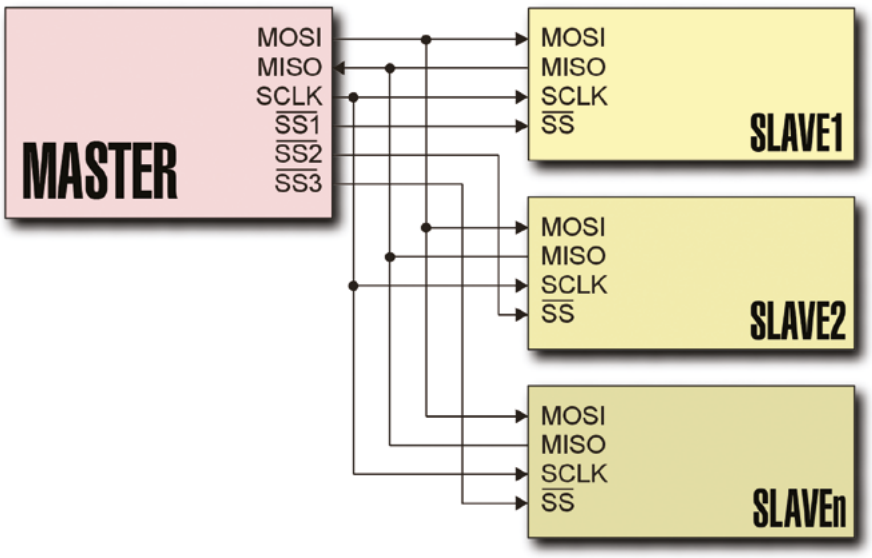
Analizowanie protokołów szeregowych oscyloskopami Rohde&Schwarz (2) – SPI, I²C

Badanie protokołów komunikacyjnych jest już obowiązkową funkcją oscyloskopów cyfrowych co najmniej średniej klasy. Nadal jednak jest ona udostępniana najczęściej jako opcjonalne rozszerzenie oprogramowania firmowego. Zwykle w cenie oscyloskopu mieści się tylko kilka najbardziej popularnych protokołów, np. SPI i UART. W cyklu artykułów przedstawiono rozwiązania dotyczące analizy protokołów zastosowane w oscyloskopach Rohde&Schwarz. W tej części artykułu omówiono badanie interfejsów SPI i I²C.



Rysunek 1. Typowa konfiguracja urządzeń komunikujących się ze sobą przez interfejs SPI

Interfejs komunikacyjny SPI jest jednym z najczęściej stosownych w sprzęcie elektronicznym. Najczęściej jest on wykorzystywany do wymiany danych pomiędzy różnymi blokami funkcjonalnymi urządzenia, między procesorem a podzespołami z zaszytą „inteligencją", wszelkiego rodzaju czujnikami, wyświetlaczami itd. Podobne przeznaczenie ma też interfejs I²C, więc można uznać, że w pewnym stopniu oba rozwiązania są dla siebie konkurencyjne. Można też spotykać podzespoły komunikujące się zarówno przez I²C, jak i SPI. Lansowany niegdyś przez firmę Dallas interfejs 1-wire, który też miał służyć do wewnętrznej wymiany danych w urządzeniach elektronicznych nie przyjął się u innych producentów. Dzisiaj jest już prawie zapomniany, nawet trudno znaleźć analizator przystosowany do stosowanego w nim protokołu.

Protokół SPI

Wracamy zatem do interfejsu SPI (Serial Peripheral Interface), którego popularność jest bardzo mocna i nic nie wskazuje na to, by miał on być w najbliższym czasie zastąpiony innym lub, co gorsze, by miano go wycofać z użytku.

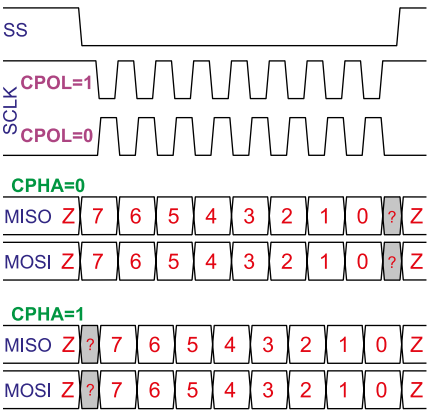
W odróżnieniu od opisywanego w pierwszej części interfejsu UART/RS232, w SPI zastosowano transmisję synchroniczną. Konsekwencją takiej koncepcji jest dołożenie linii zegarowej SCLK taktującej przesyłaniem znaków. Przyjęto również, że w połączeniu *via* SPI zawsze jest jedno urządzenie (Master) pełniące funkcję nadrzędną w stosunku do wielu urządzeń Slave, z którymi może się komunikować. Jak widać, nie ma możliwości bezpośredniego przekazywania danych pomiędzy urządzeniami Slave (**rysunek 1**). Oprócz linii SCLK interfejs SPI zawiera jeszcze dwie linie: MOSI (Master Output Slave Input) – linia, którą Master wysyła dane do Slave'a i MISO (Master Input Slave Output), którą Master odbiera dane od Slave'a. W większości zastosowań układy Slave są wybierane liniami SS. Master musi mieć ich tyle, ile jest urządzeń Slave, aby móc je jednoznacznie adresować.

Analizowanie protokołu SPI oscyloskopami Rohde&Schwarz

Jak zawsze, analizę należy rozpocząć od wybrania typu interfejsu i zdefiniowania wszystkich jego parametrów. Czynności te wykonuje się na przykład po naciśnięciu przycisku *PROTOKOL* znajdującego się na płycie czołowej oscyloskopu. Na ekranie zostaje wyświetlona zakładka zawierająca wszystkie parametry badanego interfejsu (**rysunek 2**). Ramka protokołu SPI jest prostsza niż ramka UART/RS232 - składa się z samych bitów danych, których może być od 4 do 32. Nie ma w niej bitu parzystości, ani bitów startu i stopu (transmisja synchroniczna). Jeśli jednak uwzględnimy różne powiązania linii danych (MOSI i MISO) z linią zegarową SCLK, to powstaje więcej kombinacji konfiguracyjnych. Najważniejsze parametry, to poziom linii SCLK w stanie Idle, kiedy nie są nadawane dane (niski lub wysoki), a także określenie zbocza, wskazującego ważność danych na liniach MOSI i MISO. Może to być pierwsze lub drugie zbocze sygnału SCLK po wyjściu ze stanu Idle. Wymienione kombinacje są często opisywane parametrami CPOL (CPOL=1 dla SCLK=1 w stanie Idle) i CPHA (CPHA=1 dla drugiego zbocza

Rysunek 2. Zakładka z parametrami protokołu SPI

na linii SCLK). Poszczególne kombinacje kojarzone są też z trybami pracy interfejsu SPI (od 0 do 3). Pozostaje jeszcze kwestia uaktywniania układu Slave przez Master'a. Jak wiemy, służy do tego linia SS, więc należy określić, który poziom tej linii jest aktywny (niski czy wysoki). Podanie aktywnego poziomu na linię SS powoduje wyzerowanie



Rysunek 3. Tryby pracy interfejsu SPI

logiki realizującej transmisję i przejście w stan gotowości do wymiany danych. Dane są przesyłane w takt impulsów zegarowych generowanych przez urządzenie Master. W ogólnym przypadku, jeśli linia SS przejdzie do stanu nieaktywnego bez generacji impulsów zegarowych, do transmisji nie dochodzi. W oscyloskopach R&S przewidziano też badanie przypadków, w których inicjowanie transmisji następuje bez uaktywniania linii SS. Kolejne transmisje są wówczas oddzielane odpowiednią przerwą czasową (timeoutem), w czasie której interfejs pozostaje w stanie Idle. Tryby pracy interfejsu SPI przedstawiono na **rysunku 3.**

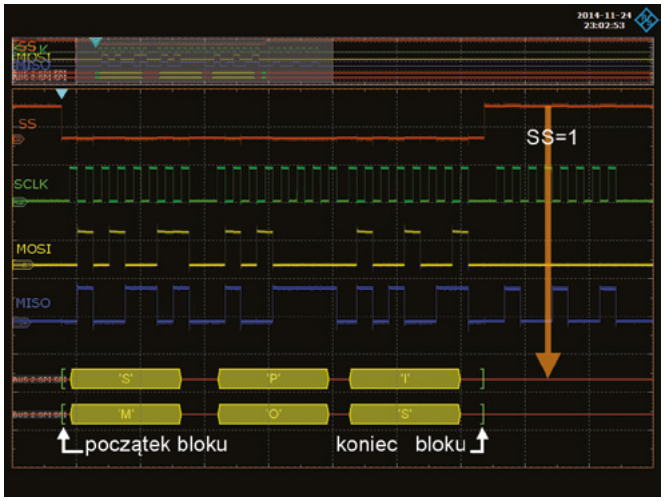
Trochę to skomplikowane, więc pora na nieco praktyki. Na rys. 2. przedstawiono przykładową konfigurację interfejsu SPI. Na szczęście grafika wyświetlana na tej zakładce bardzo pomaga w prawidłowym ustawieniu parametrów. W przykładzie wybrano:

- niski poziom na linii SCLK w stanie Idle,
- pierwsze zbocze na linii SCLK odczytujące dane,
- aktywny poziom wysoki na liniach MISO i MOSI,
- aktywny poziom niski na linii SS,
- kierunek transmisji od najstarszego bitu (MSB) do najmłodszego, dana 8-bitowa,
- uaktywnianie urządzeń Slave poziomem niskim sygnału SS.

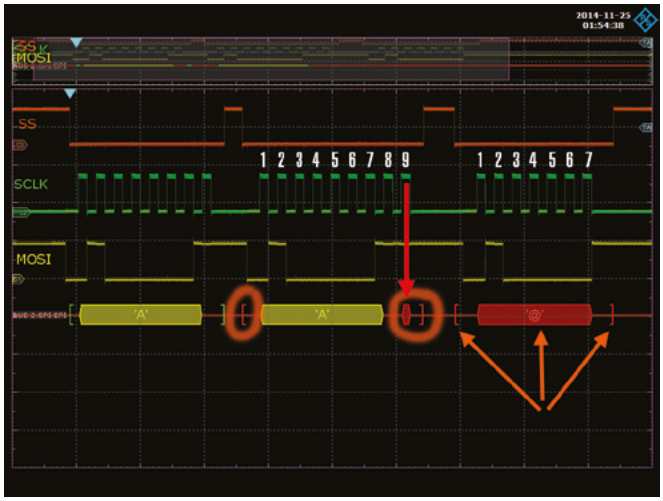
W ustawionym trybie pracy interfejsu SPI dane na liniach MOSI i MISO zmieniają się na opadającym zboczu sygnału SCLK, a odczytywane/zapisywane są na zboczu narastającym (**rysunek 4**). Analizator dekoduje dane i wyświetla je na ekranie w jednym z wybranych formatów (np. hex). Prawidłowo zdekodowana dana jest wyświetlana w kolorze żółtym. Ostatnia, czwarta dana widoczna na rys. 4 nie została zdekodowana, gdyż w trakcie jej nadawania stan linii SS jest już wysoki, a więc nieaktywny. Możliwe, że dana ta jest wysyłana do innego układu Slave, którego linia wyboru SS nie jest śledzona przez analizator protokołów.

Na **rysunku 5** przedstawiono kilka możliwych błędów transmisji interfejsem SPI. W opisanym przykładzie Master 3-krotnie przesyła znak ASCII „A". Za każdym razem układ Slave jest wybierany linią SS (niskim poziomem). Pierwszy bajt został przesłany bezbłędnie, w drugim pojawił się dodatkowy, dziewiąty impuls zegarowy. W tym przypadku jego lokalizacja po 8 prawidłowych impulsach nie zakłóciła interpretacji danej. Bajt został prawidłowo rozpoznany jako „A", jednak na wykresie BUS zawierającym dekodowane znaki został wyświetlony czerwony znacznik błędu. Kolorem czerwonym zostały także narysowane znaczniki początku i końca bloku, które normalnie mają kolor zielony. W trzecim bajcie z kolei zabrakło jednego impulsu zegarowego. Dana została zinterpretowana błędnie jako znak „@" i w całości wyświetlona w kolorze czerwonym.

Jak już było wspomniane, w specyficznym trybie pracy interfejsu SPI linia SS nie musi być wykorzystywana. Kolejne bloki danych są rozróżniane na podstawie czasu



Rysunek 4. Przykładowa transmisja realizowana interfejsem SPI

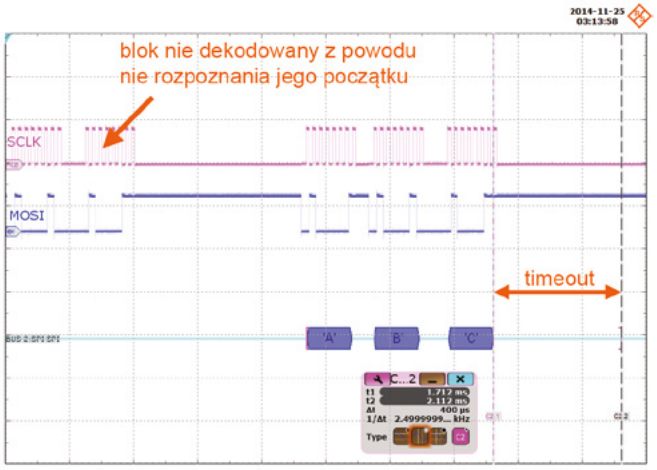


Rysunek 5. Błędy transmisji obserwowane w interfejsie SPI

przerwy między nimi. Na **rysunku 6.** przedstawiono analizę wykonaną w takim przypadku. Przyjęto, że timeout jest tu równy 400 μs. Jest to czas odmierzany od początku ostatniego impulsu zegarowego. Jeśli w tym czasie nie wystąpi nowy impuls zegarowy, analizator uznaje, że blok został zakończony i na oscylogramie umieszczany jest odpowiedni znacznik. Ale uwaga, aby blok mógł być prawidłowo rozpoznany, analizator musi prawidłowo zlokalizować jego początek. Jeśli znajdzie się on poza lewą krawędzią ekranu, to cały blok nie jest dekodowany, mimo że doskonale można rozpoznać poszczególne bajty transmisji. Jest to dobrze widoczne na rys. 6.

Wyzwalanie zdarzeniami protokołu SPI

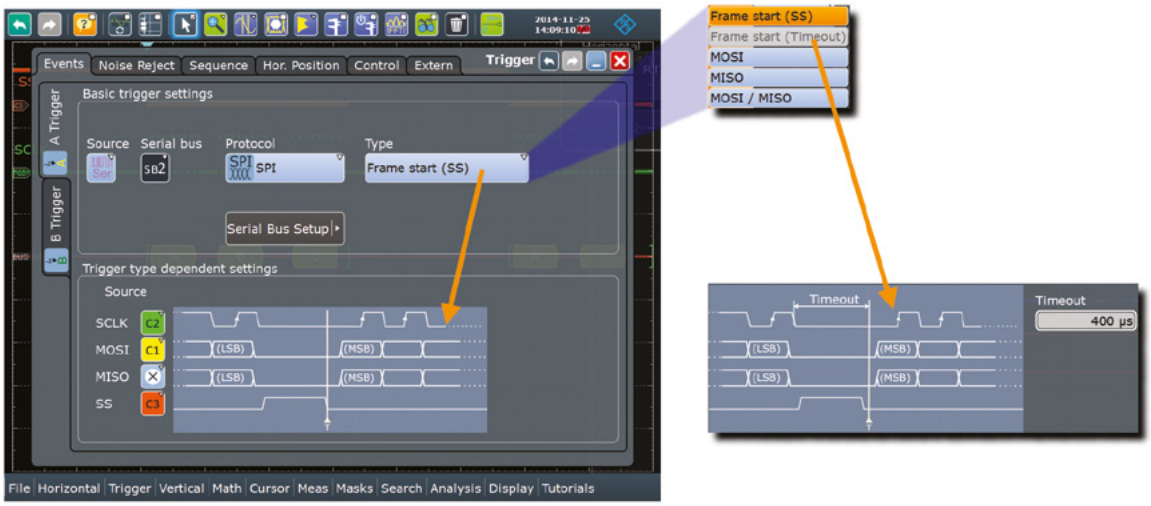
W interfejsie SPI występuje stosunkowo mało charakterystycznych zdarzeń, które mogłyby być wykorzystywane do wyzwalania. Pierwszym, nasuwającym się intuicyjnie, jest początek bloku. Zdarzenie to jest rozpoznawane po uaktywnieniu linii SS lub po przekroczeniu timeoutu rozdzielającego (rys. 8). Ostatnia opcja oznacza, że do wyzwolenia oscyloskopu konieczne jest jednoczesne spełnienie warunku dla linii MOSI i MISO. Przykładowe wyzwolenie po wykryciu sekwencji znaków 'B', 'C' na linii MOSI przedstawiono na **rysunku 9.**



Rysunek 6. Transmisja via SPI bez linii SS - koniec bloku rozpoznawany na podstawie wykrycia przekroczenie timeoutu

Protokół I²C

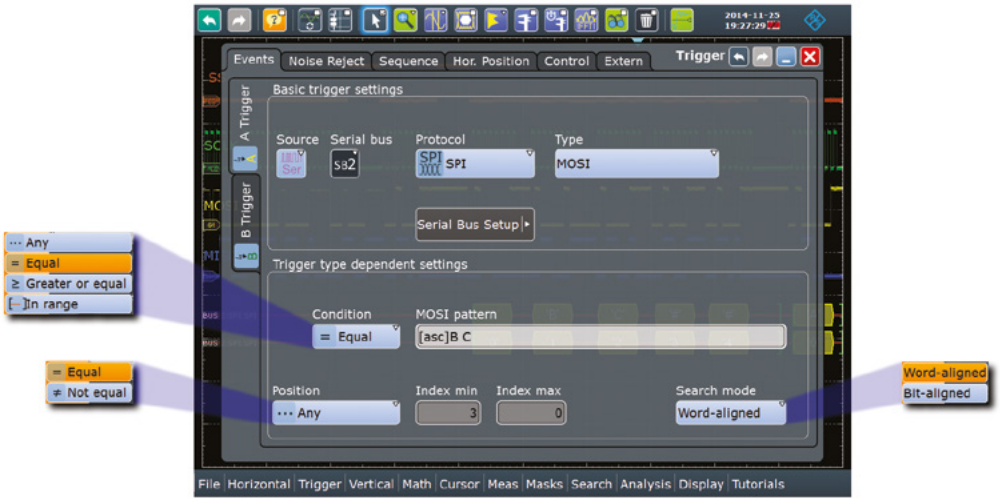
Jak już było wspomniane, interfejs I²C pełni podobne do SPI funkcje w urządzeniach elektronicznych. Oba interfejsy różnią się jednak zarówno rozwiązaniami układowymi, jak i protokołem. Zasadnicze podobieństwo to transmisja synchroniczna przebiegająca między urządzeniem Master i Slave. Dane i adresy urządzeń są przesyłane linią SDA w takt impulsów zegarowych na linii SCL. Obie linie są podciągane rezystorami do napięcia zasilającego. Wszystkie wyjścia dołączone do interfejsu I²C są typu *Open Drain*. Urządzenie nieaktywne wyłącza wyjście (zwalnia linię). Schemat blokowy interfejsu przedstawiono na **rysunku 10**. Nie trudno zauważyć, że na schemacie tym



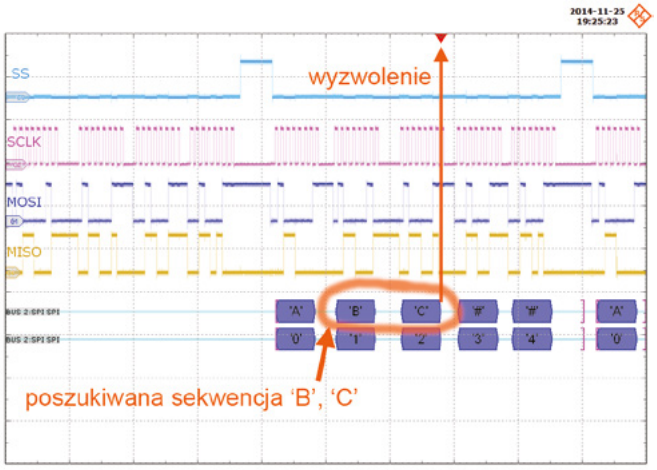
Rysunek 7. Opcje wyzwalania po wykryciu początku ramki

bloki (**rysunek 7**). Opcja związana z timeoutem jest dostępna oczywiście tylko wtedy, gdy w definicji protokołu zaznaczono opcję „Frame condition -> CLK Timeout" (rys. 2).

Jedną z metod poszukiwania określonej danej lub sekwencji danych przesyłanych liniami MOSI lub MISO jest wyzwolenie oscyloskopu po wykryciu obecności takich danych na tych liniach (**rysunek 8**). W zakładce wyzwalania należy więc wybrać opcje: Protokol->SPI i Type->MOSI, lub w zależności od potrzeb Type->MISO, a nawet Type->MOSI/MISO, a następnie wprowadzić wzorzec poszukiwanych danych



Rysunek 8. Zakładka z wyborem opcji wyzwalania po detekcji określonej sekwencji na linii MOSI (także MISO i MOSI/MISO)

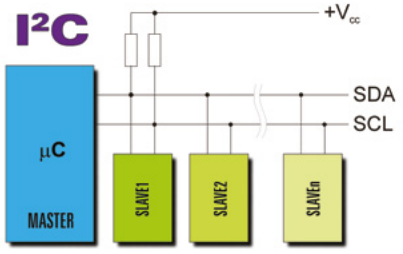


Rysunek 9. Wyzwolenie po wykryciu sekwencji znaków 'B', 'C' na linii MOSI

nie ma linii wybierających urządzenia Slave. Są one adresowane na podstawie pierwszego bajtu występującego po sekwencji startowej. Każde urządzenie ma swój unikatowy w danym systemie 7- lub 10-bitowy adres.

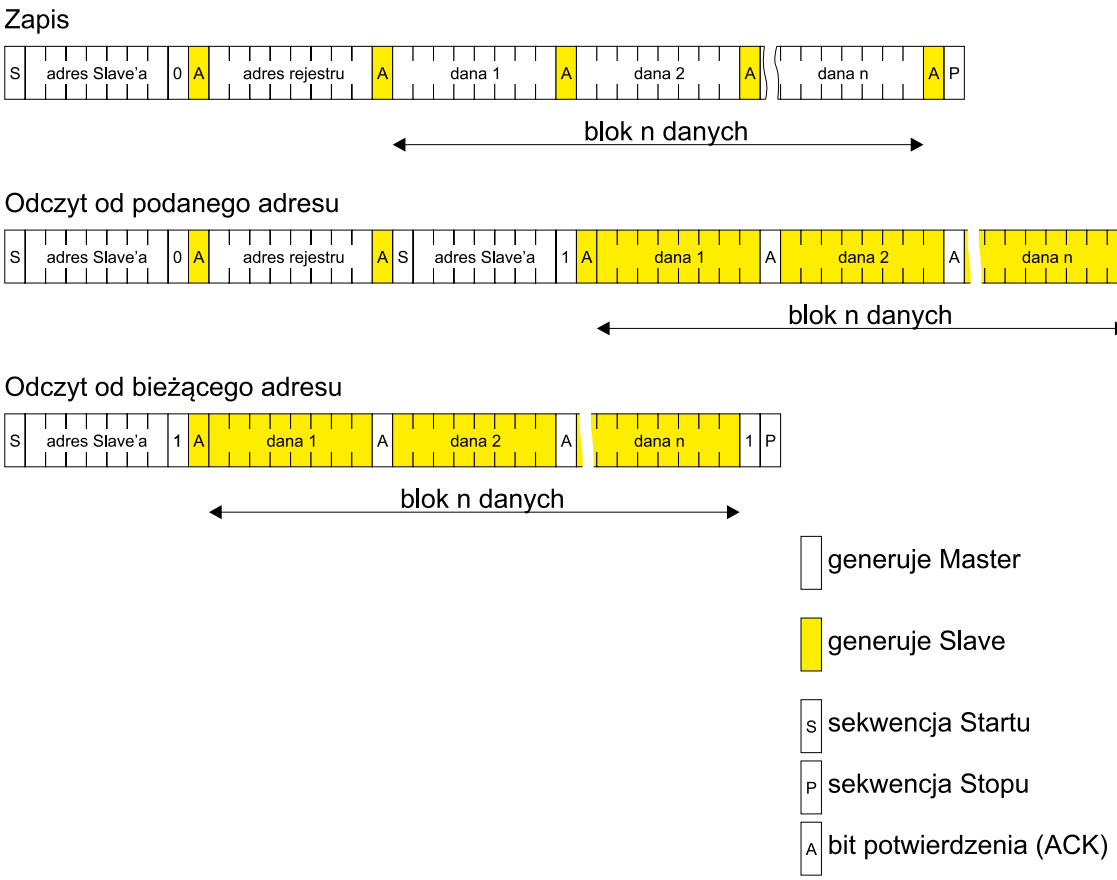
Transmisje są zawsze inicjowane przez urządzenie Master. Wysyła on adres urządzenia Slave, do którego chce wpisać, lub z którego chce czytać dane. Adres składa się najczęściej z niezmiennej części identyfikującej typ układu dołączonego do magistrali I²C oraz modyfikowalnej części identyfikującej np. konkretny układ scalony wchodzący w skład systemu mikroprocesorowego. Bezpośrednio za adresem przesyłany jest bit R/W (Read/Write) decydujący o kierunku transmitowanych danych – od urządzenia Slave do Mastera lub od Mastera do Slave.

Ze względu na rozwiązania sprzętowe przyjęte w interfejsie I²C, zastosowany w nim protokół jest dużo bardziej złożony niż w SPI. Inaczej wygląda sekwencja wymiany informacji podczas zapisu do Slave'a, inaczej podczas odczytu. W stanie bezczynności linie SCL i SDA są zwolnione (pozostają w stanie wysokim na skutek odłączenia wyjść i podciągania linii do napięcia zasilania). Transmisja jest inicjowana wygenerowaniem specjalnej sekwencji startu „S", w której sygnał SDA zmienia stan z wysokiego na niski, podczas gdy linia SCL jest utrzymywana w stanie wysokim. Następnie linia SCL przechodzi do stanu niskiego, a chwilę później linia SDA przybiera stan zgodny z wartością pierwszego bitu transmitowanej danej. Teraz Master generuje 8 impulsów zegarowych na linii SCL. Dane są zawsze 8-bitowe. Ramka jest jednak rozszerzona o dziewiąty impuls zegarowy wykorzystywany



Rysunek 10. Typowa konfiguracja urządzeń komunikujących się ze sobą przez interfejs I²C

do przesyłania bitu potwierdzającego odebranie informacji. Potwierdzeniem (ACK) odebrania informacji przez Slave'a jest wyzerowanie przez niego linii SDA na dziewiątym impulsie zegarowym. Wygenerowanie takiego potwierdzenia może wymagać pewnego czasu, układ Slave wprowadza więc Mastera w stan oczekiwania (*wait state*) przez przytrzymanie linii SDA W stanie niskim. Do momentu jej zwolnienia wszelkie operacje na magistrali I²C są wstrzymane. Brak potwierdzenia (bit ACK=1) powoduje zakończenie transmisji. Normalnie natomiast transmisja jest kończona wystawieniem sekwencji stopu — P. Jest to przejście linii SCL do stanu wysokiego podczas niskiego stanu linii SDA, a następnie ustawienie linii SDA również w stan wysoki. Protokół I²C w różnych wariantach pracy przedstawiono na **rysunku 11.** Należy zwrócić uwagę na dwukrotne przesyłanie adresu urządzenia Slave



Rysunek 11. Protokół I²C w operacjach zapisu i odczytu do/z urządzenia Slave

Rysunek 12. Zakładka konfiguracji protokołu I²C

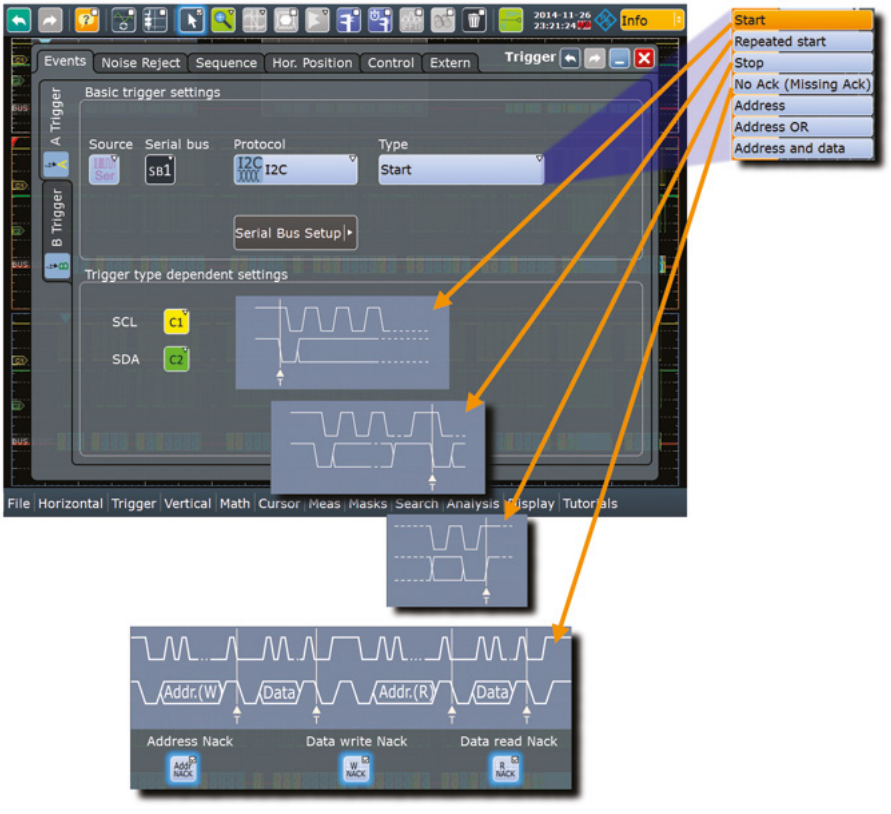
w jednej transmisji, występujące wtedy, gdy realizowany jest odczyt wskazanego rejestru układu Slave. Przed powtórnym wysłaniem adresu musi być wtedy wygenerowany tzw. powtórzony start.

Analiza protokołu I²C oscyloskopami Rohde&Schwarz

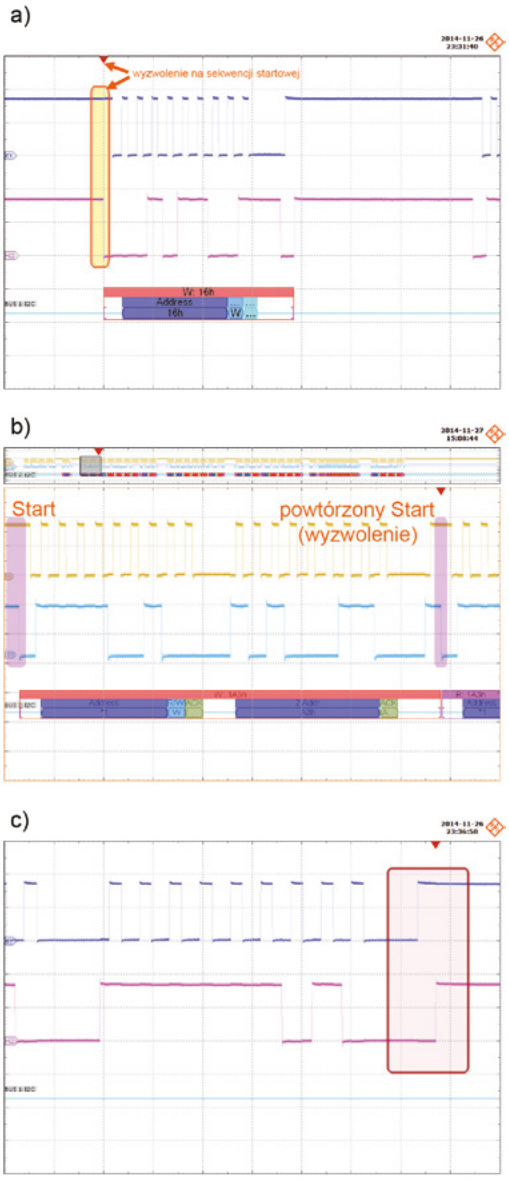
Zakładkę, na której konfiguruje się protokół I²C przedstawiono na **rysunku 12.** Czynności ograniczają się praktycznie do powiązania kanałów pomiarowych z liniami interfejsu i określenia czy bit R/W jest doklejony do adresu czy występuje w wydzielonym polu.

Możliwości analizy protokołu uwidaczniają się po naciśnięciu przycisku *TRIGGER*. Korzystając z elementów znajdujących się na wyświetlonej zakładce należy wybrać najbardziej odpowiednie zdarzenie wyzwalające, a jest z czego wybierać (**rysunek 13**).

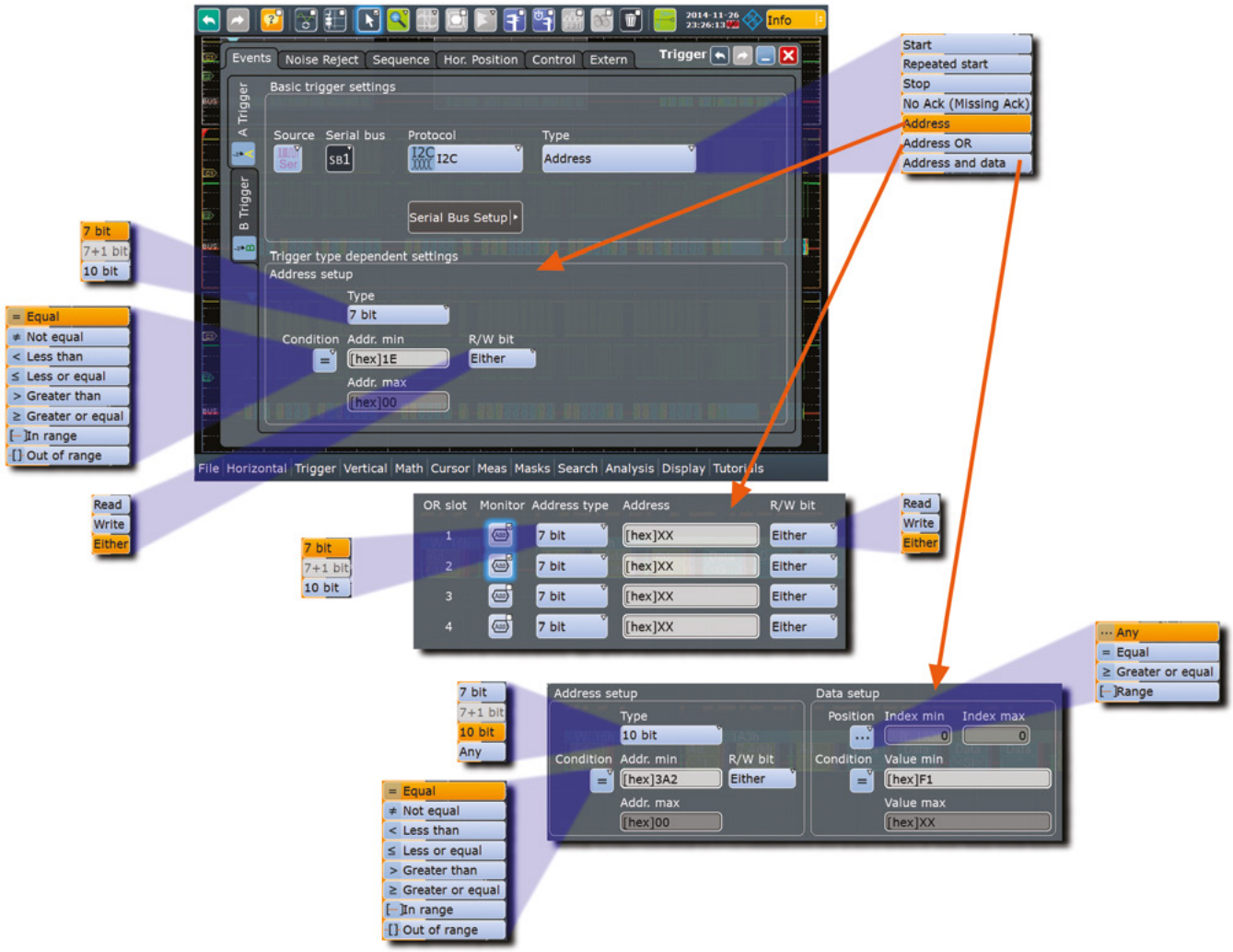
Widoczne są dwie opcje związane z sekwencją startową. Są to: sekwencja startowa rozpoczynająca transmisję i tzw. powtórzony start stosowany wtedy, gdy konieczne jest przesłanie adresu rejestru odczytywanego z urządzenia Slave. Po pierwszym starcie przesyłany jest adres Slave'a i adres rejestru, z którego będą czytane dane. W rzeczywistości są to dane przesyłane do Slave'a, więc żeby móc z niego czytać należy powtórnie wygenerować sekwencję startu z adresem Slave'a i bitem R/W oznaczającym odczyt. Dalej następuje transmisja danych trwająca do momentu, aż Slave nie wyśle potwierdzenia ACK. Brak bitu potwierdzenia oznacza, że urządzenie nie ma już danych do wysłania. Powinna być więc wygenerowana sekwencja Stop. Sekwencja ta może być również wykorzystywana jako warunek wyzwolenia. Po wybraniu każdej z tych opcji, w dolnej części zakładki jest wyświetlana grafika wyjaśniająca przebieg zdarzenia wyzwalającego. Przykłady wyzwolenia oscyloskopu sekwencją startu, powtórzoną sekwencją startu i sekwencją stopu przedstawiono na **rysunku 14.** Jeśli interfejs jest wykorzystywany intensywnie z małymi przerwami między kolejnymi transmisjami, wybór sekwencji startu lub stopu jako warunku wyzwolenia nie będzie najlepszym rozwiązaniem, gdyż trudno je będzie rozróżnić. W tym przypadku znacznie efektywniejsze będzie wyzwalanie np. po wykryciu konkretnego adresu urządzenia (**rysunek 15**). Możliwości konfiguracji układu wyzwalającego jest sporo.



Rysunek 13. Zakładka wyboru zdarzenia konfigurującego – opcje startu i stopu

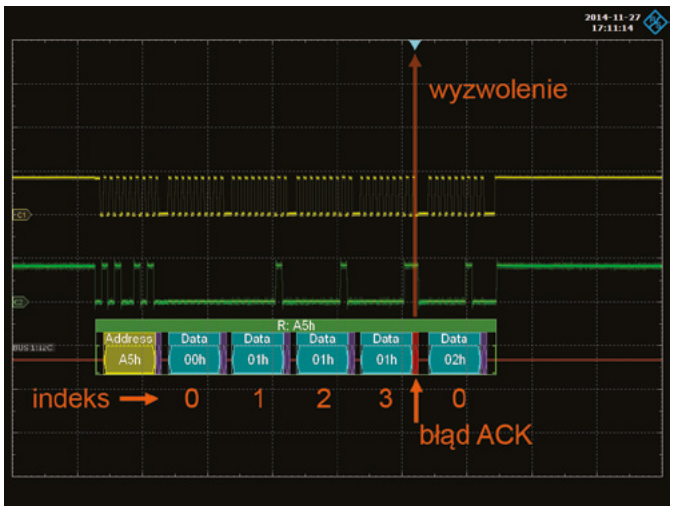


Rysunek 14. Przykłady wyzwolenia na: a) sekwencji startu, b) powtórzonej sekwencji startu, c) sekwencji stopu



Rysunek 15. Zakładka wyboru zdarzeń wyzwalających – opcje wykrywania adresu i danej

Adres może być określany jako 7–bitowa liczba bez uwzględniania występującego za nim bitu R/W, albo łącznie z bitem R/W, można też wybrać 10-bitową wersję adresu. Następnie należy wybrać relację (mniejszy, równy, większy itp.), jaką powinien spełniać adres w odniesieniu do ewentualnego parametru. Na końcu należy zdecydować, czy ma być uwzględniany kierunek transmisji (Read, Write) czy nie (Either). Można też wyzwalać oscyloskop po znalezieniu konkretnej danej występującej na wskazanej pozycji w bloku adresowanym do podanego urządzenia. Do ustawienia parametrów wyzwalania należy wówczas wybrać opcję „Adress and data". Zasada doboru parametrów jest podobna jak w poprzedniej metodzie. Na **rysunku 16** przedstawiono wyzwolenie oscyloskopu po wykryciu danej równej 01h o indeksie równym 3. Dana ta jest przesyłanej do urządzenia o adresie A5h. W chwili wyzwolenia można przy okazji zaobserwować brak potwierdzenia ACK, które powinno być wygenerowane przez urządzenie Slave.



Rysunek 16. Przykłady wyzwolenia na określonej danej występującej na wskazanej pozycji w bloku kierowanym do konkretnego urządzenia

Wnioski

Interfejsy SPI i I²C są obecnie, wobec odchodzącego do lamusa RS232, najczęściej stosowanymi rozwiązaniami wykorzystywanymi przede wszystkim do realizacji wewnętrznej komunikacji pomiędzy poszczególnymi blokami funkcjonalnymi urządzeń elektronicznych. Trzeba jednak pamiętać, że chociaż coraz rzadziej mamy do czynienia z samym interfejsem RS232, to stosowany w nim protokół jest nadal wykorzystywany i znamy go pod nazwą UART. Nadal jest to jeden z najczęściej wykorzystywanych protokołów asynchronicznych. Wymienione protokoły (UART, SPI, I²C) są na tyle uniwersalne, że można jest implementować w większości sprzętu elektronicznego powszechnego użytku i profesjonalnego. Tam, gdzie rosną wymagania funkcjonalne lub stawiane są dodatkowe obostrzenia parametrów technicznych trzeba sięgać po protokoły/interfejsy bardziej ukierunkowane na konkretne zastosowania. Będzie o tym mowa w kolejnych odcinkach.

Jarosław Doliński, EP