

NobleProg



Interfejsy szeregowo w mikrokontrolerach

Grzegorz Mazur

NobleProg



Plan prezentacji

- Interfejsy szeregowe ogólnie
- UART
- SPI
- I2C
- OneWire
- Interfejsy specyficzne dla układów



Interfejsy szeregowe

- Interfejsy, w których poszczególne bity słowa danych są przesyłane kolejno, bit, po bicie, po tych samych liniach sygnałowych
- Tradycyjna klasyfikacja i.s.
 - Synchroniczne – z przesyłanym oddzielną linią sygnałową sygnałem synchronizującym (zegarem)
 - Asynchroniczne – nie wymagające przesyłania sygnału zegarowego
- Współcześnie wiele interfejsów przesyła dane wraz z sygnałem synchronizującym po tej samej linii – sygnał zegarowy jest zakodowany w sygnale danych (np. OneWire, WS2812)



UART

- Najprostszy interfejs szeregowy
 - W wersji minimalnej – tylko dwie linie sygnałowe TxD, RxD
- Asynchroniczny – nie wymaga transmisji przebiegu taktującego
 - Komunikujące się urządzenia muszą znać szybkość transmisji
 - Wzorce częstotliwości obu urządzeń, określające szybkość transmisji, muszą mieć precyzję nie gorszą niż 2%

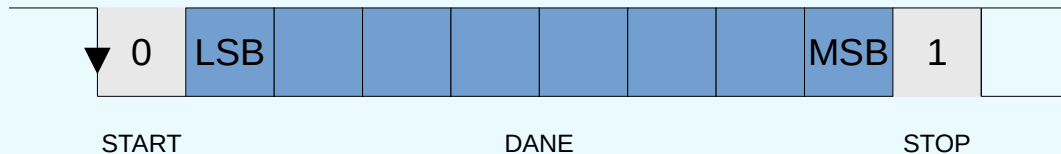


UART - transmisja

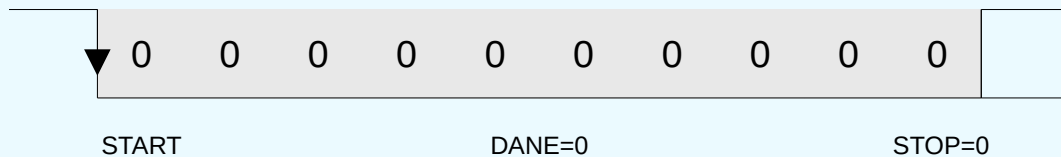
- Szybkość transmisji determinuje czas trwania bitu
- Początek transmisji sygnalizowany zmianą stanu linii – początek „bitu startu”
- Po wykryciu zbocza bitu startu odbiornik odlicza pół czasu transmisji bitu, weryfikuje ważność bitu startu, a następnie próbkuje stan linii w odstępach równych okresom bitów („w środku” bitu)

UART

Ramka danych



Symbol Break





UART – logika transmisji danych

- Stan linii:
 - Nieaktywny (SPACE) – 1
 - Aktywny (MARK) – 0
- Format ramki danych:
 - Start (1 bit o wartości 0)
 - Dane – 5..8 bitów, przesyłane w kolejności od najmniej znaczącego bitu
 - Kontrola parzystości lub znacznik – brak lub 1 bit
 - Stop – 1 lub 2 bity o wartości 1



UART – bit stop

- Bit stop ma wartość SPACE (linia nieaktywna)
- Przy ciągłej transmisji wielu ramek służy do poprawnego wykrycia bitu startu kolejnej ramki
- Wykrycie stanu MARK w miejscu bitu stop oznacza błąd (np. niezgodność szybkości nadajnika i odbiornika)



UART – stany specjalne

- Bezczynność (IDLE) – stan SPACE trwający dłużej niż określony czas (zależny od realizacji, > czasu transmisji ramki)
 - Niekiedy używany jako znacznik końca transmisji pakietu danych
- Przerwa (BREAK) – stan MARK trwający dłużej niż czas transmisji pełnej ramki (z bitami stopu)
 - Niekiedy używany jako znacznik początku



UART – wykrywanie błędów

- Fałszywy start – stan MARK trwający krócej niż czas transmisji jednego bitu
- Zakłócenia – zmieniająca się wartość linii RxD w połowie czasu transmisji bitu
 - Odbiornik UART próbkuje wartość linii RXD trzykrotnie; zakłócenie oznacza niezgodność trzech próbek
- Błąd parzystości – niewłaściwa wartość bitu parzystości
- Błąd ramki – stan MARK w czasie transmisji bitów stopu (sygnałizowany również przy BREAK)
- Przepiętnienie (overrun) – dana nieodczytana na czas z odbiornika, kolejna ramka została zgubiona



Standardy elektryczne UART

- RS-232 i podobne (EIA/TIA-562)
 - Sygnalizacja napięciowa, linia asymetryczna
- RS485, RS422
 - Linia symetryczna, dopasowana
- Pętla prądowa
- UART na poziomach logicznych TTL/CMOS



RS-232, TIA-232-F, TIA-562

- RS-232 – popularna, nie do końca poprawna nazwa
 - W rzeczywistości wiele standardów o podobnej charakterystyce elektrycznej, różnych złączach i zbiorach sygnałów
- Aktualne standardy
 - TIA-232-F – bieżąca wersja podstawowa, złącze DB25
 - TIA-574 – złącze DE9
 - EIA/TIA-561 – złącze 8P8C (RJ45)
 - EIA/TIA-562 – wersja niskonapięciowa
- Pierwotnie było to połączenie komputera z modemem
 - Zestaw sygnałów odpowiada logice modemów z lat 1970-tych



RS-232 – urządzenia DTE i DCE

- DTE – Data Terminal Equipment – urządzenie nadrzędne; zwykle komputer
- DCE – Data Communication Equipment – urządzenie podrzędne; dawniej modem
- Obecnie zwykle łączymy dwa urządzenia DTE, co nie jest objęte standardami
 - Zmieniona funkcjonalność linii
- Interfejsy UART w komputerach i mikrokontrolerach używają terminologii DTE
- Moduły GSM są urządzeniami DCE

RS-232 - sygnały

	DTE DCE	
GND		Masa sygnałowa
TXD	→	Dane nadawane przez DTE
RXD	←	Dane odbierane przez DTE
RTS	→	Gotowość DTE do nadawania
CTS	←	Gotowość DCE do odbioru danych z DTE
DTR	→	Sygnalizacja włączenia DTE
DSR	←	Sygnalizacja włączenia DCE
CD	←	DCE gotowy do transmisji danych do DTE
RI	←	(Dzwonienie) – DCE nawiązuje połączenie



Logika sygnałów RS232

- Wszystkie linie poza RXD/TXD są zanegowane
- Poziomem aktywnym wszystkich linii jest poziom niski
- W RS-232 poziomowi niskiemu odpowiada napięcie dodatnie, a wysokiemu – ujemne



Standard napięciowy RS-232-F i nowsze

- Sygnały bipolarne
- Odbiór:
 - 0 powyżej 3 V
 - 1 poniżej -3V
 - W praktyce odbiorniki RS-232 pracują z progiem ok. +2 V!
- Nadawanie – w nowszych wersjach standardu mniejsza amplituda
 - RS-232-F ± 5 V
 - TIA-562 ± 3.3 V
 - TIA-694 ± 3 V, odbiór ± 2 V



Synchronizacja transmisji w RS-232

- W praktyce UART służy zwykle do połączenia dwóch równoprawnych urządzeń pełniących rolę DTE
- Istotną informacją dla nadawcy jest gotowość odbiorcy do odbioru danych
 - Wejście nadawcy: CTS
 - Wyjście odbiorcy: linia RTS zmienia znaczenie na RTR (Ready To Receive)
- Linie RXD/TXD i RTR/CTS są łączone „na krzyż”



TIA-232 – szybkości transmisji

- Maksymalne szybkości wg standardu:
 - TIA-232-F: 20 kb/s
 - TIA-562: 64 kb/s
 - TIA-694: 512 kb/s
 - Większość układów nadajników/odbiorników gwarantuje szybkości do 120 kb/s, niektóre do 250 kb/s lub 1 Mb/s
- Powszechnie używane szybkości [b/s]:
 - Dawne: 110, 150, 300, 600, 1200, 2400, 4800
 - Tradycyjne, używane: 9600, 19200, 38400, 57600, 76800, 115200
 - Dodatkowo dostępne w PC: 230400, 460800, 921600



RS485

- Standard transmisji umożliwiający pracę wielu urządzeń z interfejsami UART na jednej szynie (typowo do 32)
- Linia symetryczna, różnicowa
- Możliwy zasięg:
 - 1200 m przy 100 kb/s
 - 12 m przy 10 Mb/s
- Nowe układy – do 100 Mb/s



Protokoły logiczne korzystające z RS485

- Profibus
- Fieldbus
- DMX512
- Modbus



UART w STM32

- „ułamkowe podzielniki” - umożliwiają uzyskanie dokładnej częstotliwości transmisji przy dowolnej częstotliwości zegara
- Możliwość transmisji przy użyciu DMA



Programowanie UART w STM32

- BRR – dzielnik
- CR2 – opcje dodatkowe
- CR3 – opcje dodatkowe i DMA
- Na końcu – CR1 – tryb pracy, włączanie przerwań, włączenie nadawania/odbioru, włączenie UART

```
USART1->BRR = (PCLK_FREQ + BAUD / 2) / BAUD;  
USART1->CR1 = USART_CR1_TE | USART_CR1_RE | USART_CR1_UE;
```



STM32 – ciekawe opcje w CR2

- Negacja linii RX, TX
 - umożliwia współpracę z RS232 bez transceivera (konieczny rezystor szeregowy $\geq 47k$ na linii RX)
 - Ułatwia wstawienie np. transoptora
- Zamiana linii RX i TX – kiedy pomylimy się rysując schemat...



UART – źródła przerwań

- Wolne miejsce w buforze nadajnika – TXE
- Dane w buforze odbiornika – RXNE
- Błędy odbioru – FE, ORE, NF
 - włączane razem z RXNE
- Koniec transmisji – TC (przydatne dla RS485)
- Przerwa w odbiorze – IDLE
- ... i kilka innych



UART – obsługa przerwań

- Przy włączonym RXNE konieczne sprawdzenie NF, FE i ORE
- Zgłoszenie przerwania TXE kasowane wyłącznie przez zapis danych
 - Przerwanie TXE należy włączać w UART wyłącznie wtedy, gdy są gotowe dane do wysłania, a wyłączać – przed zapisem ostatniej danej



UART i DMA

- Pożyteczne przy nadawaniu
- Problematiczne przy odbiorze



Dodatkowe linie UART w STM32

- -CTS (wejście) – możliwość sprzętowego blokowania nadawania
- -RTS (wyjście, w rzeczywistości jest to -RTR) – gotowość do odbioru
- DE – sprzętowe sterowanie kierunkiem bufora dla RS485 (tylko w nowszych wersjach UART – F0, L0, L4, niedostępne w F1, F4)



UART w praktyce

- Odbiór: Ponieważ odbiorca nie ma kontroli nad odbieranym strumieniem danych, najczęściej niskopoziomowa interpretacja danych (składanie w ramki, edycja wiersza tekstu, weryfikacja poprawności) na ogół realizuje się w funkcji wywoływanej z obsługi przerwania odbioru znaku, pisanej w konwencji automatu
 - W zasadzie bezużyteczne są funkcje typu „odbierz N znaków”
 - Bufor cykliczny może być pożyteczny przy niejednostajnym tempie interpretacji danych (np. drukarka)
- Nadawanie:
 - Przesyłanie bloków/komunikatów – można użyć przerwania lub DMA
 - Przy nierównomiernym strumieniu (pojedyncze znaki i dłuższe komunikaty) pożyteczne jest zastosowanie bufora cyklicznego pełniącego rolę kolejki danych, opróżnianego w przerwaniu zwolnienia bufora nadawania UART



UART i HAL STM32

- Praktycznie nieużywalne mechanizmy odbioru danych UART w HAL oraz wspólne przerwania nadawania i odbioru UART w mikrokontrolerach STM32 znacznie ograniczają możliwości zastosowania HAL do obsługi UART
- Często stosowane rozwiązanie (z wadami)
 - Użycie funkcji HAL_UART_Transmit_IT() lub HAL_UART_Transmit_DMA() do nadawania z programowym testem gotowości lub ustawianiem znacznika w wywołaniu zwrotnym TransmitCplt
 - Użycie funkcji HAL_UART_Receive_IT() do odbioru pojedynczych znaków z wywołaniem zwrotnym, w którym znajduje się interpretacja znaku i powtórne wywołanie tej funkcji



SPI, Microwire,

- Grupa interfejsów międzyukładowych o wspólnej ogólnej charakterystyce
 - Transmisja synchroniczna, równocześnie w obu kierunkach
 - Sygnał zegarowy SCK generowany przez układ nadrzędny
 - Dwie linie danych MOSI i MISO (Master Out, Slave In, Master In, Slave Out)
 - Sygnał wyboru/uaktywnienia urządzenia NSS (Slave Select)



SPI

- Transmisja zawsze dwukierunkowa
 - W każdym cyklu zegara Master nadaje i odbiera jeden bit
 - Slave może ignorować lub nie odbierać danych, ale Master o tym nie wie
 - Slave może nie nadawać danych, ... ale Master o tym nie wie
 - Master odbiera tyle bitów, ile nadał
 - Slave jest zawsze gotowy do transmisji



SPI

- Na szynie SPI może być wiele układów podrzędnych
 - Często stosowane rozwiązanie
 - Każdy układ podrzędny ma własną, oddzielną linię NSS
- Jest możliwa konfiguracja z wieloma układami nadrzędnymi
 - Rzadko stosowana



SPI – wybór układu podrzędnego

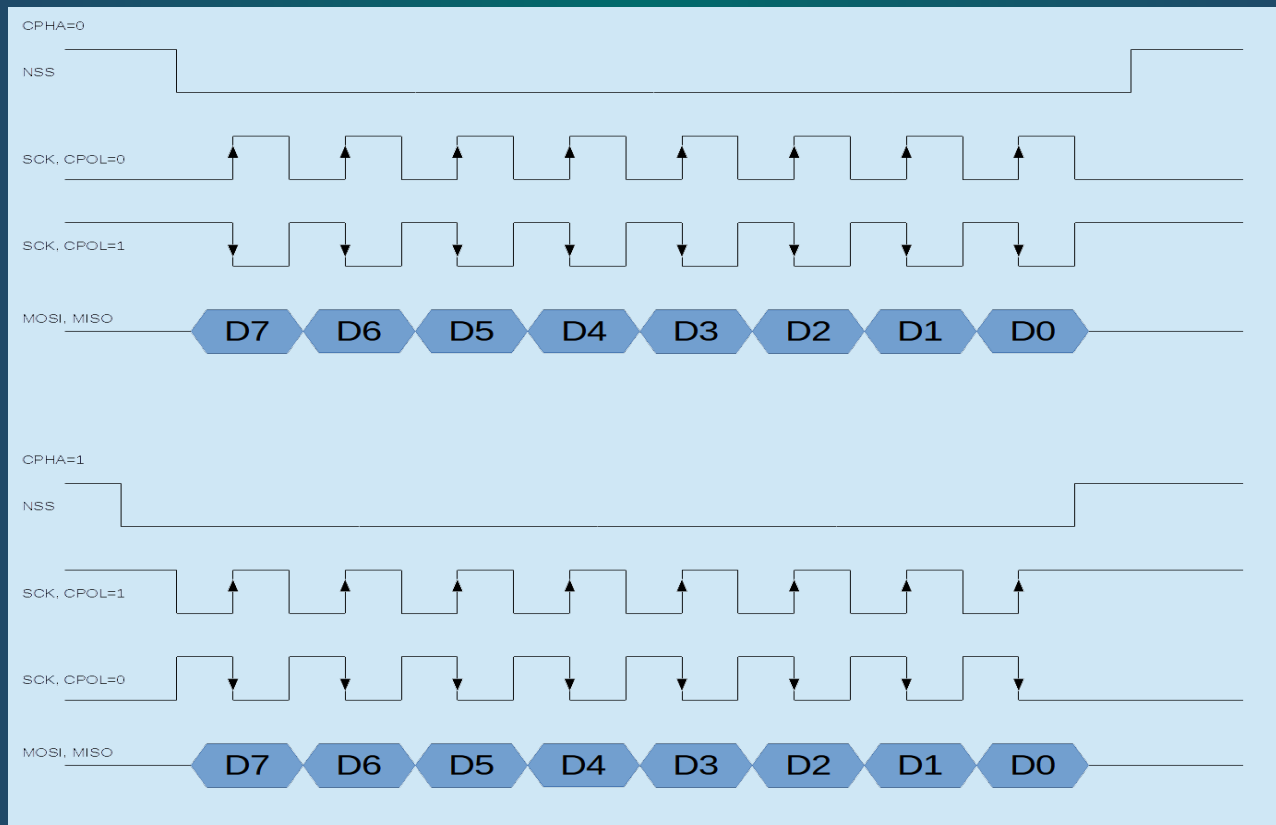
- Uaktywnienie linii NSS oznacza wybór układu i początek transmisji ramki
- Niektóre układy Slave przy konfiguracji z jednym Slave mogą pracować jako stale aktywne (NSS stale aktywne lub brak linii NSS)
 - Wtedy początek ramki jest sygnalizowany przez określony wzorzec danych na linii MOSI, np. jedynka po serii zer



Parametry CPOL i CPHA

- CPOL – poziom linii SCK w stanie beczynności (polaryzacja zegara)
- CPHA – faza zegara
 - 0 – dane próbkowane pierwszym zboczem i kolejnymi nieparzystymi
 - 1 – dane próbkowane drugim zboczem i kolejnymi parzystymi
- Większość układów peryferyjnych może działać przy CPOL=0 i CPHA=0 albo CPOL=1 i CPHA=1

SPI





SPI w STM32

- Dwie wersje
 - Uboższa 8/16 bit – F2, F4
 - Bogatsza 4..16 bit – F0, L0, L4...
- Uwaga – przy danych ≤ 8 b:
 - Dostęp do rejestru DR muszą być jawnie 8-bitowe z atrybutem volatile (domyślnie są volatile 16-bitowe)
 - Musi być ustawiony bit FRXTH w rejestrze CR2
- Różne zachowanie sprzętowego sterowania linią NSS w różnych rodzinach mikrokontrolerów
 - sprawdzić w Reference Manual
 - W wielu przypadkach niezbędne jest sterowanie przez oprogramowanie, z linią NSS zaprogramowaną jako wyjście GPIO



SPI w STM32

- Częstotliwość transmisji – podzielniki tylko 2^n w stosunku do zegara szyny peryferiala
 - Jeżeli niezbędna jest konkretna wartość częstotliwości – należy odpowiednio dobrać częstotliwość taktowania μC !
- SSM, SSI – ważne bity w rejestrze CR1
- Wykrywanie zakończenia transmisji – bity RXNE i/lub BSY w rejestrze stanu
- Ze względu na dużą szybkość transmisji SPI, użycie przerwań do obsługi SPI powoduje zwykle wysycenie mocy obliczeniowej – praktycznie cały czas procesora jest zużywany na wejście i wyjście z procedury obsługi przerwania
- Warto używać DMA, nawet przy małych rozmiarach bloków danych



SPI - inicjowanie

- CR2 – opcje, w tym:
 - Długość ramki danych
 - FRXTH – ustawić 1 przy ramkach do 8 bitów!
 - DMA
- CR1
 - Dzielnik częstotliwości, włączenie - SPE
- Zmiana ustawień w CR1, CR2 możliwa tylko przy wyzerowanym SPE



I2C (TWI)

- Interfejs międzyukładowy, synchroniczny
- 2 linie
 - SCL – zegar
 - SDA - dane
- Szyna z możliwością adresowania – może łączyć wiele układów
- Transmisja zwykle znacznie wolniejsza niż w SPI



I2C - komunikacja

- Linie SCL i SDA są typu „otwarty dren”
- Niezbędne rezystory podciągające 2..10k
- Podczas transmisji danych stan SDA zmienia się przy nieaktywnym (wysokim) stanie SCL
- Zmiana stanu SDA przy niskim SCL oznacza:
 - Z 1 na 0 – START
 - Z 0 na 1 – STOP



I2C - komunikacja

- Transmisję taktuje układ nadrzędny
- Układ podrzędny może spowolnić transmisję, „przytrzymując” linię SCL w stanie niskim

μC

I2C

- Ramki 8-bitowe tworzące pakiety
- Po każdej ramce następuje bit potwierdzenia, nadawany przez odbiorcę ramki
 - 0 – ACK – pomyślny odbiór, gotowość
 - 1 – NACK – brak odpowiedzi, niegotowość lub błąd



I2C – komunikacja

- Pakiet rozpoczyna się od symbolu START
- Pierwszy oktet pakietu jest nadawany zawsze przez MASTER i zawiera adres układu podrzędnego oraz bit określający kierunek transmisji następnych oktetów
 - Oktet adresu jest potwierdzany przez zaadresowany układ podrzędny
 - Brak potwierdzenia = brak układu o danym adresie lub niegotowość



I2C

- Kolejne bajty pakietu są transmitowane w kierunku określonym przez pierwszy bajt
- Zmiana kierunku wymaga ponownego rozpoczęcia pakietu
- Transmisja kończy się symbolem specjalnym STOP



I2C

- Zazwyczaj transakcja I2C składa się z jednej lub dwóch faz: zapisu lub zapisu i odczytu
 - W fazie zapisu podaje się np. polecenie i adres wewnątrz układu potrzebne do późniejszego odczytu, wtedy po fazie zapisu nie ma STOP
- `<START><addr+dir><dane><START><addr+dir><dane><STOP>`



I2C w STM32

- Złożony peryferial, skomplikowane oprogramowanie
 - Wiele stanów, złożone sekwencje czynności
 - Specjalna obsługa transmisji dwóch ostatnich bajtów



Oprogramowanie I2C

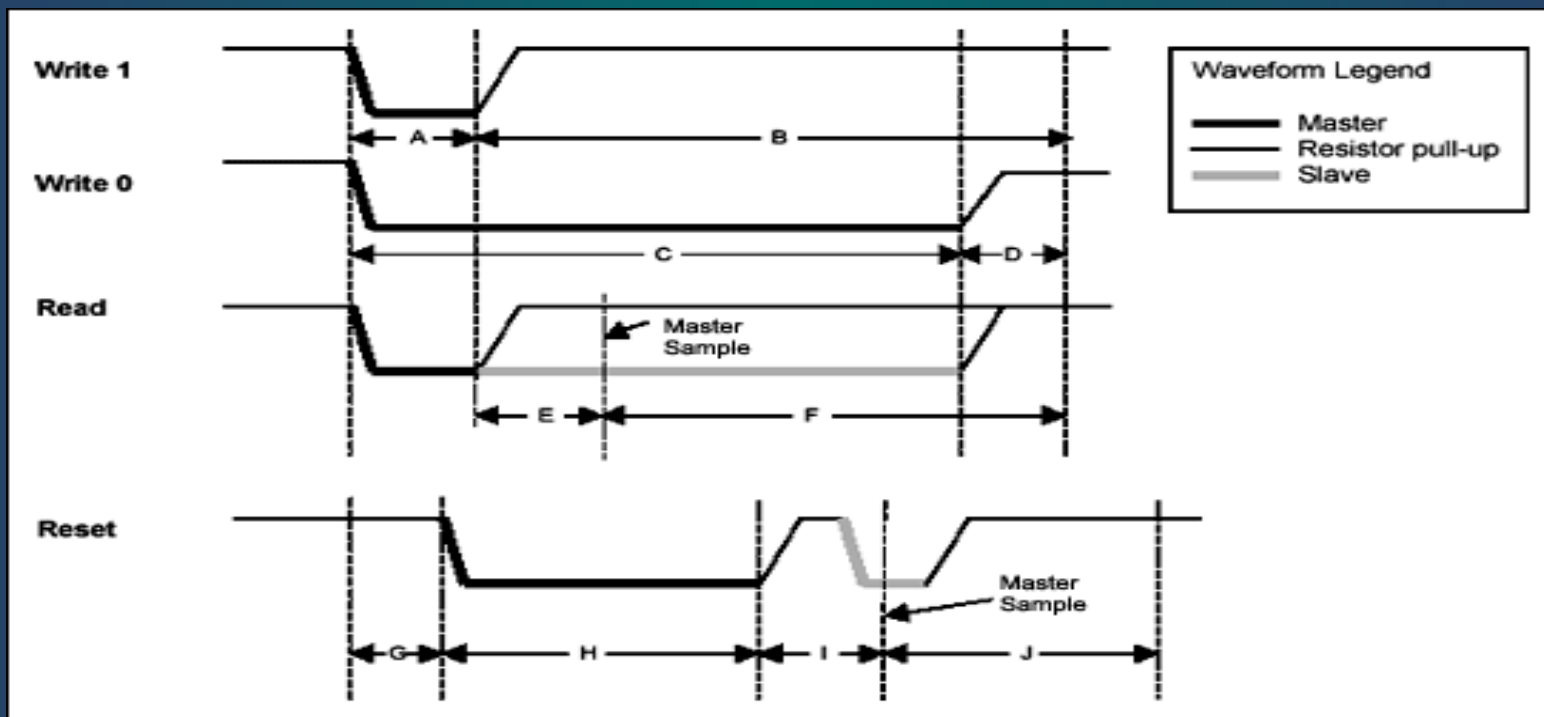
- Użyjemy biblioteki HAL
- Pożyteczne funkcje: MemoryWrite, MemoryRead – zapewniają współpracę z większością układów I2C
- Funkcje HAL STM32 dostępne w 3 wersjach:
 - Podstawowe – blokujące, synchroniczne
 - Z przyrostkiem `_IT` – asynchroniczne z użyciem przerwań, zalecane do użycia przy I2C
 - Z przyrostkiem `_DMA` – z użyciem przerwań i DMA
 - Zalecane użycie funkcji działających z użyciem przerwań, a w przypadku długich bloków danych (np. pamięci EEPROM) – DMA



OneWire

- Standard firmy Dallas (obecnie Maxim)
- Dwukierunkowa transmisja po jednej linii
- Opcjonalne zasilanie układów z linii danych
- Możliwe połączenie wielu układów na jednej linii
- Układy mają nadane fabrycznie unikatowe adresy 64-bitowe
- Możliwe wykrycie adresów wszystkich układów na szynie

OneWire – komunikacja





OneWire - realizacja

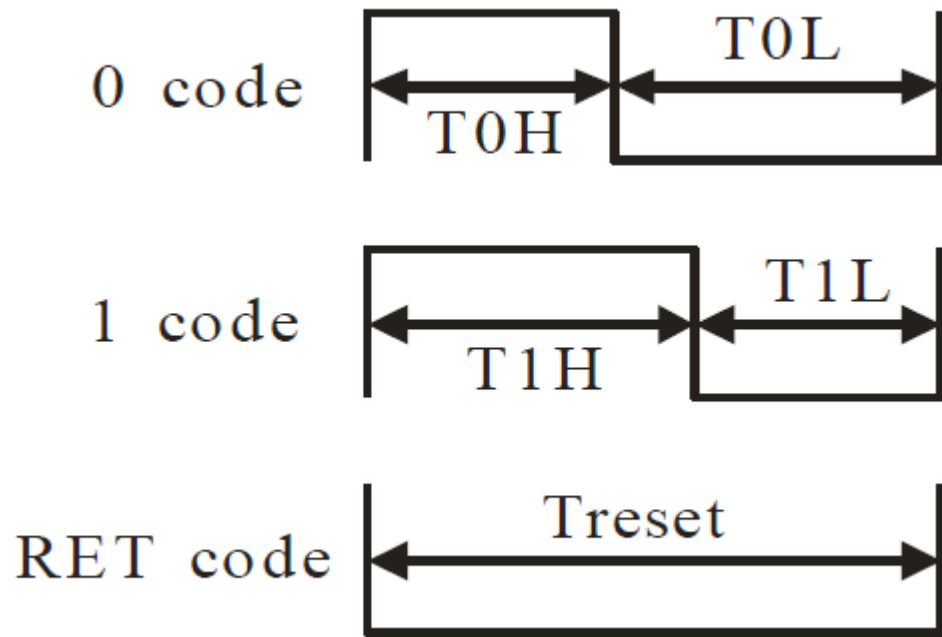
- Brak obsługi sprzętowej w mikrokontrolerach z powodów licencyjnych
- Możliwe realizacje:
 - Całkowicie programowa z opóźnieniami
 - Timer wielokanałowy, przerwania
 - 2 lub 3 przerwania na bit
 - UART – zwarte RX i TX, 1 bit na ramkę, konieczna zmiana szybkości dla RESET



OneWire - oprogramowanie

- Dwu- lub trójpoziomowy automat
 - Poziom bitu
 - Poziom transakcji
 - Poziom logiczny – sekwencja transakcji

WS2812 – komunikacja





WS2812 – realizacje na STM32

- Timer
 - Tryb PWM, ładowanie wypełnienia przez DMA
 - Okres 3, wypełnienie 1 lub 2
 - Każdy bit kodowany słowem 16-bitowym (wypełnienie PWM) – duża zajętość pamięci



WS2812 – realizacje na STM32

- Użycie interfejsu szeregowego
 - Każdy bit WS2812 kodowany na trzech bitach – 0:100, 1:110
- SPI
 - Używana tylko linia MOSI
 - Wymagana częstotliwość 2.4 MHz +- 10%
 - 8 bitów WS2812 przesyłanych w trzech ramkach 8-bitowych SPI
- UART
 - Wymagana negacja linii TXD – stanem nieaktywnym musi być 0
 - 3 bity WS2812 kodowane w ramce UART z 7 bitami danych (+ START i STOP – łącznie 9 bitów nadawanych w ramce UART)
 - Przydatne opcje dostępne w nowszej wersji UART STM32 – negacja linii i ew. odwrócenie kolejności bitów (STM32G)