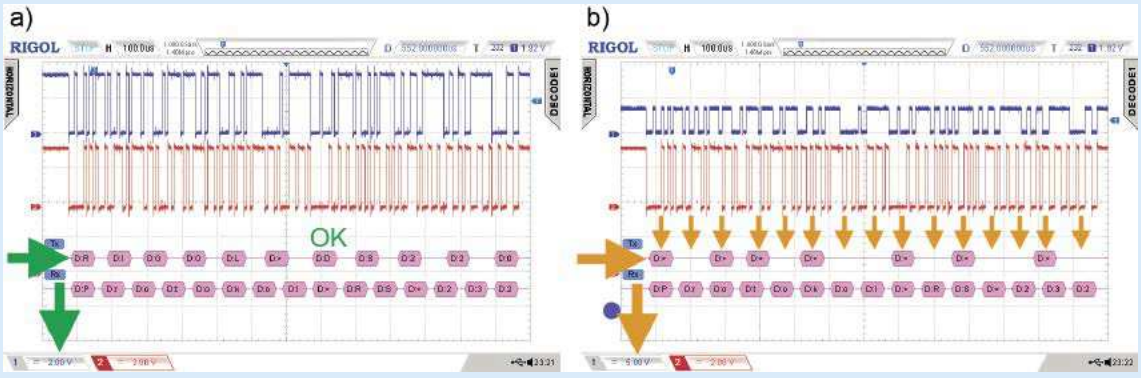
Analiza protokołów (2)

Analizowanie pracy interfejsu RS-232/UART

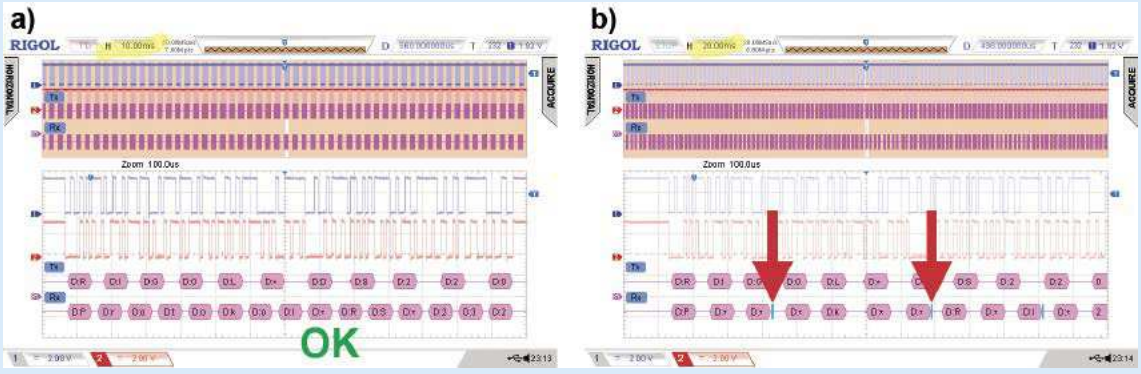
Termin „analiza protokołów" znany jest elektronikom nie od dziś. Specjaliści różnych dziedzin interpretują go jednak zgoła odmiennie. Inżynier telekomunikacji zajmujący się systemami łączności radiowej czy telefonii komórkowej analizę protokołów będzie rozumiał inaczej niż informatyk, a jeszcze inaczej konstruktor projektujący układy elektroniczne. W artykule zajmiemy się protokołami wykorzystywanymi w popularnych interfejsach komunikacyjnych.

Kontynuujemy pomiary interfejsu RS-232 (UART) korzystając z funkcji analizatora protokołów oscyloskopu Rigol DS2202. Wiemy już, że do uzyskania stabilnego oscylogramu ważne jest ustawienie odpowiedniego warunku wyzwalania. Dla uchwycenia określonej sytuacji prawie zawsze będziemy korzystać z wyzwalania pojedynczego („Single") lub „Normal" (jeśli poszukiwana sekwencja powtarza się co jakiś czas). Przy dobieraniu nastaw oscyloskopu Rigol DS2202 pracującego jako analizator protokołów należy uwzględniać kilka ważnych zagadnień. Jednym z nich jest prawidłowe ustawienie progów napięciowych i czułości kanałów pomiarowych. Chociaż pozornie parametry te nie są ze sobą powiązane, może zdarzyć się, że niedokładne ustawienie progów napięciowych „TxThreshold" i „RxThreshold" będzie przyczyną powstawania błędów ujawniających się przy mniejszych czułościach kanałów. Sytuację taką przedstawiono na **rysunku 11.** Napięcie progowe dla logiki 3,3 V zostało niedokładnie ustawione na 912 mV. Jak widać, przy czułości kanału równej 2 V/dz analizator dekodował dane poprawnie, ale po jej przełączeniu na 5 V/dz zaczęły pojawiać się błędy.

Błędy mogą powstawać również wtedy, gdy zostanie ustawiona zbyt wolna podstawa czasu. Skłania nas do tego względnie długi rekord akwizycji (28 Mpunktów przy włączonych dwóch kanałach). Przy wolnej podstawie czasu w rekordzie jest zapisywany duży



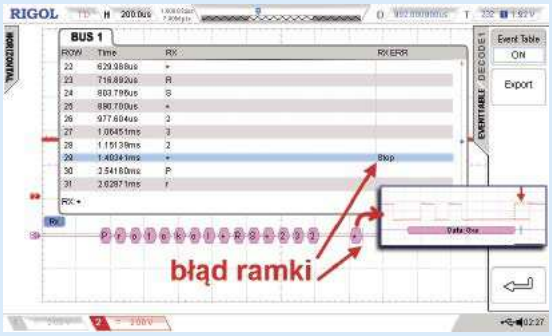
Rysunek 11. Wpływ czułości kanału pomiarowego na interpretację danych przy niedokładnie określonych progach napięciowych TxThreshold/RxThreshold



Rysunek 12. Błędy interpretacji danych wynikające ze zbyt wolnej podstawy czasu

fragment transmisji, a jego szczegóły mogą być później powiększane funkcją *Zoom*. Dla użytkownika jest to więc wygodny sposób pracy. Niestety, po wybraniu zbyt wolnej podstawy czasu, w wynikach pojawia się sygnalizacja błędów, które w rzeczywistości nie występują (**rysunek 12**). Co gorsze, w pierwszej chwili trudno stwierdzić, czy błąd wynika ze złego działania urządzenia czy źle dobranych nastaw oscyloskopu. Zawsze więc warto wykonać kilka pomiarów z różnymi nastawami, w szczególności zmieniając podstawę czasu, długość rekordu akwizycji i czułość kanałów.

Do dalszych rozważań zakładamy, że oscyloskop jest prawidłowo skonfigurowany. Skorzystamy teraz z tabelarycznej prezentacji zdekodowanych danych. Każdemu zdarzeniu występującemu w interfejsie przypisywany jest stempel czasowy zapisywany następnie w tabeli wraz ze zdekodowaną daną i ewentualnymi informacjami o błędach. Dzięki temu możliwe jest precyzyjne określanie relacji czasowych pomiędzy różnymi punktami transmisji. Punktem odniesienia jest moment wyzwolenia (**rysunek 13**). Tabele mogą być zapisywane w postaci plików CSV w pamięci zewnętrznej typu *pendrive* dołączanej do gniazda znajdującego się na płycie czołowej oscyloskopu.

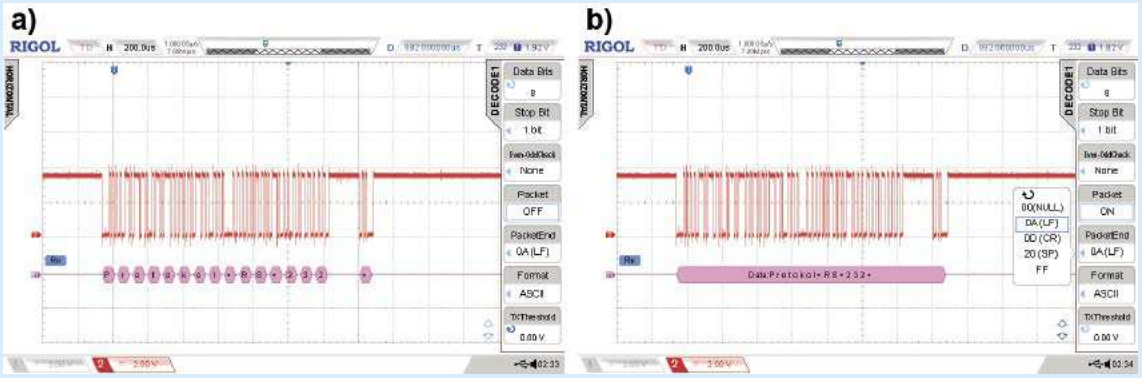


Rysunek 13. Tabelaryczna forma prezentacji zdekodowanych danych

Funkcja analizatora protokołów oscyloskopu DS2202 umożliwia także dekodowanie transmisji pakietowych. Pakiet jest złożony z ciągu danych zakończonych jednym z kilku zdefiniowanych znaczników. Są to: znak NULL (bajt o wartości 0), znak LF (0Ah), znak CR (0Dh), znak spacji (20h) lub znak FFh. W normalnym trybie, przy wyłączonym dekodowaniu pakietów, każdy znak jest interpretowany niezależnie, włącznie ze znacznikiem końca pakietu (**rysunek 14a**). Po włączeniu opcji dekodowania pakietowego wszystkie znaki oprócz znacznika końca pakietu są interpretowane jako jeden komunikat (**rysunek 14b**). Najlepiej jest to widoczne, gdy przesyłane są informacje tekstowe.

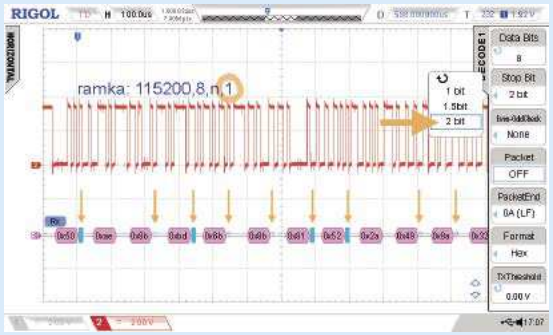
Nieprawidłowe ustawienie parametrów transmisji protokołu RS-232

Analizator protokołów zachowuje się tak samo, jak każdy odbiornik dołączony do danego interfejsu komunikacyjnego. Oczywiste jest zatem, że parametry jego pracy powinny być zgodne z przyjętymi w badanym urządzeniu. Niektóre błędy w nastawach są zauważalne natychmiast po wyświetleniu wyników na ekranie, ale są też takie, które nie dają jednoznacznej informacji. Przykładem może być nieprawidłowo określona kolejność nadawanych bitów (parametr „Endian"). W interfejsie RS-232 dane są wysyłane zwykle od najmłodszego do najstarszego bitu. Jeśli w konfiguracji analizatora zostanie wybrana opcja „Endian"= MSB, to oczywiście procedura dekodująca będzie działała zgodnie z tą regułą – bity będą interpretowane od najstarszego do najmłodszego. Dla analizatora będzie to jednak sytuacja normalna, żadne błędy nie zostaną



Rysunek 14. Interpretacja danych jako: a) pojedyncze znaki, b) pakiety

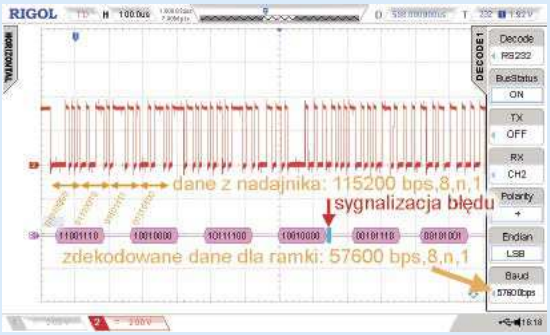
Rysunek 15. Błędy interpretacji danych wynikające z nieprawidłowo ustawionego parametru „Endian"



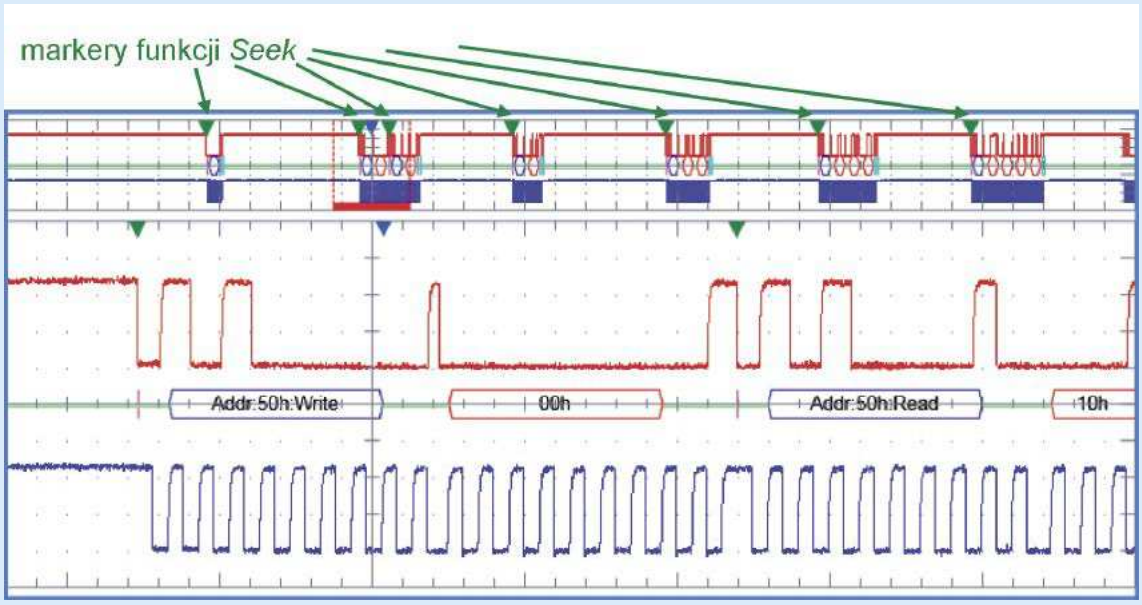
Rysunek 16. Błędy interpretacji danych wynikające z nieprawidłowo ustawionego parametru „Stop Bit"

zasygnalizowane. Przypadek taki przedstawiono na **rysunku 15.**

Innym parametrem, który przy błędnym wprowadzeniu będzie generował błędy dekodowania jest „Stop Bit", czyli liczba bitów stopu. Jak już było powiedziane, parametr ten ma najczęściej wartość 1, ale interfejsy dopuszczają także inne nastawy: 1,5 lub 2. Wprowadzenie np. „Stop Bit"=2 nie musi jednak oznaczać sygnalizacji błędów po każdym znaku. Zależy to od tego czy kolejne znaki są nadawane bezpośrednio jeden za drugim czy z niewielkimi przerwami. Może się zdarzyć, szczególnie przy dużych szybkościach transmisji, że procesor nie będzie nadążał z przygotowaniem kolejnych danych



Rysunek 17. Błędy interpretacji danych wynikające z nieprawidłowo ustawionej szybkości transmisji („Baud")



Rysunek 18. Wynik przeszukiwania danych funkcją *Seek* dostępną w droższych oscyloskopach

do wysłania, a ponieważ w stanie oczekiwania na znak wyjście nadajnika jest w stanie wysokim, to będzie on interpretowany jako ewentualny bit stopu. Przy nastawie „Stop Bit"=2 sygnalizacja błędu nie wystąpi. Błędów po każdym znaku nie będzie także, gdy któryś z bitów danych zostanie zinterpretowany jako bit startu, a bit danych wysyłany w chwili odpowiadającej bitowi stopu będzie miał wartość „1". Przypadek taki jest dobrze widoczny na **rysunku 16** (drugi znak został tu zdekodowany bez sygnalizacji błędu, chociaż jego interpretacja jest oczywiście nieprawidłowa, gdyż jest on przesunięty względem oryginału).

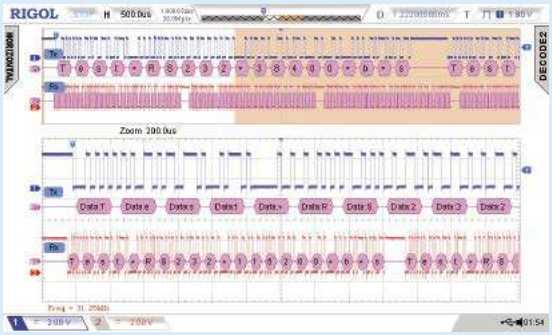
Podobnie może się zdarzyć, gdy w parametrach analizatora protokołów wprowadzimy nieprawidłową szybkość transmisji. Błędy na pewno pojawią się w zdekodowanych danych, ale tak jak poprzednio, nie przy każdym odebranym znaku. Interpretacja danych będzie jednak nieprawidłowa (**rysunek 17**).

Powyższe przypadki przedstawiono, w celu zwrócenia uwagi na znaczenie odpowiedniej konfiguracji oscyloskopu wykorzystywanego jako analizator protokołów. Źle dobrane nastawy będą przyczyną generowania błędów, których źródło może być utożsamiane z badanym urządzeniem, a przecież właśnie do sprawdzania poprawności jego działania jest wykorzystywany analizator protokołów. Zawsze przed przystąpieniem do pomiarów warto dokładnie upewniać się czy wszystkie parametry protokołu zostały zdefiniowane prawidłowo.

Metodyka pracy z analizatorem protokołów

Po analizator protokołów sięgamy zazwyczaj wtedy, gdy musimy zlokalizować przyczynę błędnego działania jakiegoś interfejsu komunikacyjnego. Przyczyną takiego stanu mogą być usterki techniczne urządzenia, albo błędy oprogramowania. Zwykle więc poszukiwane będą albo określone dane, albo inne zdarzenia charakterystyczne dla wykorzystywanego

Rysunek 19. Wyzwalanie szerokością impulsu przy pracy z analizatorem protokołów



Rysunek 20. Badanie dwóch niezależnych interfejsów komunikacyjnych

protokołu komunikacyjnego. W droższych oscyloskopach, dysponujących dużymi rekordami, dostępne są bardzo rozbudowane funkcje przeszukiwania zdekodowanych danych. Dzięki nim użytkownik może zapisywać długie fragmenty transmisji w rekordzie akwizycji, a następnie, już w trybie off-line, lokalizować ewentualne nieprawidłowości. Wprowadzanie warunków poszukiwania odbywa się w sposób zbliżony do definiowania zdarzenia wyzwalającego. Wszystkie znalezione miejsca są zaznaczane markerami, a specjalne narzędzie przeszukiwania, np. *Wave Inspector* w oscyloskopach firmy Tektronix, pozwala szybko przemieszczać się między nimi (**rysunek 18**). Szczegóły są pokazywane w oknie *Zoom*.

W oscyloskopie DS2202 Rigola niestety nie ma takich funkcji, wyszukiwanie zdarzeń jest więc trudniejsze i polega przede wszystkim na odpowiednim definiowaniu warunku wyzwalania. Była o tym mowa w pierwszej części artykułu. Można jeszcze dodać, że korzystając z analizatora protokołów warunek wyzwalania nie musi być związany z tą funkcją. Należy szukać takich zdarzeń, które będą na tyle charakterystyczne dla transmisji, że pozwolą uzyskać stabilny oscylogram. Na przykład, jeśli badamy transmisję bloków danych wysyłanych z pewnymi odstępami czasu, korzystne może być ustawienie warunku wyzwalania impulsem o szerokości dobranej do przerw między blokami (**rysunek 19**).

Użytkownicy oscyloskopu DS2202 mogą badać dwa takie same (nie koniecznie te same) lub dwa różne interfejsy. Wynika to z dostępnych kanałów analizatora protokołów w tym przyrządzie. Z kolei w każdym interfejsie może występować kilka linii sygnałowych, tu jednak ograniczeniem jest liczba kanałów pomiarowych (dwa w oscyloskopie DS2202). Na **rysunku 20** przedstawiono badanie dwóch interfejsów RS232. Pierwszy kanał oscyloskopu dołączono do linii TxD, czyli do nadajnika pracującego z szybkością 38400 b/s. Drugi kanał wykorzystano do badania odbiornika interfejsu pracującego z szybkością 115200 b/s (linia RxD). Oba przebiegi są dla siebie asynchroniczne, więc na ekranie nie będą stabilnie wyświetlane. W zależności od potrzeb należy zdecydować czy wyzwalanie ma być określone dla kanału 1 czy 2.

W następnym odcinku omówimy badanie interfejsu SPI.

Jarosław Doliński, EP

Redakcja Elektroniki Praktycznej dziękuje firmie NDN z Warszawy, autoryzowanemu dystrybutorowi i serwisowi firmy Rigol, za udostępnienie oscyloskopu Rigol DS2202 dla potrzeb tego artykułu.

Analiza protokołów (3)

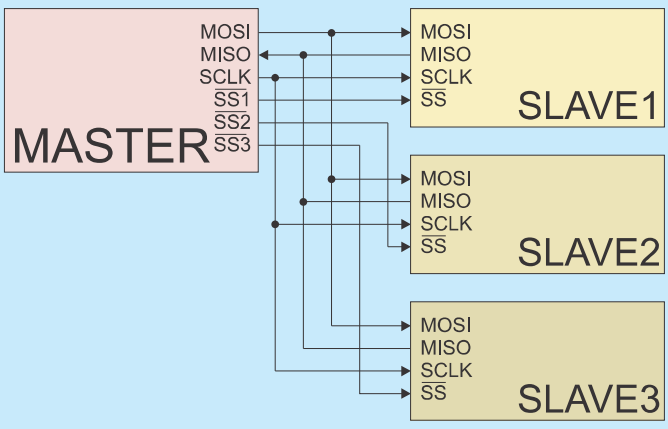
Analizowanie interfejsu SPI

Termin „analiza protokołów" znany jest elektronikom nie od dziś. Specjaliści różnych dziedzin interpretują go jednak zgoła odmiennie. Inżynier telekomunikacji zajmujący się systemami łączności radiowej czy telefonii komórkowej analizę protokołów będzie rozumiał inaczej niż informatyk, a jeszcze inaczej konstruktor projektujący układy elektroniczne. W artykule zajmiemy się protokołami wykorzystywanymi w popularnych interfejsach komunikacyjnych.

W poprzednich odcinkach poznaliśmy ogólną zasadę obsługi analizatora protokołów oraz metody badania protokołu RS-232/UART. Mieliśmy do czynienia z transmisją asynchroniczną, różniącą się zasadniczo od transmisji synchronicznej stosowanej w interfejsie SPI. W konsekwencji konieczne stało się zwiększenie linii interfejsu. Dodano więc linię SCLK, wykorzystywaną do przesyłania przebiegu zegarowego taktującego wszystkie nadajniki i odbiorniki, a także linię (SS) służącą do sprzętowego wybierania urządzenia podrzędnego – Slave, komunikującego się z urządzeniem nadrzędnym – Master. Dane pojawiają się wprawdzie jednocześnie na wejściach wszystkich odbiorników, ale analizuje je tylko ten Slave, który został wybrany sygnałem SS. Biorąc pod uwagę zależności między poszczególnymi sygnałami można wyróżnić cztery tryby pracy interfejsu, którym musimy się bliżej przyjrzeć.

Tryby pracy interfejsu SPI

Urządzeń Slave może być w systemie kilka, ale w danej chwili Master prowadzi komunikację tylko z jednym, wybierając go podaniem niskiego poziomu na linię SS. Jeżeli Master ma komunikować się z kilkoma urządzeniami, musi mieć odpowiednią liczbę wyjść sterujących SS_i. Program dla Mastera musi być napisany tak, aby podczas transmisji, w danej chwili wystawiać poziom aktywny tylko na jednym wyjściu SS_i. Po przesłaniu danych wszystkie linie sterujące powinny być ustawione w poziom nieaktywny (wysoki). Przykładowe połączenia urządzeń komunikujących się ze sobą przez interfejs SPI przedstawiono na **rysunku 21.** Jak widać z schematu, koncepcja interfejsu SPI nie przewiduje bezpośredniego komunikowania się ze sobą urządzeń Slave. Jeśli wymaga tego aplikacja, musi być stworzony odpowiedni protokół realizujący sekwencję połączeń np. Slave1–>Master, Master–>Slave2.



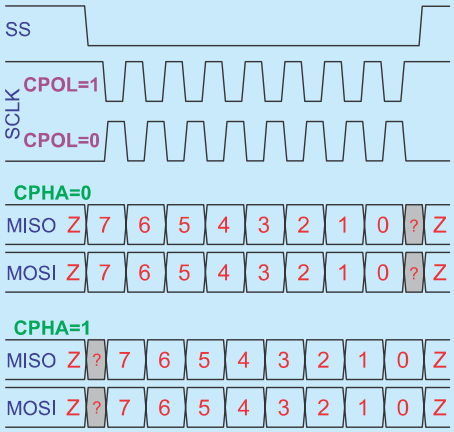
Rysunek 21. Przykładowe połączenia urządzeń komunikujących się ze sobą przez interfejs SPI

Tabela 1. Tryby pracy interfejsu SPI

Tryb	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Wróćmy do budowy interfejsu SPI. Zawiera on co najmniej 4 linie. Dwie z nich już znamy. Są to: SCLK (Serial Clock) – przebieg zegarowy taktujący transmisją i SS (Slave Select) – sygnał wyboru urządzenia Slave (poziomem niskim). Dane są przesyłane liniami: MOSI (Master Output Slave Input) w kierunku od urządzenia Master do Slave oraz MISO (Master Input Slave Output) od urządzenia Slave do Master. Należy zwrócić uwagę na to, że w wyniku przyjęcia takiej konwencji wszystkie wyprowadzenia MOSI są łączone ze sobą, podobnie jest z wyprowadzeniami MISO oraz SCLK. W interfejsie RS-232 wyjście nadajnika TxD należało łączyć krzyżowo z wejściem odbiornika RxD.

Zanim zaczniemy pomiary należy jeszcze rozważyć pewien problem techniczny związany z taktowaniem danych na liniach MOSI i MISO. Interfejs SPI może pracować w kilku trybach definiowanych na podstawie polaryzacji i fazy przebiegu zegarowego. Możliwe warianty są definiowane umownymi parametrami CPOL i CPHA. Parametr CPOL decyduje o poziomie sygnału zegarowego SCLK, jaki jest utrzymywany podczas nieaktywności interfejsu, CPHA określa natomiast zbocza sygnału zegarowego, na których następuje zmiana poziomu na liniach MOSIO i MISO. Na przeciwnych zboczach odbiorniki odczytują dane



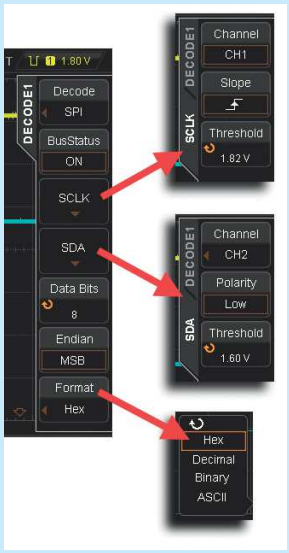
Rysunek 22. Możliwe tryby pracy interfejsu SPI

z tych linii. Możliwe kombinacje przedstawiono na **rysunku 22**. Odpowiadają im jednoznacznie określone tryby pracy interfejsu SPI, co wyjaśniono w **tabeli 1**.

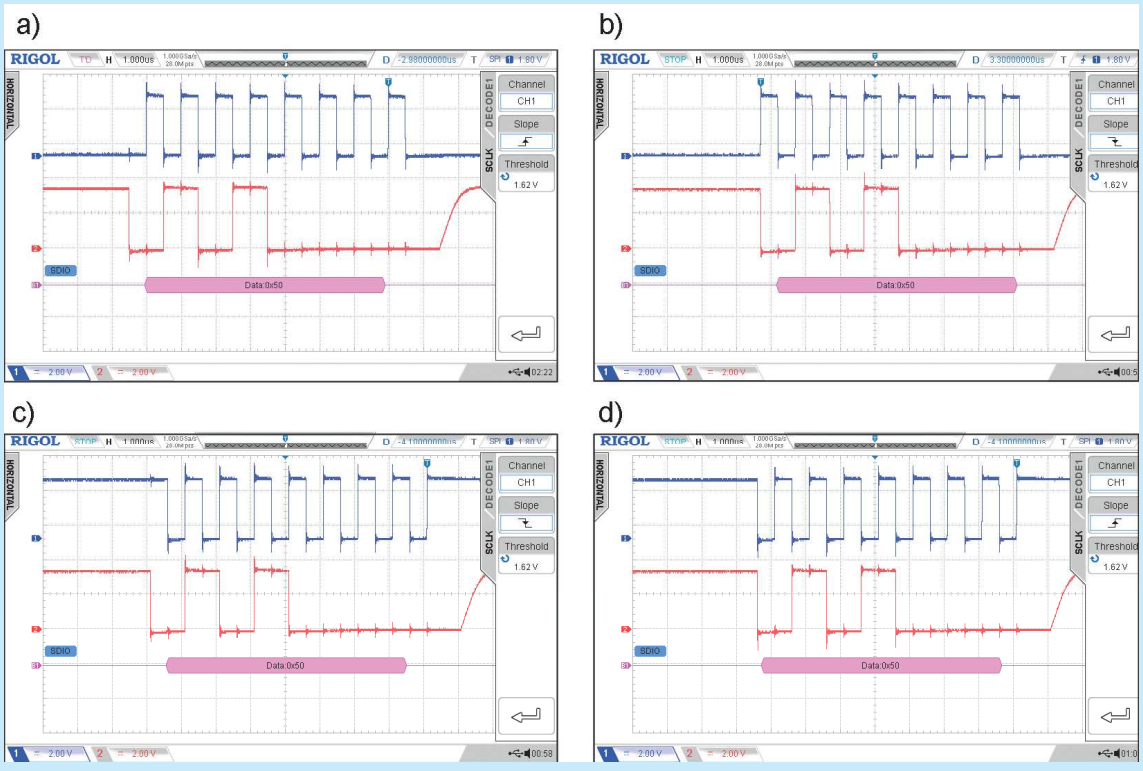
W protokole SPI przewidziano ramkę zawierającą od 5 do nawet 32 bitów (standardowo 8). Nie wykorzystuje się bitów parzystości. Z oczywistych powodów nie ma też bitów startu i stopu. W większości praktycznych zastosowań bity są wysyłane w kolejności od najstarszego do najmłodszego. Transmisja synchroniczna nie wymaga stosowania znormalizowanych szybkości przesyłania danych, jest ograniczona głównie możliwościami fizycznymi układów elektronicznych i mediów przesyłających sygnały. Częstotliwość przebiegu SCLK może dochodzić nawet do 100 MHz.

Analiza protokołu SPI

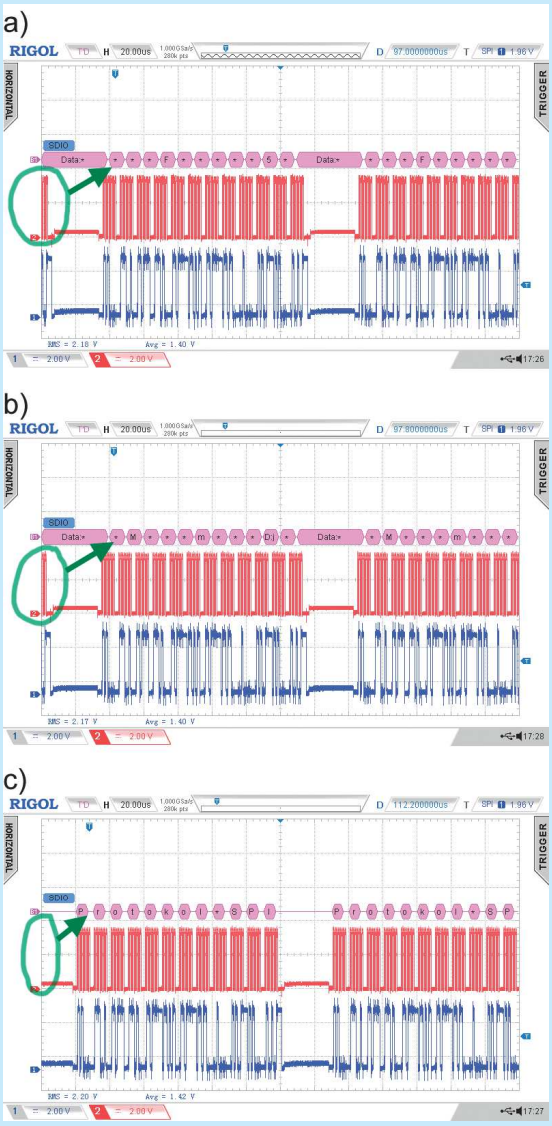
Podstawowym problemem związanym z analizą protokołu SPI za pomocą oscyloskopu Rigol DS2202 jest ograniczona do 2 liczba kanałów pomiarowych. Z tego względu zrezygnowano z badania linii SS. Analizator nie rozróżnia też linii komunikacyjnych MOSI i MISO, zamiast nich linię komunikacyjną nazwano ogólnie SDA (Serial Data). Menu konfigurowania analizatora protokołu SPI przedstawiono na **rysunku 23**. Kanały oscyloskopu nie są na sztywno przypisane do poszczególnych sygnałów. Po rozwinięciu opcji „SCLK” i „SDA” można dokonać odpowiedniego skojarzenia kanałów z sygnałami interfejsu. W tym podmenu definiowane są też dwa parametry decydujące o trybie pracy analizatora, przy czym nie mają one bezpośredniego związku z parametrami interfejsu CPOL i CPHA. W opcji „SCLK” widzimy parametr *Slope*, który wybiera zbocze sygnału zegarowego (narastające lub opadające), na którym następuje odczyt danej z linii SDA (czyli MOSI lub MISO) – rysunek 23a. Analizator nie musi mieć zdefiniowanej fazy CPHA, określana jest za to polaryzacja sygnału. Parametr ten ustawiamy w opcji „SDA” –> *Polarity* (*Low* lub *High*). „Polarity”=Low oznacza, że niski poziom napięcia odpowiada logicznemu zeru, natomiast wysoki poziom logicznej jedynce. Odwrotnie jest, gdy „Polarity”=High. Wtedy niski poziom napięcia odpowiada logicznej jedynce, a wysoki poziom logicznemu zeru. Ostatecznie będą nas właściwie interesować tylko przypadki, w których brany jest pod uwagę poziom



Rysunek 23. Menu konfigurowania analizatora protokołu SPI



Rysunek 24. Możliwe kombinacje konfiguracji protokołu SPI w oscyloskopie DS2202



Rysunek 25. Wpływ pozycjonowania oscylogramu w osi czasu na poprawność działania analizatora protokołów

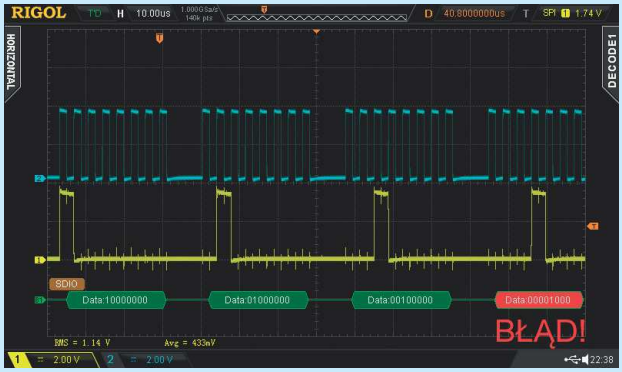
linii SCLK w czasie nieaktywności interfejsu i zboczе sygnału SCLK, na którym jest czytany znak w analizatorze protokołu. Wszystkie kombinacje przedstawiono na **rysunku 24**.

Przed przystąpieniem do pomiarów należy określić pozostałe parametry interfejsu SPI, a więc liczbę bitów danej („Data bits") oraz kolejność wysyłania bitów („Endian"). Zgodnie z ogólną zasadą, jeśli opcji „Endian" nadano wartość MSB, oznacza to, że bity są wysyłane od najstarszego do najmłodszego, natomiast dla „Endian"=LSB bity są wysyłane od najmłodszego do najstarszego. W większości przypadków będziemy wybierać opcję MSB.

Ostatnim parametrem jest „Format". Definiuje on sposób interpretowania zdekodowanych danych. Może to być zapis heksadecymalny, dziesiętny, binarny lub w postaci znaków ASCII.

Wyzwalanie SPI

Analizator mamy już prawie gotowy do pracy, brakuje tylko warunku wyzwolenia. Badając protokół SPI sięgamy naturalnie po opcje dostępne dla takiego pomiaru. W menu „Triger" wybieramy opcję *SPI*, a w niej



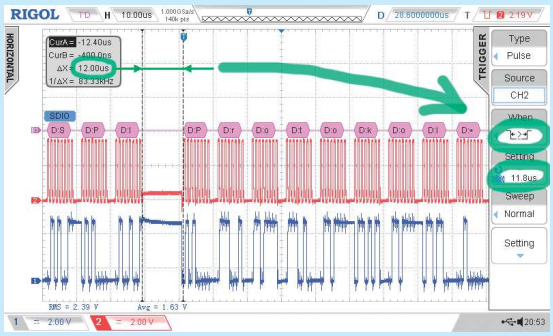
Rysunek 26. Błąd odczytu danej niemieszczącej się na ekranie

jedynym zdarzeniem, z którego możemy skorzystać jest wyzwolenie na określonej danej przesyłanej interfejsem. Definiowanie danej polega na odpowiednim ustawianiu kolejnych jej bitów, co jest metodą skuteczną, ale niezbyt wygodną. Należy zwrócić uwagę na to, że bity są numerowane w kolejności odwrotnej, do jakiej jesteśmy przyzwyczajeni – najstarszy jest bit numer 0, najmłodszy bit dla danej 8-bitowej ma numer 7. Konieczne jest jeszcze powtórne, niezależne od ustawień protokołu, ale zgodne z nim, przypisanie linii interfejsu SPI (SCLK i SDA) do kanałów oscyloskopu. Dla uzyskania stabilnego oscylogramu, w większości przypadków konieczne będzie przełączenie układu akwizycji w tryb Normal.

Po wykonaniu opisanych czynności spodziewamy się wyzwolenia oscyloskopu na zdefiniowanej danej i wyświetleniu stabilnego oscylogramu. Pojawiają się jednak dwie wątpliwości. Pierwsza dotyczy sytuacji, gdy w przesyłanym komunikacie występuje więcej niż jeden znak podany jako wzorzec dla układu wyzwalania. Istnieje niebezpieczeństwo wyzwalania na każdym z nich, i dlatego należy raczej szukać innej danej lub zupełnie innego warunku wyzwalania. Druga wątpliwość jest związana z ograniczoną liczbą kanałów. Zauważmy, że odbiornik analizatora protokołu SPI zaczyna dekodować daną po wystąpieniu 8 impulsów zegarowych (zakładamy, że z takimi ramkami mamy do czynienia). Prawdziwy odbiornik interfejsu SPI rozpoczyna zliczanie impulsów zegarowych zawsze po uaktywnieniu linii SS, której jednak analizator oscyloskopu w ogóle nie bada. Jak zatem pracuje odbiornik analizatora? Informacji takiej nie znajdziemy w instrukcji oscyloskopu, ale stosunkowo łatwo można zauważyć, że jako pierwszy jest liczony pierwszy impuls zapisany w rekordzie akwizycji. Bardzo prawdopodobne jest, że będzie to środkowy bit którejś z danych, a to oznacza, że kolejne dane będą interpretowane nieprawidłowo. Przypadki takie przedstawiono na **rysunku 25.** Jak widać, dopiero po odpowiednim przesunięciu oscylogramu w osi czasu, komunikat jest dekodowany prawidłowo (rysunek 25c). Przekłamaniu ulega też dana nie mieszcząca się w rekordzie. Wystarczy, że poza ekranem znajdzie się choćby jeden impuls zegarowy, aby dana została zdekodowana błędnie. Jest wówczas wyświetlana na czerwonym tle (**rysunek 26**).

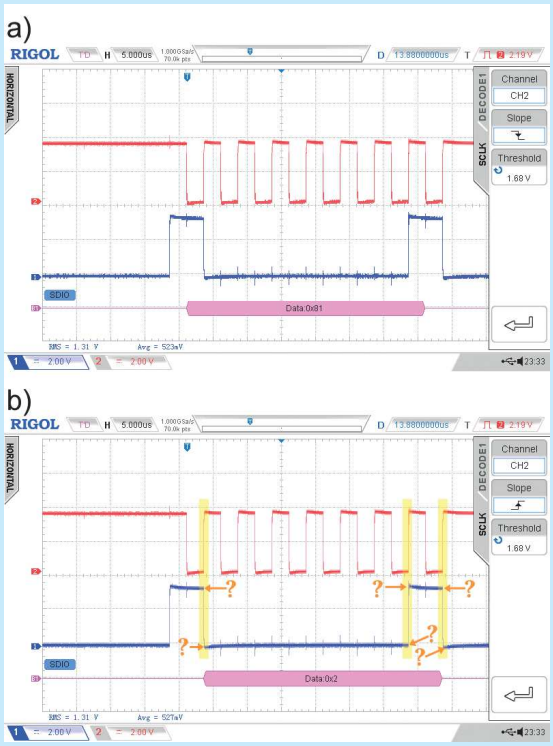
Ustalenie odpowiedniego warunku wyzwalania jest niezbędne do poprawnej pracy analizatora protokołów. Opcje wyzwalania przypisane do danego proto-

kołu nie zawsze są najlepszym wyborem, często równie dobre okazuje się po prostu wyzwalanie zboczem lub impulsem. Na przykład, jeśli dane są przesyłane w paczkach z pewnymi minimalnymi odstępami, bardzo dobre wyniki daje wyzwalanie impulsem o zadanej szerokości, tak jak to robiliśmy badając interfejs RS-232. Przykład takiego wyzwalania przedstawiono na **rysunku 27.**



Rysunek 27. Wyzwalanie analizatora protokołu SPI impulsem o określonej szerokości

Rysunek 28. Błąd interpretacji danych przy nieprawidłowym ustawieniu parametru „Endian”



Rysunek 29. Interpretacja danych dla różnych nastaw parametru „Slope” a) ustawienie prawidłowe, b) ustawienie błędne

Skutki błędnych nastaw

Ustawienie parametrów protokołu niezgodnych z parametrami badanego urządzenia powoduje oczywiście błędną interpretację odczytywanych przez analizator danych. Tak, jak w przypadku protokołu RS-232, nie zawsze możemy być świadomi takiego stanu. O nieprawidłowym wybraniu parametru „Endian” zorientujemy się natychmiast, jeśli będą transmitowane znaki ASCII – przesyłane informacje będą zupełnie nieczytelne. Jeśli jednak dane mają interpretację liczbową, natychmiastowe wykrycie błędu nie będzie oczywiste (**rysunek 28**).

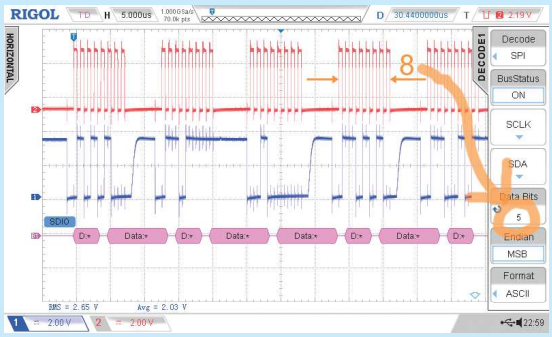
Błąd dekodowania nie zostanie również wyświetlony, gdy zostanie wybrane złe zbocze impulsu zegarowego. Na **rysunku 29** przedstawiono oscylogram uzyskany podczas badania interfejsu SPI pracującego w trybie 2 (CPOL=1, CPHA=0). Podczas nieaktywności interfejsu linia SCLK jest utrzymywana na poziomie logicznej „1”, zmiany poziomów linii danych następują na narastającym zboczu impulsów zegarowych, natomiast odczyt danych na zboczu opadającym. Opcja „Slope” powinna więc być ustawiona na zbocze opadające. Wybranie zbocza narastającego, tak jak na rysunku 29b, powoduje, że może być odczytany zarówno poziom bezpośrednio przed impulsem, jak też za nim. Decydują o tym wręcz warunki propagacji sygnału w danym urządzeniu. Przy odrobinie szczęścia może zdarzyć się, że odczyt będzie poprawny, nie mniej, mamy do czynienia z błędną konfiguracją analizatora.

Pozostała jeszcze reakcja na nieprawidłowo ustawioną długość ramki. Jeśli kolejne znaki są nadawane z małymi przerwami, wychwycenie tego błędu nie będzie trudne. Szybko zauważymy, że długość danych nie odpowiada paczkom impulsów zegarowych, poza tym dane będą dekodowane błędnie (**rysunek 30**). Gorzej, jeśli transmisja odbywa się bez przerw między kolejnymi znakami, co jest w interfejsie SPI możliwe. Jedynym objawem błędu jest wówczas nieprawidłowa interpretacja danych. Oscyloskop nie sygnalizuje jednak żadnego błędu w takim przypadku.

Omówiliśmy zagadnienia związane z analizą protokołu SPI. Należy dodać, że część uwag dotyczących RS-232 jest wspólna dla wszystkich interfejsów obsługiwanych przez analizator protokołów oscyloskopu Rigol DS2202. Zdekodowane dane nadal można przeglądać w oknie Zoom oraz w tabeli wyników.

W następnym odcinku omówimy badanie interfejsu I²C.

Jarosław Doliński, EP



Rysunek 30. Błąd interpretacji danych wynikający ze złego podania długości ramki

Redakcja Elektroniki Praktycznej dziękuje firmie NDN z Warszawy, autoryzowanemu dystrybutorowi i serwisowi firmy Rigol, za udostępnienie oscyloskopu Rigol DS2202 dla potrzeb tego artykułu.

Analiza protokołów(4)

Analizowanie interfejsu I²C

Termin „analiza protokołów" znany jest elektronikom nie od dziś. Specjaliści różnych dziedzin interpretują go jednak zgoła odmiennie. Inżynier telekomunikacji zajmujący się systemami łączności radiowej czy telefonii komórkowej analizę protokołów będzie rozumiał inaczej niż informatyk, a jeszcze inaczej konstruktor projektujący układy elektroniczne. W artykule zajmiemy się protokołami wykorzystywanymi w popularnych interfejsach komunikacyjnych.

W kolejnej części cyklu omówiono protokół stosowany w interfejsie I²C. Do przesyłania danych wykorzystywane są tylko dwie linie, zatem dwukanałowy oscyloskop Rigol DS2202 pracujący jako analizator protokołów będzie odpowiednim przyrządem.

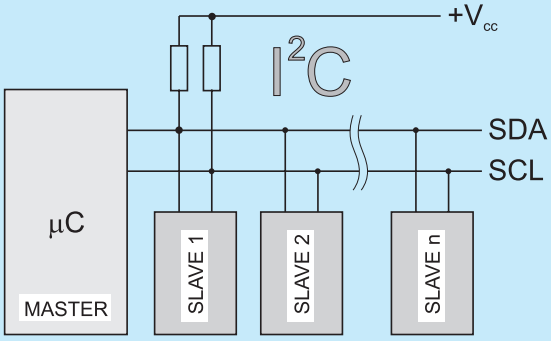
W wyniku burzliwego rozwoju systemów mikroprocesorowych, jaki miał miejsce na początku lat 80. pojawiła się konieczność opracowania wydajnego interfejsu, który byłby wykorzystywany do komunikacji między mikroprocesorem a urządzeniami peryferyjnymi rozumianymi głównie jako specjalizowane układy scalone lub moduły funkcjonalne montowane najczęściej na jednej płytce z procesorem. Stąd m.in. wywodzi się nazwa interfejsu. I²C to akronim od *Inter-Integrated-Circuit*. Zastosowane w interfejsie rozwiązania sprzętowe oraz przyjęty protokół przesyłania danych zapewniają dwukierunkową transmisję w systemie zawierającym wiele nadajników/odbiorników. Założenia techniczne powstały w firmie Philips, która widząc dużą przydatność praktyczną swojego opracowania zastrzegła do niego prawa autorskie. Nawiasem mówiąc jest to często stosowany zabieg pozwalający zwiększać zyski. W rezultacie pomysły zostały skopiowane w trudnych dziś do zliczenia odmianach tego interfejsu, pozwalających omijać zawiłe uwarunkowania prawne.

Hardware I²C

Tematem tej części artykułu jest wprawdzie analiza protokołu I²C, ale trudno ją będzie przeprowadzić bez choćby podstawowej wiedzy o sprzętowych rozwiązaniach interfejsu.

Jak wiemy, w jednym systemie mikroprocesorowym można stosować wiele urządzeń komunikujących się *via* interfejs I²C. Do przesyłania danych wykorzystywana jest wyodrębniona magistrala składająca się z linii SDA, którą są przesyłane adresy i dane oraz linii SCL, którą przesyłany jest przebieg zegarowy taktujący transmisją. Procesor pełni rolę urządzenia nadrzędnego (Master), a pozostałe peryferia są urządzeniami podrzędnymi (Slave) – **rysunek 31**. Możliwa jest również topologia zawierająca kilka mikroprocesorów, ale ich funkcje w systemie muszą być rozdzielone – zawsze Masterem może być tylko jeden procesor.

Do linii interfejsu dołączane są urządzenia o bliżej nieokreślonym położeniu względem siebie. W ogólnym



Rysunek 31. Typowy układ połączeń komunikujących się za pośrednictwem interfejsu I²C

przypadku nie jest też znana ich liczba. W takiej rozproszonej topologii może dochodzić do znacznego zniekształcania sygnałów cyfrowych, uniemożliwiającego prawidłową interpretację danych. Zawsze więc należy stosować odpowiednio dobrane rezystory terminujące. Wszystkie wyjścia dołączone do interfejsu I²C są typu *Open Drain*. Urządzenie nieaktywne wyłącza wyjście – nazywamy to zwolnieniem linii. Jednym z zadań rezystorów terminujących jest również podciąganie linii do stanu wysokiego.

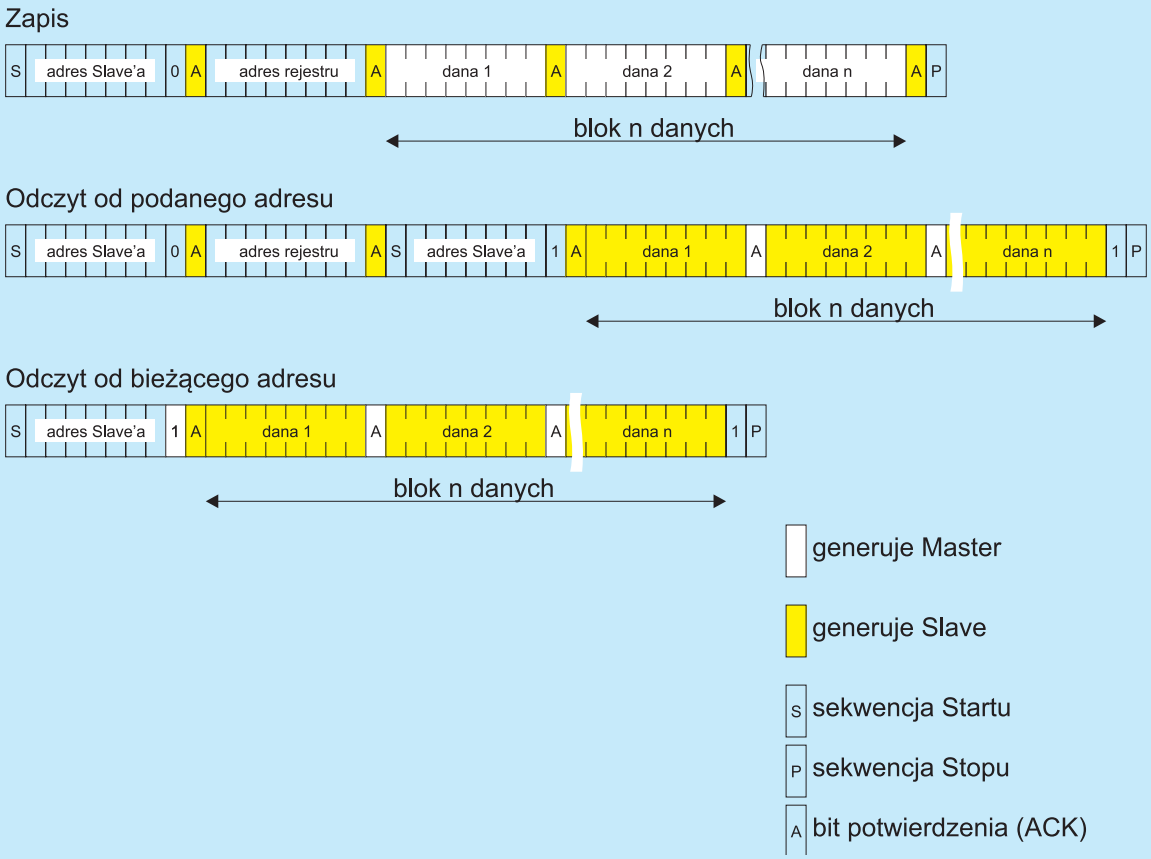
Każda wymiana informacji między urządzeniami Master i Slave jest inicjowana przez Mastera. Master wysyła adres urządzenia Slave, do którego chce wpisać, lub z którego chce odczytać informację. Oczywiście każde urządzenie dołączone do danego interfejsu musi mieć unikatowy adres własny. Adres ten jest najczęściej na sztywno zapisywany w danym typie układu scalonego na etapie jego produkcji. Aby jednak nie dochodziło do konfliktów w przypadku stosowania kilku układów tego samego typu w jednym systemie, przewidziano możliwość modyfikacji kilku bitów adresowych przez użytkownika. Całkowity adres składa się więc z niezmiennej części identyfikującej typ układu i z modyfikowalnej części identyfikującej np. konkretny układ scalony wchodzący w skład systemu mikroprocesorowego. Specyfikacja I²C przewiduje stosowanie adresów 7- lub 10-bitowych

Protokół I²C

W stanie bezczynności linie SCL i SDA są zwolnione (pozostają w stanie wysokim na skutek odłączenia wyjść).

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 31063, pass: 8iyw2174

Rysunek 32. Sekwencja START i STOP oraz bity potwierdzeń ACK



Rysunek 33. Warianty protokołu I²C: a) bezpośredni zapis danych do urządzenia Slave, b) odczyt wskazanego rejestru urządzenia Slave, c) bezpośredni odczyt z urządzenia Slave

Transmisja jest inicjowana wygenerowaniem sekwencji startu „S" (**rysunek 32**). Odpowiada jej opadające zbocze sygnału SDA, podczas gdy linia SCL jest utrzymywana w stanie wysokim. Stany linii SDA mogą się zmieniać tylko wtedy, gdy linia SCL jest w stanie niskim. Po wygenerowaniu sekwencji startu linia SCL jest zerowana, a w chwilę później linia SDA przybiera stan zgodny z wartością pierwszego bitu transmitowanej danej. Następnie na linii SCL generowanych jest 8 impulsów taktujących kolejne bity przesyłane linią danych (SDA). W protokole I²C nie przewidziano modyfikacji długości danych. Zawsze są one 8-bitowe. W każdej ramce pojawia się jednak dziewiąty impuls zegarowy przeznaczony na przesłanie bitu potwierdzającego odebranie informacji. Jeśli operacja, np. nadawania, przebiegła prawidłowo, układ Slave wymusza zerowanie linii SDA (co oznacza pozytywne potwierdzenie — ACK). Stan ten może się przedłużać, np. z powodu dłuższej zajętości odbiornika, wynikającej choćby z obsługi przerwania. Układ Slave wprowadza Mastera w stan oczekiwania (*wait state*) przez przytrzymanie linii SCL W stanie niskim. Do momentu jej zwolnienia wszelkie operacje na magistrali I²C są wstrzymane. Brak potwierdzenia (bit ACK=1) powoduje zakończenie transmisji. Transmisję kończy wystawienie sekwencji stopu — P, czyli wygenerowanie narastającego impulsu na linii SDA, podczas gdy linia SCL pozostaje w stanie wysokim (rysunek 32).

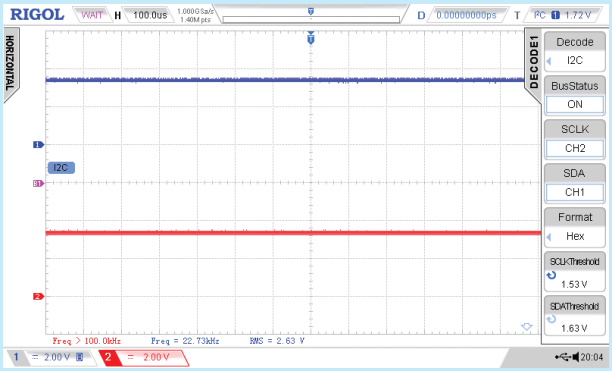
W podstawowej specyfikacji I²C prędkość transmisji jest równa 100 kb/s. W trybie fast jest ona równa 400 kb/s. Od chwili opracowania interfejsu I²C był on kilkakrotnie modyfikowany. Aktualnie prędkość transmisji może dochodzić do 3,4 Mb/s (*High-Speed Mode*).

Warianty pracy

W zależności od funkcji realizowanych przez dany układ peryferyjny stosowane są różne warianty komunikacji. Schematycznie przedstawiono je na **rysunku 33**.

W pierwszym przypadku (rysunek 33a) układ Master zapisuje informację do Slave. Po wygenerowaniu sekwencji startu (S) Master przesyła linią SDA adres układu Slave, do którego kieruje dane (transmisja przebiega od najstarszego do najmłodszego bitu). Po 7 bitach adresu, ósmy bit ma wartość „O" oznaczającą, że dane będą zapisywane do Slave'a. Bajt ten odbierają wszystkie układy dołączone do magistrali, ale tylko ten, którego adres jest zgodny z nadanym odpowiada wystawiając bit potwierdzenia ACK poprzez ściągnięcie linii SDA do stanu niskiego. Następnie nadawane są kolejne bajty, które zawsze muszą być potwierdzane bitem ACK (niski stan) przez układ Slave, do którego kierowane są dane.

Na rysunku 33b przedstawiono blokowy odczyt danych ze Slave'a, począwszy od rejestru o wskazanym adresie. Transmisja jest inicjowana wystawieniem sekwencji startu (S), a następnie zaadresowaniu odpowiedniego Slave'a. Bit ósmy występujący bezpośrednio po adresie, tak jak w poprzednim przypadku ma wartość „0". Po zidentyfikowaniu bitu potwierdzenia wygenerowanym przez Slave'a Master wysyła adres rejestru, od



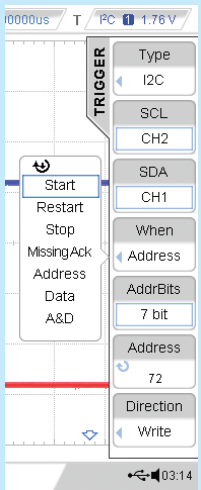
Rysunek 34. Menu konfiguracyjne analizatora protokołu I²C

którego będą odczytywane informacje. Ponownie jest on potwierdzany przez układ Slave bitem ACK. Master musi teraz ustawić logikę Slave'a obsługującą interfejs w tryb odczytu jego rejestrów. Jest to realizowane przez ponowne wysłanie adresu urządzenia Slave z bitem R/W (8 bit) równym 1, co oznacza odczyt. W takt kolejnych impulsów zegarowych przesyłane są bity danych z urządzenia Slave, a bit potwierdzenia ACK wystawia tym razem Master. Adresy rejestrów Slave'a są inkrementowane po przesłaniu każdej 8-bitowej danej. Transmisja kończy się brakiem potwierdzenia (9 bit równy „1") i wygenerowaniem sekwencji stopu (P). Według podobnego schematu przebiega zapis danych do wskazanego rejestru, jednakże przy powtórzeniu adresu Slave'a bit R/W jest równy „0", co oznacza zapis. Dalsza część komunikacji przebiega zgodnie ze schematem przedstawionym na rysunku 33a.

Na rysunku 33c przedstawiono przebieg transmisji podczas bezpośredniego odczytu urządzenia Slave.

Analiza protokołu I²C

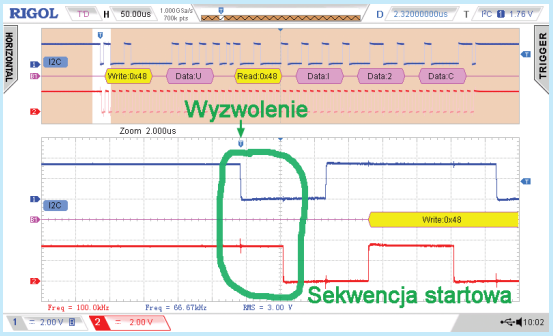
Pracę rozpoczynamy od dołączenia kanałów pomiarowych oscyloskopu do linii interfejsu, np. kanał CH2 do linii SCL, kanał CH1 do linii SDA. Taką konfigurację należy zadeklarować w menu wyświetlanym po naciśnięciu przycisku *Decode1* (lub *Decode2*) – **rysunek 34**. Ramka protokołu I²C nie zawiera zmiennych elementów, w menu „Decode1" („Dekode2") nie ma więc żadnych pól, od ustawienia których zależałaby interpretacja danych na liniach interfejsu. Do obowiązkowych czynności należy jedynie ustawienie prawidłowych poziomów progowych „SCLKThreshold" i „SDAThreshold". Najlepiej, gdy oba te parametry mają wartości równe połowie napięcia logiki badanego interfejsu. Przykładowo, dla układów wykonanych w technologii 3,3 V, poziomy progowe powinny być ustawione na ok. 1,65 V. Właściwie na tym kończy się konfigurowanie analizatora protokołów. W zależności od potrzeb można jeszcze wybrać format interpretowanych danych: Hex, Decimal, Binary lub ASCII. Na drugiej stronie menu „Decode" znajduje się pole „Offset". Po naciśnięciu przypisanego do niego przycisku, za pomocą uniwersalnego pokrętła regulacyjnego ustalana jest najbardziej odpowiednia wysokość wyświetlania dekodowanych danych.



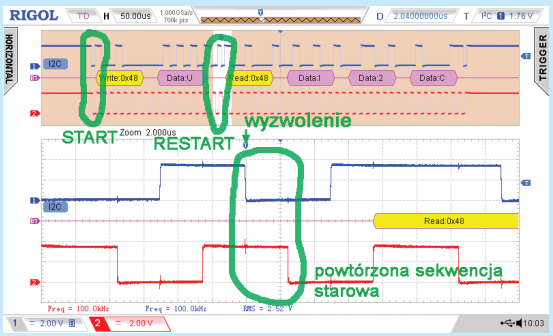
Rysunek 35. Menu wyzwalania I²C

Wyzwalanie protokołu I²C

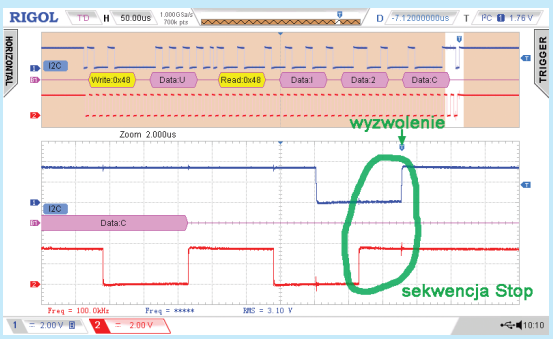
Przebiegi występujące na liniach interfejsu I²C zostaną wyświetlone na ekranie nawet po zastosowaniu zwykłego wyzwalania zboczem. Prawdopodobnie obraz nie będzie jednak stabilny, konieczne będzie skorzystanie z jednej z opcji wyzwalania, które są dedykowane dla protokołu I²C.



Rysunek 36. Wyzwolenie na sekwencji Start



Rysunek 37. Wyzwolenie na powtórzonej sekwencji startowej (Restart)



Rysunek 38. Wyzwolenie na sekwencji Stop

Menu wyzwalania dla protokołu I²C przedstawiono na **rysunku 35.** Pierwsze trzy pola są powtórzeniem nastaw konfiguracyjnych analizatora, a więc np.: „Type"=I²C, „SCL"=CH2, „SDA"=CH1. Warto już na początku wybrać tryb wyzwalania Normal, który w większości przypadków będzie najbardziej odpowiedni. W trybie „Auto" oscylogram może być niestabilny.

W polu „When" należy wybrać jeden z kilku warunków wyzwalania. Wyboru dokonujemy w zależności od tego, co chcemy obserwować. Może to być:

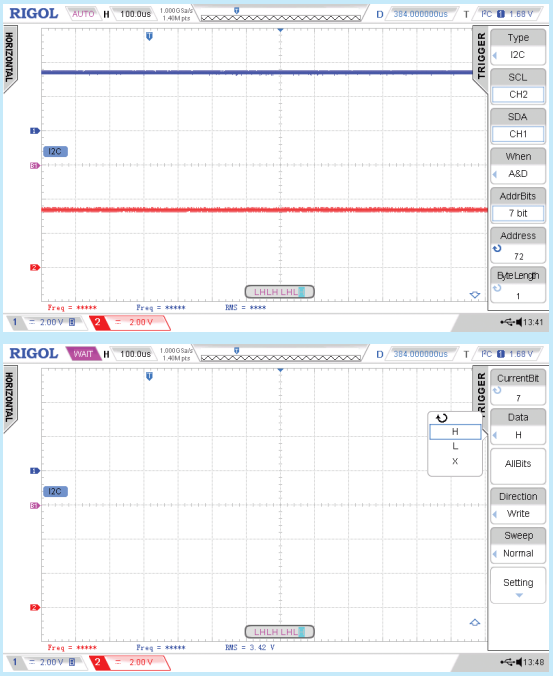
- **Start.** Oscyloskop jest wyzwalany na sekwencji startowej. Jest to dobry wybór na przykład do obserwacji powtarzających się cyklicznie komunikatów. W sesjach z powtórzeniem sekwencji startowej, stosowanej np. podczas odczytu rejestru o określonym adresie, przesłanym po adresie urządzenia Slave, sekwencja startowa występuje dwukrotnie – na początku i po przesłaniu adresu rejestru. Układ wyzwalania oscyloskopu DS2202 rozróżnia te sekwencje. Po wybraniu opcji *Start* wyzwolenie następuje zawsze

Rysunek 39. Wyzwolenie na błędzie potwierdzenia *MissingAck*

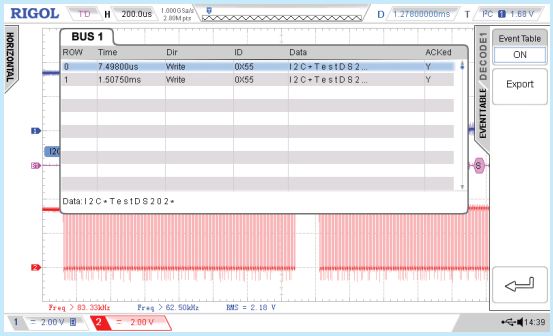
na pierwszej sekwencji startowej (**rysunek 36**). Aby wyzwalać oscyloskop na drugiej sekwencji należy zastosować opcję *Restart*.

- **Restart.** Ta opcja jest wykorzystywana do wyzwolenia oscyloskopu w chwili powtórzenia sekwencji startowej. Jest przydatna w trybach zapisu lub odczytu rejestru urządzenia o podanym adresie. Oscyloskop jest wyzwalany po przesłaniu adresu rejestru i powtórzonej sekwencji startowej (**rysunek 37**). Mówimy o powtórzeniu sekwencji startowej (restarcie), gdyż przed jej wystawieniem nie było sekwencji Stop.
- **Stop.** Wyzwalanie na sekwencji *Stop* kończącej transmisję (**rysunek 38**).
- **MissingAck.** Wyzwalanie po wykryciu błędnego potwierdzenia (bit ACK w ramce protokołu I²C). Na **rysunku 39** przedstawiono przykładową sytuację, w której wykryto błędne potwierdzenie po przesłaniu adresu urządzenia Slave. Zgodnie z protokołem, na 9. bicie zegarowym powinien wystąpić niski stan na linii SDA, co oznaczałoby poprawne potwierdzenie. Analizator wykrył stan wysoki, a wykryty błąd jest sygnalizowany czerwonym znacznikiem.
- **Address.** Wyzwalanie po wykryciu określonego adresu urządzenia Slave. Pamiętajmy, że w bajcie adresowym przesyłana jest też na jego najmłodszym bicie informacja o kierunku transmisji (R/W). Jeśli R/W=0, nastąpi zapis do układu Slave, jeśli R/W=1, nastąpi odczyt układu Slave. Sam 7-bitowy adres jest liczony od bitu nr 1 do bitu nr 7. Najlepiej zilustruje to przykład. Jeżeli cały bajt adresowy ma postać 10010000, to adres jest równy 1001000, czyli 48h i mamy do czynienia z zapisem. Jeżeli bajt adresowy ma postać 01001101, to adres jest równy 0100110, czyli 26h i mamy do czynienia z odczytem. W wyświetlonym na ekranie oscyloskopu adresie nie jest uwzględniany bit R/W (wg powyższej konwencji). Po wybraniu opcji *Address* należy określić długość adresu (7 lub 10 bitów). Adres jest niestety podawany w zapisie dziesiętnym (a więc zwykle trzeba go będzie przeliczać z postaci szesnastkowej). Kolejny parametr („Direction"), określający kierunek transmisji może przybierać wartości: *Write* (czemu odpowiada R/W=0), *Read* (czemu odpowiada R/W=1) lub *R/W* (wyzwolenie nastąpi na zadanym adresie w trybie zapisu, albo odczytu). Na **rysunku 40a** przedstawiono wyzwolenie po przesłaniu adresu 75h w trybie zapisu, a na rysunku 40b wyzwolenie nastąpiło na adresie 48h w trybie odczytu.
- **Data.** Wyzwolenie po wykryciu zdefiniowanej danej lub sekwencji danych. Analizowany ciąg danych może mieć długość od 1 do 5 bajtów. Definiowanie danych jest dość żmudne, gdyż wymaga indywidualnego ustawienia każdego bitu. Dla danej 5-bitowej trzeba więc wykonać aż 40 operacji. Bit może przyjmować wartości „L", „H" lub „X". „X" oznacza, że dany bit nie będzie sprawdzany w odebranym ciągu. Przy definiowaniu bitów przyjęto dość mylącą konwencję, w której najstarszy bit ma numer 0, najmłodszy zaś numer 7. Opcja ta byłaby bardzo przydatna w praktyce, niestety, w testowanym egzemplarzu oscyloskopu Rigol DS2202 nie działała.
- **A&D**. Działa za to poprawnie wyzwalanie sekwencją adres i dana. Ten tryb wyzwalania jest bardzo przydatny na przykład podczas lokalizacji początku

Rysunek 40. Wyzwolenie na adresie 75h w trybie: a) zapisu, b) odczytu



Rysunek 41. Tabelaryczna postać dekodowanych danych dla protokołu I²C



Rysunek 42.

transmisji do określonego rejestru urządzenia Slave o zadanym adresie. Na przykład, gdy chcemy wyzwolić oscyloskop na początku przesyłania danych do rejestru o numerze 55h należącego do urządzenia Slave o adresie 48h, należy wybrać opcję wyzwalania *A&D*, następnie zadeklarować długość adresu (np. 7 bitów), prowadzić ten adres (niestety tu znowu obowiązuje postać dziesiętna, adres 48h jest więc równy 72 i taką wartość należy wprowadzić w polu „Address"), teraz deklarujemy długość danej – w naszym przykładzie będzie równa 1. Pierwsza strona opcji wyzwalania powinna wyglądać tak, jak na **rysunku 41a,** ale to nie koniec. Przechodzimy na drugą stronę, na której znajdujemy znane już opcje ustawiania bitów danych, a dodatkowo należy tu również określić kierunek transmisji (*Read, Write, R/W*). Druga strona opcji wyzwalania dla naszego przykładu powinna wyglądać jak na rysunku 41b. Przyjęto, że dana ma wartość 55h, czyli 01010101. Oscylogram dla tego przykładu przedstawiono na **rysunku 42.**

Niezależnie od trybów wyzwalania oscyloskop Rigol DS2202 pracujący jako analizator protokołu I²C, może wyświetlać zdekodowane dane jako oscylogramy lub w postaci tabelarycznej. Tabela jest włączana po naciśnięciu przycisku *Decode1* (*Decode2*) i przejściu na drugą stronę menu. Należy uaktywnić polecenie „Event Table". Zdekodowane w ten sposób dane przedstawiono na **rysunku 43**.

W następnym odcinku omówimy ostatni, dostępny w oscyloskopie Rigol DS2202 protokół. Będzie to analiza magistrali równoległej.

Jarosław Doliński, EP

Redakcja Elektroniki Praktycznej dziękuje firmie NDN z Warszawy, autoryzowanemu dystrybutorowi i serwisowi firmy Rigol, za udostępnienie oscyloskopu Rigol DS2202 dla potrzeb tego artykułu.

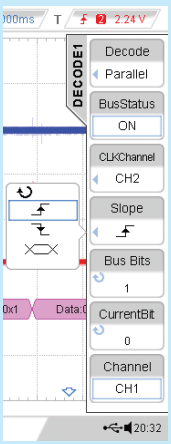
Analiza protokołów (5)

Analizowanie protokołu równoległego (parallel)

Termin „analiza protokołów" znany jest elektronikom nie od dziś. Specjaliści różnych dziedzin interpretują go jednak zgoła odmiennie. Inżynier telekomunikacji zajmujący się systemami łączności radiowej czy telefonii komórkowej analizę protokołów będzie rozumiał inaczej niż informatyk, a jeszcze inaczej konstruktor projektujący układy elektroniczne. W artykule zajmiemy się protokołami wykorzystywanymi w popularnych interfejsach komunikacyjnych.

Dotarliśmy do ostatniego odcinka cyklu artykułów o analizie protokołów. Do omówienia pozostał protokół równoległy (*parallel*), a więc taki, w którym informacja jest przekazywana równolegle kilkoma liniami tworzącymi interfejs. Dodatkowa linia strobująca wyznacza moment zapisu/odczytu danych. Zdarzenie to zachodzi w chwili występowania wybranego zbocza sygnału strobującego.

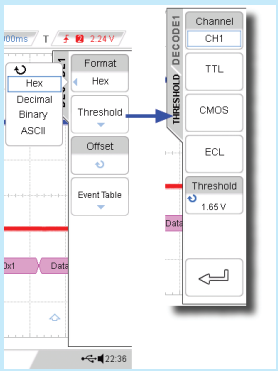
Pamiętamy, że jako przyrząd wzorcowy przyjęto do niniejszego cyklu artykułów oscyloskop Rigol DS2202. Jest on wykorzystywany we wszystkich eksperymentach i pomiarach. Niestety, ten typ oscyloskopu ma tylko dwa kanały analogowe, więc jego przydatność w badaniu interfejsów równoległych jest bardzo ograniczona. Fragment firmware'u realizujący funkcję analizatora protokołów równoległych jest jednak wykorzystywany również w oscyloskopach 4-kanałowych, a także w modelach z 16 kanałami cyfrowymi. Za pomocą tych oscyloskopów można dokonywać pełnej analizy protokołu *Parallel*.



Rysunek 43. Okno konfiguracji protokołu *Parallel* (pierwsza strona)

Konfiguracja analizatora protokołu Parallel

Jak zatem badać interfejs równoległy oscyloskopem DS2202? Liczba fizycznych linii danych musi być oczywiście ograniczona do... 1, drugi kanał jest natomiast wykorzystywany do obsługi linii strobującej. Przypisanie kanałów pomiarowych do linii interfejsu jest dowolne, a odpowiednią konfigurację przeprowadza się po naciśnięciu przycisku *Decode1*/*Decode2* i wybraniu opcji „Decode"=Parallel (**rysunek 43**). Wyświetlanie dekodowanego stanu interfejsu równoległego jest możliwe po ustawieniu opcji „BusStatus"=On. Z kolei opcja „ClkChannel" definiuje kanał oscyloskopu przypisany do linii strobującej. W artykule przyjęto, że będzie to kanał 2. Opcja „Slope" określa zbocze sygnału strobującego (zegarowego), na którym następuje odczyt linii danych i ich interpretacja (dekodowanie). Możliwe są wszystkie przypadki, tzn. odczyt na: narastającym zboczu, opadającym zboczu, narastającym lub opadającym zboczu. Następna pozycja menu konfigurującego „BusBits" określa liczbę linii danych w badanym interfejsie. W przypadku oscyloskopu DS2202 jedyną sensowną nastawą jest „BusBits"=1, gdyż tylko jedną linię interfejsu równoległego można analizować. Firmware nie ogranicza jednak tego parametru i nawet w DS2202 może być on różny od jedności. O tym, jakie będą konsekwencje takiej nastawy będzie mowa w dalszej części artykułu.



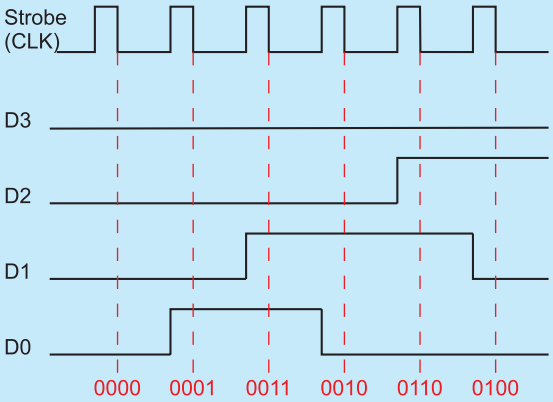
Rysunek 44. Okno konfiguracji protokołu *Parallel* (druga strona)

Następne dwie opcje, tj.: „CurrentBit" i „Channel" przypisują kanały pomiarowe do kolejnych linii interfejsu równoległego. Ponieważ wcześniej przyjęliśmy, że do linii strobującej jest dołączony kanał 2, ustawiamy „CurrentBit"=0 i „Channel"=CH1 (rysunek 43). Oznacza to, że kanał 1 oscyloskopu jest dołączony do linii 0 interfejsu.

Druga strona menu konfigurująca protokół *Parallel* (**rysunek 44**) zawiera ponadto opcje formatu wyświetlanych wyników – „Format" (Hex, Decimal, Binary, a nawet ASCII), ustalenia poziomu progowego – „Threshold", pionowego przesuwu dekodowanych informacji – „Offset", a także włączenia stablicowanej postaci wyników – EventTable". Opcja ASCII dla 1-bitowego interfejsu wygląda dość kuriozalnie, ale przymykamy na to oko. Pewnego komentarza wymaga też opcja „Threshold". Po jej rozwinięciu pojawia się kilka pozycji do wyboru. Każda z nich oznacza przyjęcie predefiniowanego poziomu progowego. I tak:

- TTL ma próg 2,5 V, a więc oznacza starą, 5-woltową technologię TTL,

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 28637, pass: 752sjb64



Rysunek 45. Przebiegi w przykładowym interfejsie równoległym

- CMOS ma próg 1,65 V, a więc nadaje się również dla innych technologii 3-woltowych, np. LVTTL,
- ECL ma próg –1,3 V.

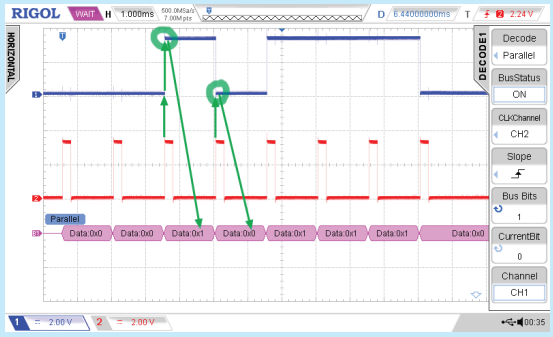
Możliwe jest też ręczne ustawienie progu o dowolnej wartości.

Analiza protokołu Parallel

Zasadę działania interfejsu równoległego przedstawiono na **rysunku 45**. Przyjęto, że jest to interfejs składający się z 4 linii danych i linii zegarowej (strobującej). Stany linii danych są zmieniane na narastających zboczach sygnału zegarowego, natomiast odczyt linii następuje na zboczach opadających. Takiego interfejsu nie da się zbadać oscyloskopem DS2202, gdyż jak wiemy, nie ma on wystarczającej liczby wejść. Można natomiast badać pojedyncze linie danych.

Na **rysunku 46** przedstawiono przykładowy oscylogram analizy protokołu *Parallel*. Kanał 1 dołączono do linii danych, kanał 2 do linii sygnału zegarowego.

Rysunek 46. Przykładowy wynik analizy protokołu *Parallel*



Rysunek 47. Interpretacja danych zmieniających się na zboczu odczytu

Rysunek 48. Interpretacja danych na obu zboczach zegarowych

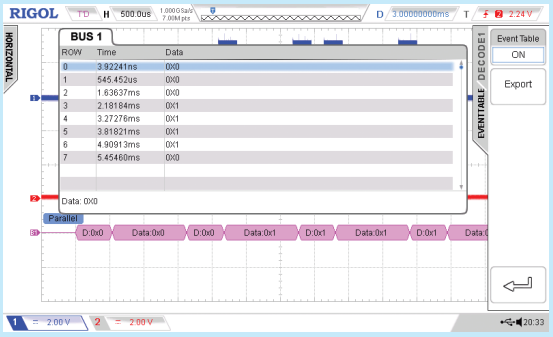
Interpretacja danych następuje na opadającym zboczu sygnału zegarowego. Zdekodowane dane są umieszczane na wykresie B1 („Parallel") za zboczem strobującym.

Nasuwa się tu pytanie, jak analizator zinterpretuje dane, jeśli będą się one zmieniały na zboczach odczytu. Przypadek taki przedstawiono na **rysunku 47.** Jak widać w opcji „Slope" menu, do pomiaru wybrano zbocze narastające, i na tym zboczu następują zmiany stanów na linii danych. Jest to sytuacja nieprawidłowa, ale analizator musi ją jakoś zinterpretować. Na wykresach z rysunku 47 widać, że analizator dekodował stan, jaki występował na linii danych tuż za zboczem zegarowym. Nie musi to być jednak reguła.

Analizator protokołu Parallel może być skonfigurowany również tak, aby dekodował dane na obu zboczach. Wynik będzie wówczas poprawny jedynie wtedy, gdy stany linii interfejsu nie będą się zmieniały na zboczach sygnału zegarowego. Trudno podać przykład interfejsu równoległego pracującego zgodnie z tym założeniem, nie mniej jednak konstruktorzy oscyloskopu DS2202 przewidzieli taką możliwość (**rysunek 48**).

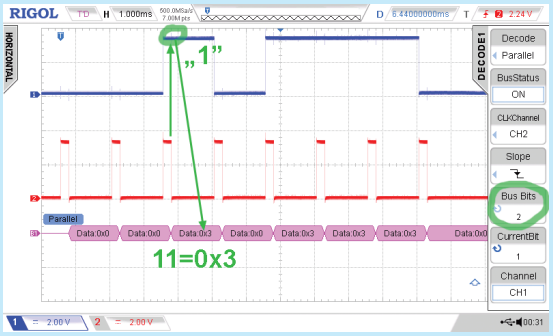
Podstawową formą wizualizacji danych dekodowanych przez analizator protokołów, przy tym najczęściej stosowaną, są wykresy czasowe. Niekiedy bardziej czytelne są jednak informacje przekazywane w postaci tabelarycznej. Wiemy, że możliwość taka istniała dla wcześniej omawianych protokołów, dla protokołu równoległego również ją zachowano. Tabelę zawierającą zdekodowane dane przedstawiono na **rysunku 49.** Można ją eksportować w formacie CSV do pliku zapisywanego na pendrivie, a następnie analizować np. w Excelu.

Na zakończenie pozostało jeszcze rozwikłanie pewnej wątpliwości, mianowicie: co będzie się działo, jeśli liczba linii interfejsu równoległego zostanie określona nieprawidłowo? Popełnienie błędu od dołu nie jest możliwe, gdyż najmniejsza wartość parametru „BusBits" jest



Rysunek 49. Tabelaryczna forma wyników analizy protokołu *Parallel*

równa 1, a jeden kanał zawsze jest dostępny w oscyloskopie. Jednak już przy wartości „BusBits”=2 pojawia się wątpliwość, gdyż w oscyloskopie DS2202 nie ma możliwości fizycznej analizy dwóch linii danych. Analizator nie będzie jednak sygnalizował błędu. Wszystkie linie zostaną wirtualnie ze sobą połączone, a ich stan będzie taki sam, jaki występuje na linii, do której jest dołączony fizyczny kanał oscyloskopu. Przykład przedstawiono na **rysunku 50**. Przyjęto, że interfejs ma dwie linie danych i że są one przypisane do kanału 1 oscyloskopu. W zaznaczonym na oscylogramie impulsie zegarowym odczytano stan wysoki na wejściu CH1, zatem analizator interpretuje stan wysoki na obu liniach interfejsu (D0 i D1), czemu odpowiada heksadecymalny zapis 0x03.



Rysunek 50. Interpretacja danych w przypadku zadeklarowania nieprawidłowej liczby danych interfejsu równoległego

Uwagi

W cyklu artykułów o analizie protokołów ograniczono się do opcji dostępnych w oscyloskopie Rigol DS2202. Ogólnie zasada pracy z tego typu urządzeniami jest podobna, niezależnie od producenta i od rodzaju oscyloskopu (stacjonarny, USB czy analizator stanów logicznych z opcją analizy protokołów). Mogą oczywiście występować drobne różnice wynikające z rozwiązań konstrukcyjnych czy przyjętych założeń, ale nie powinny mieć istotnego wpływu na obsługę tych urządzeń. Jest jednak jeden warunek konieczny – użytkownik powinien dobrze znać i rozumieć badany protokół, by móc prawidłowo interpretować analizowane zdarzenia.

Jarosław Doliński, EP

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 28637, pass: 752sjb64