Wykład 7.

Magistrala I2C - szeregowa, dwukierunkowa magistrala przesyłu danych. Związany jest z nią standard I2C, który określa warstwę fizyczną i warstwę łącza danych w modelu warstwowym komunikacji sieciowej *OSI*. Powszechnie stosowana, używana m.in. do wykrywania modułów w pamięci – jakiego jest rodzaju, jak jest duża, itp. Używana w **HDMI, PCI, PCIExpress**.

Cechy charakterystyczne

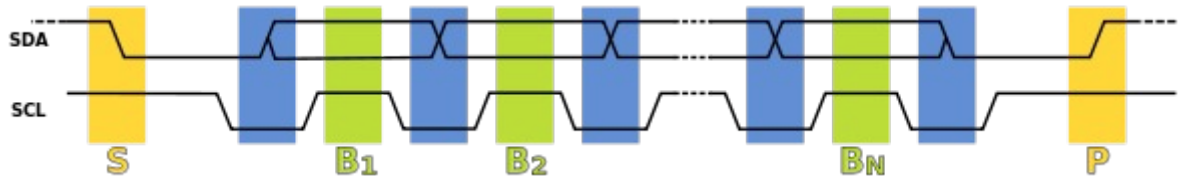
- Jej zaletą w stosunku do UART i SPI jest ***mała liczba potrzebnych przewodów do zaimplementowania komunikacji między dużą liczbą urządzeń.***
- I2C nie transmituje dość szybko, więc od tej magistrali ***nie należy oczekiwać szybkiej transmisji***.

Rodzaje urządzeń występujących na magistrali I2C

- EEPROM – przez I2C jest robiona konfiguracja, można również uzyskać dane na temat podłączonego urządzenia
- Czujniki temperatury przyprocesorowe (odczyt temperatury rdzenia)
- Przetworniki audio – wymagają dwóch magistrali: I2CS (transmisja dźwięku), I2C (transmisja metadanych: liczba kanałów, bit rate, sample rate, etc.)
- Zegary czasu rzeczywistego
- Czujniki przyspieszenia i odległości w robotyce
- Sterowanie diodami LED w smartfonach
- Komunikacja pomiędzy układami w telewizorach i innym sprzęcie RTV

Warstwa fizyczna

- Dwa przewody dwukierunkowe:
  - **SDA** – linia danych
  - **SCL** – linia zegara (synchronizacja)

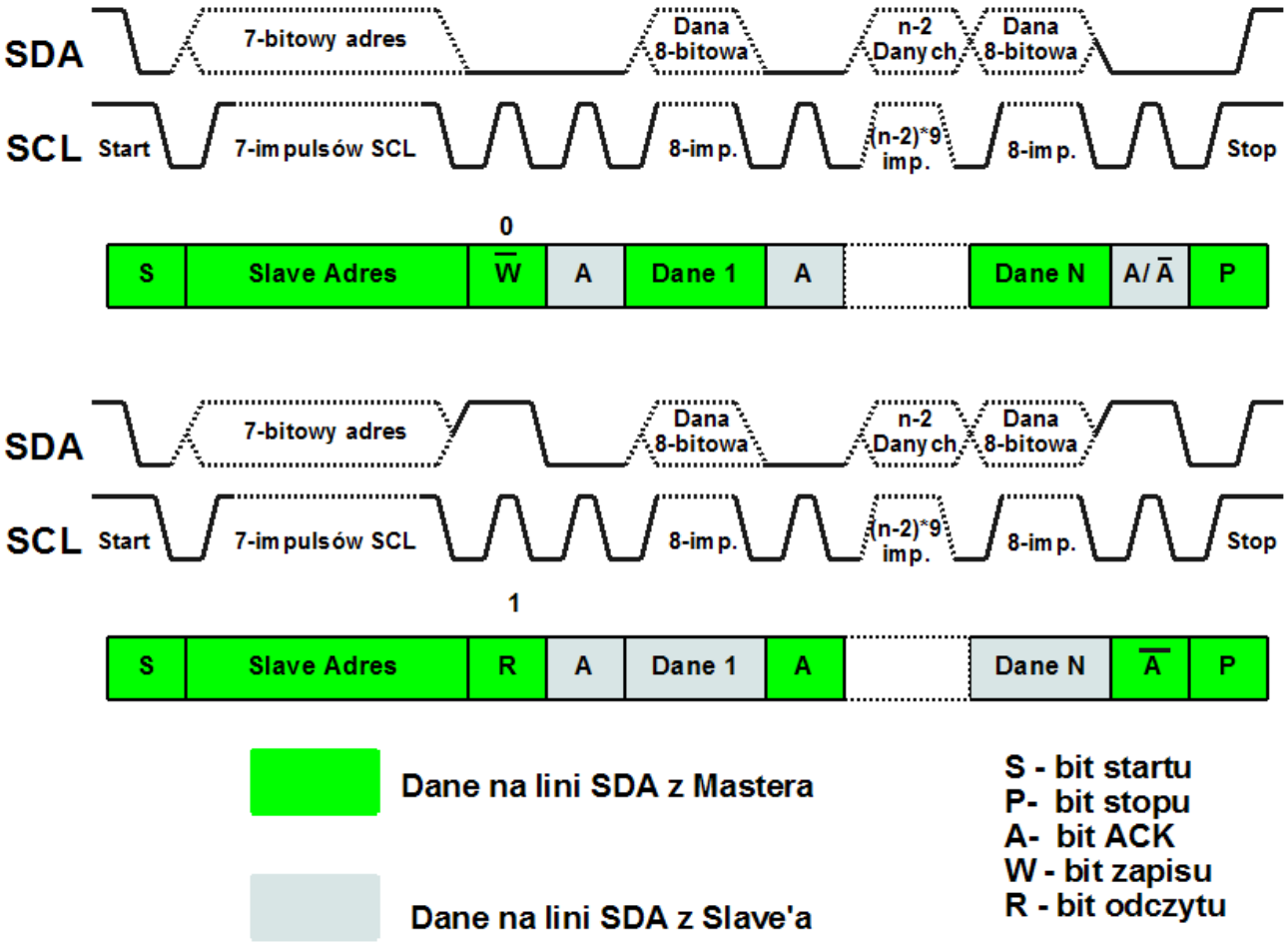


Rys. 1. Przebieg transmisji z wykorzystaniem linii SDA i SCL

- **Obydwie linie są na stałe podciągnięte do źródeł prądowych poprzez rezystory pull-up**
- Jest typu *master-slave (tylko master jest nadajnikiem)*
- Każde urządzenie podłączone do magistralli musi mieć tranzystor od strony masy. Obecność większej liczby urządzeń niedaleko siebie oznacza obecność dużej liczby źródeł prądowych i rezystorów pull-up niedaleko siebie, co powoduje powstanie pojemności spowalniającej zmiany stanu napięcia układu z 0 do 1 (0V → 5V)
- Rezystory pull-up mają wartości typowo od setek do dziesiątek tysięcy Ohm. Nie mogą być zbyt małe ze względu na duże straty transmisji (transmisja nie jest zbyt szybka)
- Podstawowa wersja I2C zakłada pracę w trybie pojedynczego mastera, ale jest możliwy tryb multi-master:
- Tryb wielu masterów = wiele nadajników:
  - Każdy nadajnik jest typu **otwarty kolektor** lub **otwarty dren**, co powoduje, ze na liniach pojawia się tzw. **iloczyn na drucie** (1 – recesywne, 0 – dominujące). Tworzy to mechanizm wykrywania kolizji: każde urządzenie nadając ‘1’ jednocześnie sprawdza, czy na magistrali rzeczywiście pojawił się stan wysoki. Jeśli tak nie jest, to znaczy, że inne urządzenie nadało ‘1’ w tym samym czasie i urządzenie zaprzestaje nadawania.
- Data order – od najstarszego (MSB) do najmłodszego bitu (LSB)
- W przypadku jednoczesnego nadawania urządzenie nadające adres rozgłoszeniowy slave o wyższym numerze wycofa się pierwsze,a to nadające adres niższy zostanie. Jest to tzw. **arbitraż ze stałym przydziałem priorytetów** – urządzenia o niższych adresach mają wyższy priorytet od urządzeń o wyższych adresach.
- W szczególnych przypadkach, gdy dwóch masterów nadaje pod ten sam adres slave, priorytety rozróżnia się następująco:
  - wygrywa ten, który zapisuje
  - odczytać mogą oboje naraz
  - jeśli chcą zapisać to samo naraz – też mogą

- W przypadku zapisów różnych danych – albo zapobiegamy takim sytuacjom, albo implementujemy mechanizm, który je rozstrzygnie

- Długość linii magistrali jest ograniczona maksymalną pojemnością, która wynosi **400 pF**, co w praktyce ogranicza długość linii do **kilku m**.
- Transmisja dwóch 0 V na dwóch odbiornikach nie przeszkadza jak w przypadku SPI



Rys. 2. Przebieg transmisji w magistrali I2C – widok szczegółowy

Warstwa łącza danych

- Tak samo jak w SPI czy UART, dane są grupowane bajtowo – przesyłane w pakietach po 8 bitów
- 7-bitowa przestrzeń adresowa odbiorników (slaveów). Możliwość zaadresowania 112 urządzeń (16 adresów jest zarezerwowanych, jeden z nich to tzw. **general call** – adres 0, który powoduje wysłanie danych do wszystkich odbiorników).

- ✗ Pierwszy nadawany bajt to zawsze adres slave. Po przesłaniu bajtu przesyłany jest bit potwierdzenia ACK (lub NACK, gdy się nie uda) w kierunku przeciwnym.
- ✗ Kolejne wersje standardu umożliwiają rozszerzenie przestrzeni adresowej do 10 bitów. W tym przypadku pierwszy przesłany bajt zawiera 5 z góry ustalonych bitów oraz dwa najstarsze bity adresu 10 bitowego. W drugim bajcie przesyła się już 8 bitów adresu.
- ✗ Kolejne przesyłane bajty to są już właściwe dane

Magistrala TWI w ATMEGA 328P

- To skopiowana magistrala I2C pod inną nazwą, aby uniknąć opłat licencyjnych

Przebieg transmisji

- Zanim zaczniemy cokolwiek transmitować, musi być spełniony **warunek startu**:
  - zbocze opadające na SDA
  - stan wysoki na SCL
- Po jego spełnieniu rozpoczyna się cykl transmisji 1 bitu, powtarzany 8 razy:
  - SCL jest opuszczany do stanu niskiego na chwilę do momentu rozpoczęcia transmisji od mastera na SDA
  - jeśli SCL uzna, że transmisja przebiegła prawidłowo, wraca do stanu wysokiego
- Każdy odbiornik podłączony do szyny magistrali nasłuchuje swojego adresu. Jeśli slave rozpozna swój adres, to nasłuchuje kolejnego **bitu b7** informującego o rodzaju transmisji – odczyt/zapis
- Następnie, po otrzymaniu bitu b7, slave nadaje **bit potwierdzenia ACK**, że rozpoznał swój adres i rodzaj transmisji i jest gotowy do odbioru.
- Zapis: master przesyła ciąg impulsów na linii zegara, oraz bity na linii danych tak samo, jak powyżej. Na koniec otrzymuje bit potwierdzenia od slave.
- Odczyt: master nadaje po linii zegara, ale slave nadaje dane, master tylko słucha.

Clock stretching

- Jest wtedy, gdy slave nie nadąża za masterem. Przykładowo bufor zapisu po stronie EEPROM się już zapełnił, i slave musi wysłać masterowi, by poczekał trochę na zwolnienie bufora. Wtedy master wyłącza swój tranzystor w momencie zablokowania zegara przez slave, ale dalej obserwuje linię. Gdy slave zwalnia zegar, master kontynuuje.

Wzór na częstotliwość zegara SCL

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot (PrescalerValue)}$$

- TWBR – wartość rejestru TWI Bit Rate Register – dzielnik dla generatora częstotliwości zegara w postaci 1 bajtowej liczby.
- PrescalerValue – wartość dwóch bitów 1:0 z TWI Status Register. Możliwe wartości: 1, 4, 16, 64

Wykład 8.

System czasu rzeczywistego (RTC)– system, który zapewnia odpowiednio szybką reakcję na bodziec w rzeczywistym świecie. Innymi słowy, stosowany wtedy, gdy zależy nam na realizacji zadań w ściśle określonych ramach czasowych. Przykład: system sterujący maszyną CNC.

Rodzaje systemów czasu rzeczywistego

- **miękki** - system odpowiada w rozsądnym czasie, ale może być trochę zmienny i niewielkie odchyły są akceptowalne. Przykłady: odtwarzanie multimediów, system zbierania danych pogodowych, system sprzedaży biletów w kasach biletowych
- **twardy** – czasy są ściśle przestrzegane. Niezareagowanie w porę oznacza katastrofę. Przykład: systemy przemysłowe, system podtrzymywania życia, system kontrolny ABS w samochodzie, sekwencja awaryjnego wyłączania silnika rakietowego

Wielozadaniowość – cecha, która umożliwia wykonywanie wielu zadań naraz. Typowo mikrokontroler ma jeden procesor, lecz za pomocą planisty może być wielozadaniowy. Algorytm szeregowania zadań stanowi istotną część planisty przydziału pracy procesora.

Wielozadaniowość z wywłaszczaniem

- Może nastąpić przerwanie wykonania procesu, czyli odebranie mu procesora i przekazanie sterowania do planisty (co się nazywa **wywłaszczaniem**).
- Pełne wywłaszczanie zapewniają tylko mechanizmy sprzętowe działające niezależnie od oprogramowania. Realizowane przez dołączanie wywłaszczania do procedury obsługi przerwania zegarowego.
- Wywłaszczanie najczęściej następuje po kwancie czasu. W przypadku mikrokontrolerów można też samemu zdecydować, kiedy wywłaszczyć dane zadanie.

Wielozadaniowość kooperatywna

- Proces sam zgłasza swoją gotowość do zmiany kontekstu poprzez wywołanie systemowe funkcji planisty.
- Następnie pozostaje w uśpieniu do czasu, aż znów zostanie mu przydzielony czas procesora

FreeRTOS – to twardy system operacyjny czasu rzeczywistego pod warunkiem spełnienia odpowiednich kryteriów pisanych programów.

Po co używać FreeRTOS w programach na mikrokontrolery

- Bardziej złożonych programów nie pisze się w pętli głównej programu.
- Istnieje potrzeba posiadania choćby niewielkiego podzbioru systemu operacyjnego, aby móc strukturalizować nasze programy.

Cechy charakterystyczne FreeRTOS

- planowanie przydziału procesora w oparciu o priorytety z kwantem czasu
- nie można blokować przerwania i procesu bezczynności (IdleHook)

- programista sam musi zadbać o odpowiedni dobór priorytetów pod kątem ewentualnego głodzenia procesów o niskich priorytetach przez wysokie
- procedura **taskYield** wywołuje algorytm szeregowania.
- **IdleHook** - procedura bezczynności, w niej można uśpić procesor
- implementuje struktury danych: **kolejki, liczniki programowe, semafory, muteksy**

Stany zadań

Zadanie może mieć 1 z 4 stanów:

- Running - tylko 1 zadanie może mieć ten stan w danym momencie – stan bycia wykonywanym
- Ready - czekanie na bycie wykonanym
- Blocked - wykonujacy się proces może być zablokowany - TaskDelay, proces może czekać na wejście, przyjście znaku z magistrali.
- Suspended - stan opcjonalny, proces jest tam umieszczany ręcznie. Możemy tam umieścić proces, który będzie miał pracować tylko wtedy, gdy ma coś do roboty. Np. funkcja pauzy w maszynie, gdy coś chcemy wymienić.

Kolejki

- Służą do komunikacji pomiędzy procesami. To współdzielony fragment pamięci chroniony przez muteks/semafor i o określonym porządku dodawania/zdejmowania elementów
- Rodzaj kolejki: FIFO (taka sama jak np. kolejka w sklepie – kto pierwszy przyjdzie, ten pierwszy jest obsłużony)
- Istnieje specjalny wariant nieblokujący procedury umieszczającej bajt w kolejce
- Przykładowe operacje: push, pop, is_empty, is_full

Semafory

- chroniona zmienna będąca kontrolą dostępu do wspólnych zasobów w wielozadaniowym świecie procesów
- nie przenosi mechanizmu dziedziczenia priorytetów
- Rodzaje semaforów:
  - binarny - zwykła flaga

- zliczający - do pewnej wartości. Przydatne, gdy mamy kilka zasobów i po ich wzięciu chcemy zablokować zadanie (trzeba zliczyć ile razy został podniesiony)

Muteksy

- szczególny rodzaj semafora binarnego, używany w celu uniknięcia równoczesnego wspólnego zasobu przez odwołujące się procesy w sekcji krytycznej
- Implementuje mechanizm dziedziczenia priorytetów
- Służa tylko do komunikacji między zadaniami
- Nie można ich używać w przerwaniach i idlehook.

Liczniki programowe

FreeRTOS posiada liczniki programowe, które nie zużywają ekstra zasobów jak w przypadku liczników sprzętowych. Wszystkie liczniki programowe współdzielą kod procedury obsługi licznika - nie ma sensu blokowania procedury obsługi licznika.

Różnica pomiędzy vTaskDelay a delay_ms

- **delay_ms:** w przypadku delay_ms procesor liczy n jałowych iteracji (*nop*) w pętli. W świecie wielozadaniowości to system operacyjny musi wyliczyć tę wartość w kontekście pracy innych zadań w stosunku do naszego – nie jesteśmy w stanie określić tej wartości, więc użycie tej procedury nie ma sensu w tym przypadku
- **vTaskDelay**: Algorytm szeregowania patrzy na zbiór zadań gotowych do wykonania. vTaskDelay informuje algorytm, że przez pewien czas zadanie jest gotowe do wykonania. Powrót gotowości jest wykonywane w przerwaniu zegarowym procesora.