Systemy wbudowane - wykład 7

Przemek Błaśkiewicz

11 kwietnia 2019

I^2C aka IIC aka TWI

- Inter-Integrated Circuit

I^2C aka IIC aka TWI

- Inter-Integrated Circuit
- używa dwóch linii przesyłowych i wykorzystuje architekturę master/slave

I^2C aka IIC aka TWI

- Inter-Integrated Circuit
- używa dwóch linii przesyłowych i wykorzystuje architekturę master/slave
- jednostki mają adres (7 lub 10 bitów)

I^2C aka IIC aka TWI

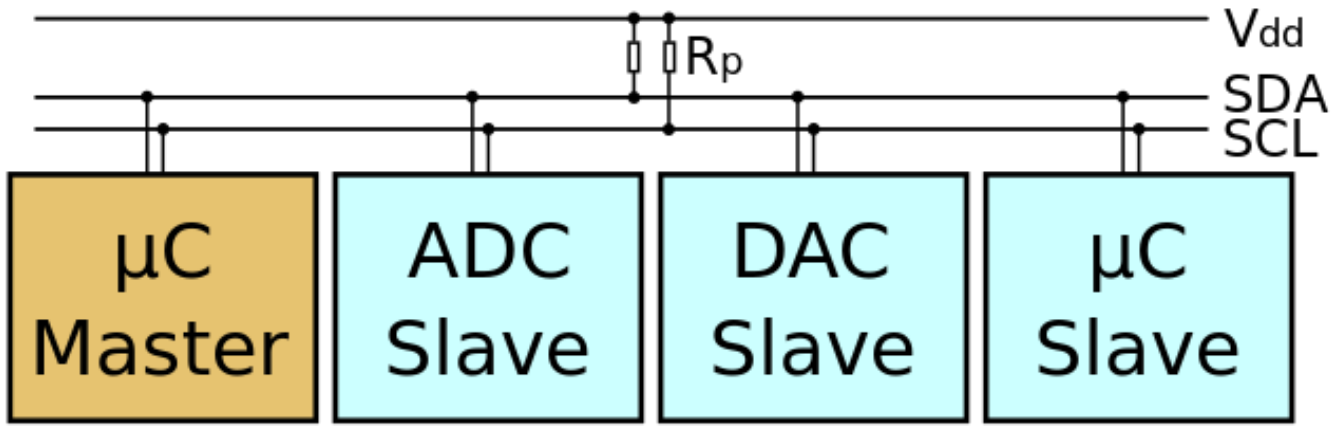
- Inter-Integrated Circuit
- używa dwóch linii przesyłowych i wykorzystuje architekturę master/slave
- jednostki mają adres (7 lub 10 bitów)
- typowe transfery 100 kbit/s (standard), 10 kbit/s (low speed), ale można używać dowolnych częstotliwości (do 3.4Mbps w ostatniej specyfikacji)

I^2C aka IIC aka TWI

- Inter-Integrated Circuit
- używa dwóch linii przesyłowych i wykorzystuje architekturę master/slave
- jednostki mają adres (7 lub 10 bitów)
- typowe transfery 100 kbit/s (standard), 10 kbit/s (low speed), ale można używać dowolnych częstotliwości (do 3.4Mbps w ostatniej specyfikacji)
- liczba urządzeń na magistrali ograniczona adresacją i pojemnością (elektryczną) linii

I^2C aka IIC aka TWI

- Inter-Integrated Circuit
- używa dwóch linii przesyłowych i wykorzystuje architekturę master/slave
- jednostki mają adres (7 lub 10 bitów)
- typowe transfery 100 kbit/s (standard), 10 kbit/s (low speed), ale można używać dowolnych częstotliwości (do 3.4Mbps w ostatniej specyfikacji)
- liczba urządzeń na magistrali ograniczona adresacją i pojemnością (elektryczną) linii



I2C - schemat komunikacji

1. Master wysyła bit START oraz 7 bitów adresu slave

I2C - schemat komunikacji

1. Master wysyła bit START oraz 7 bitów adresu slave
2. Master wysyła bit 0 lub 1 (chęć pisania lub czytania slave)

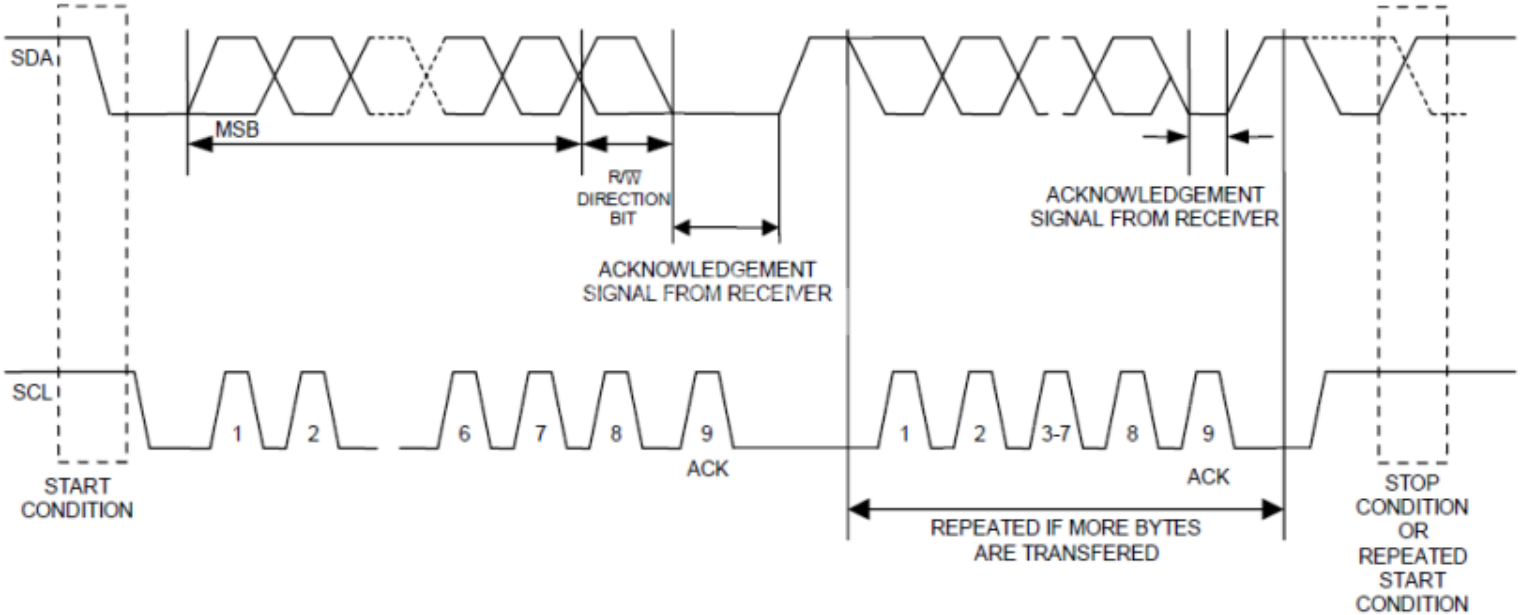
I2C - schemat komunikacji

1. Master wysyła bit START oraz 7 bitów adresu slave
2. Master wysyła bit 0 lub 1 (chęć pisania lub czytania slave)
3. Slave odpowiada (jeśli jest) bitem ACK

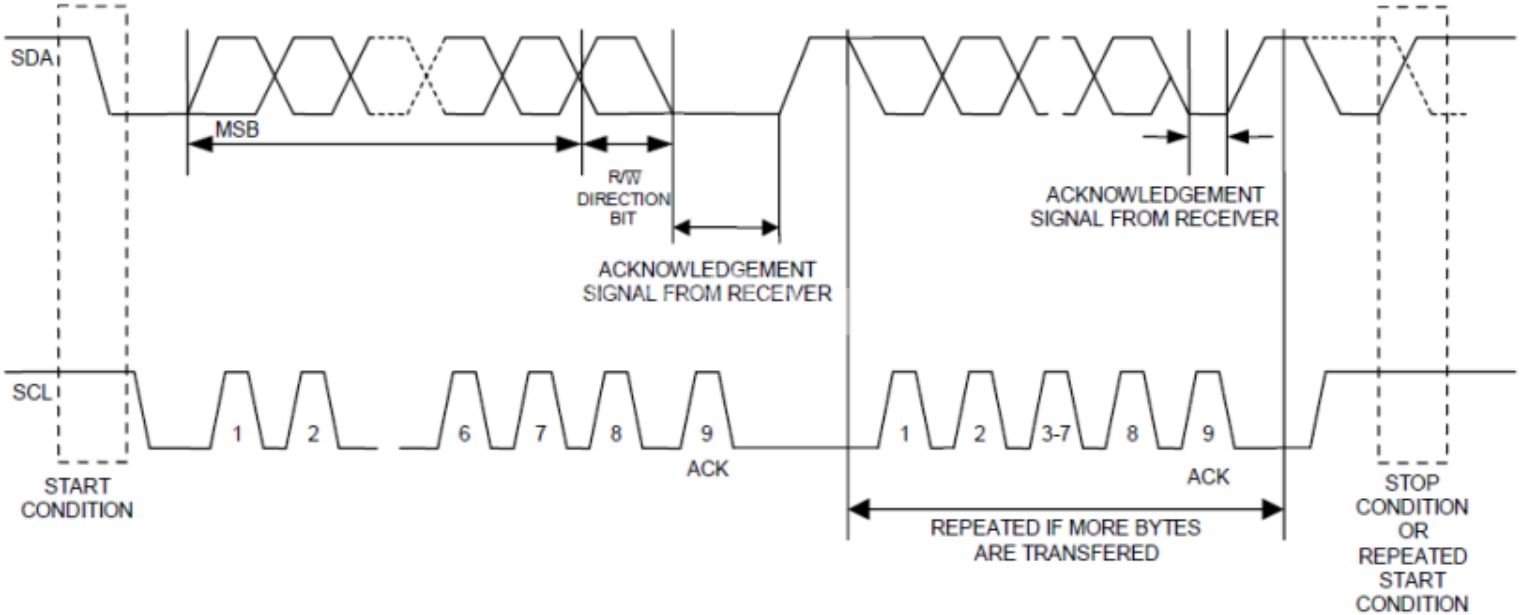
I2C - schemat komunikacji

1. Master wysyła bit START oraz 7 bitów adresu slave
2. Master wysyła bit 0 lub 1 (chęć pisania lub czytania slave)
3. Slave odpowiada (jeśli jest) bitem ACK
4. Dane są przesyłane (MSB najpierw)

I2C – timing

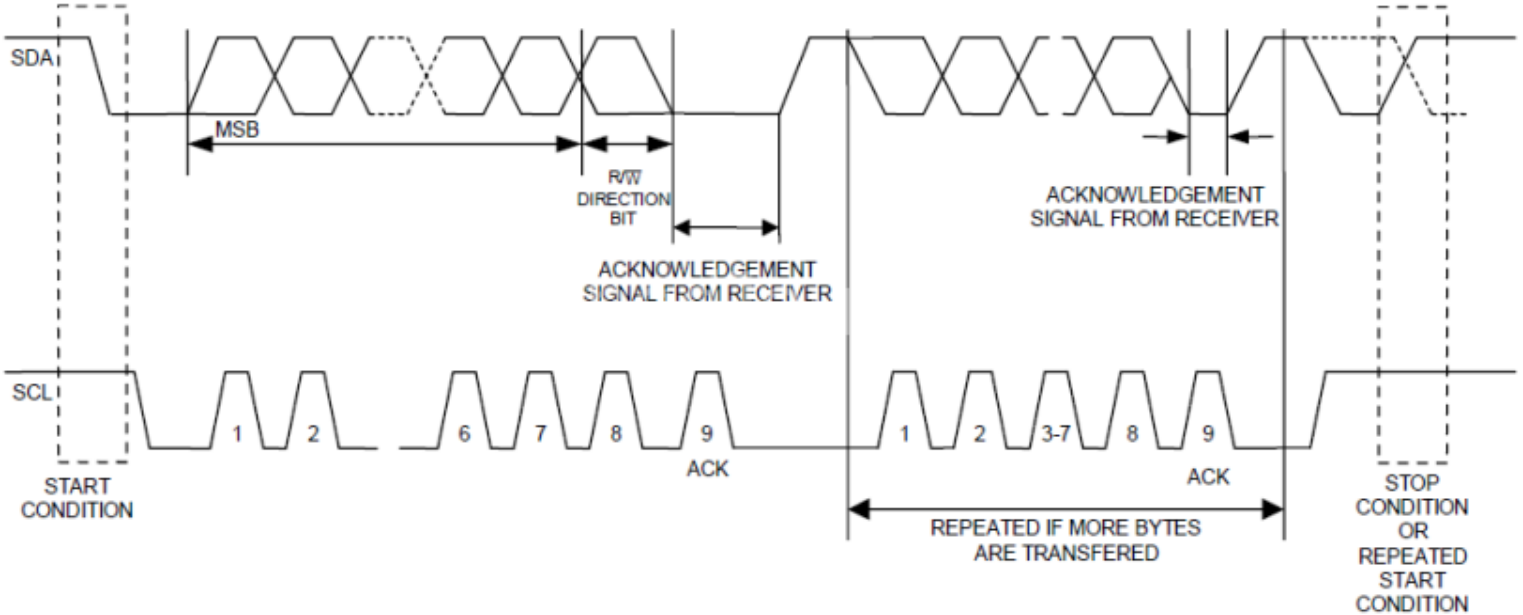

SDA
MSB
R/W
DIRECTION
BIT
ACKNOWLEDGEMENT
SIGNAL FROM RECEIVER
ACKNOWLEDGEMENT
SIGNAL FROM RECEIVER
SCL
1
2
6
7
8
9
ACK
1
2
3-7
8
9
ACK
START
CONDITION
REPEATED IF MORE BYTES
ARE TRANSFERED
STOP
CONDITION
OR
REPEATED
START
CONDITION

I2C – timing



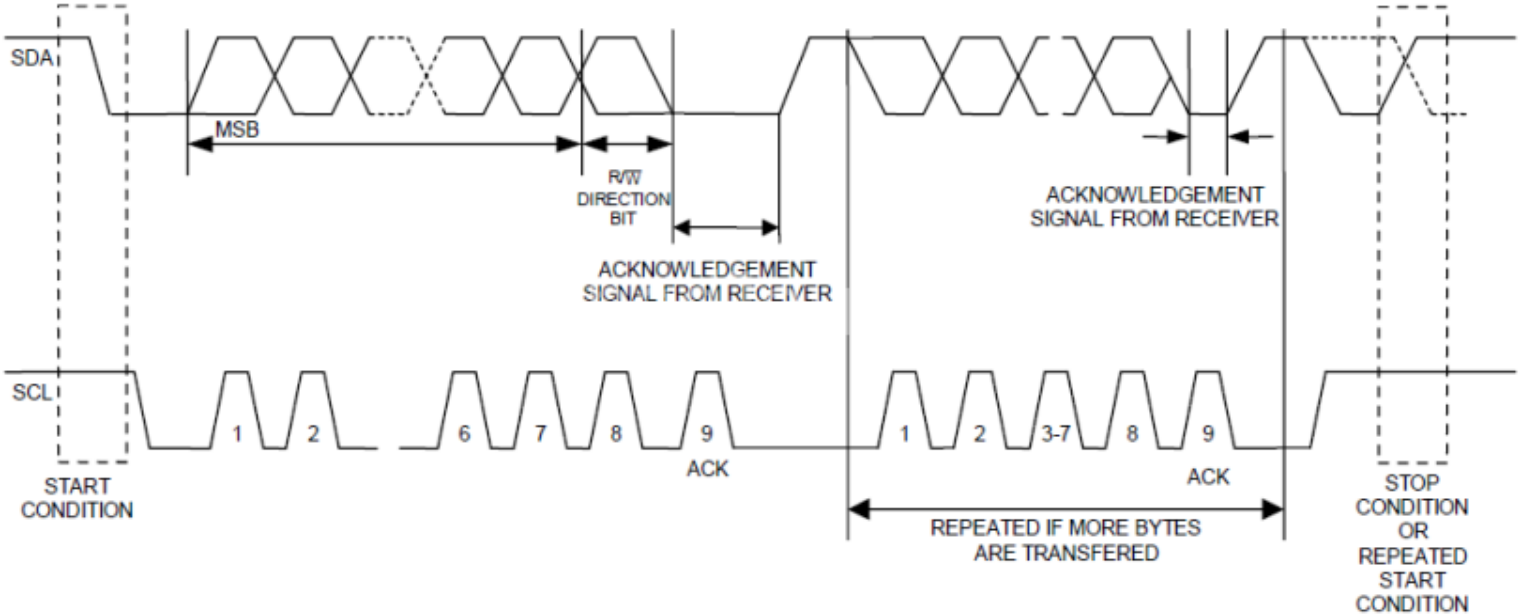
- bit START : SCL Hi oraz SDA $1 \rightarrow 0$;

I2C – timing



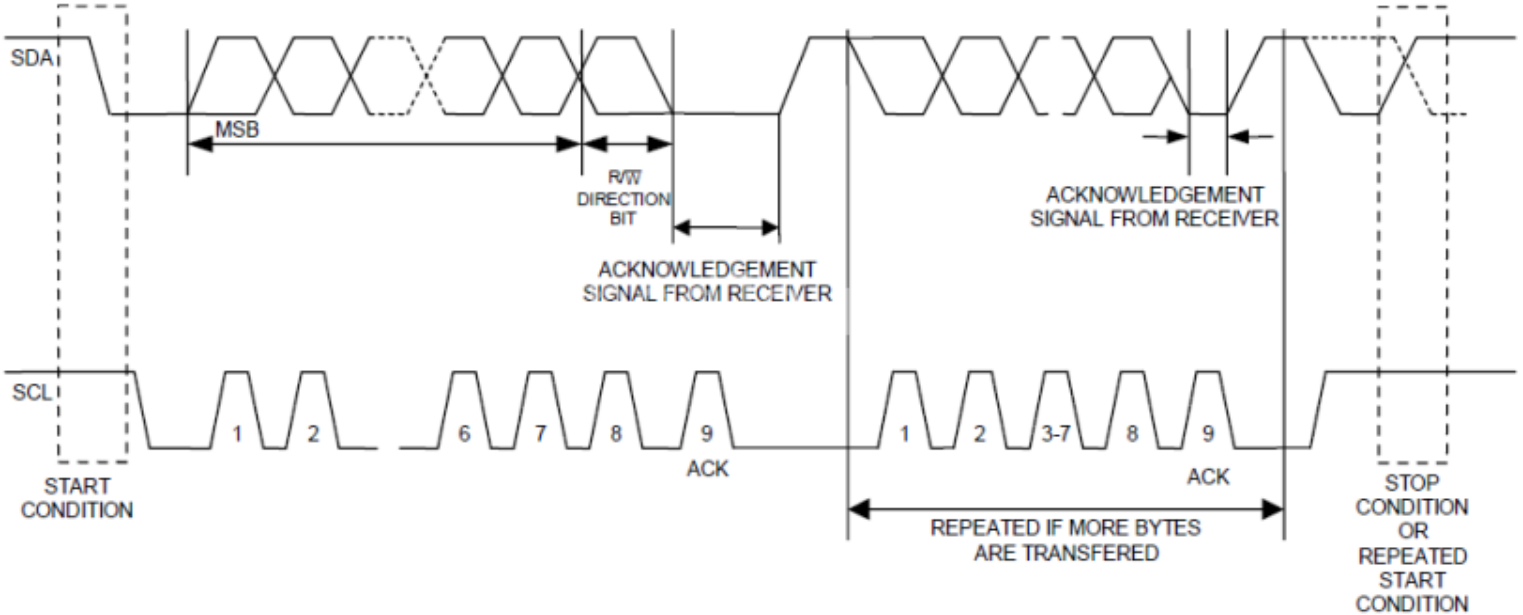
- bit START : SCL Hi oraz SDA $1 \rightarrow 0$;
- bit STOP : SCL Hi oraz SDA $0 \rightarrow 1$

I2C – timing



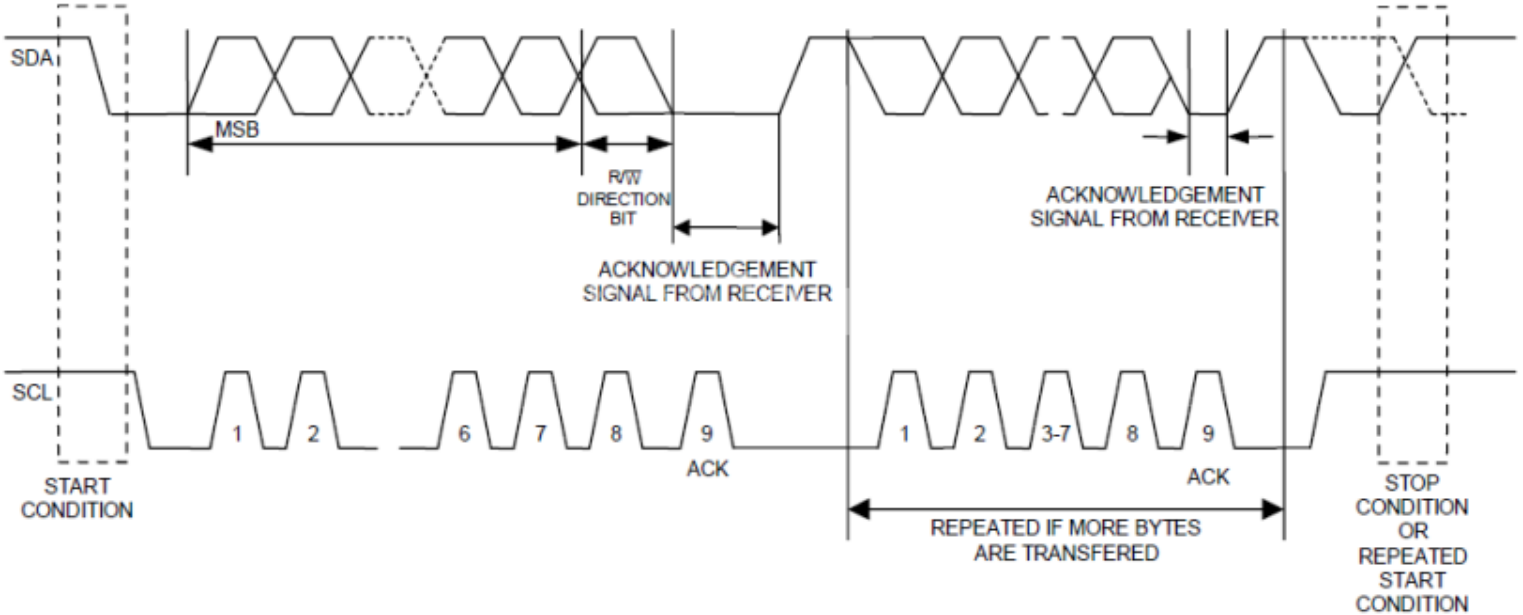
- bit START : SCL Hi oraz SDA $1 \rightarrow 0$;
- bit STOP : SCL Hi oraz SDA $0 \rightarrow 1$
- (przy transmisji danych) SDA jest ustalone przy SCL Hi

I2C – timing



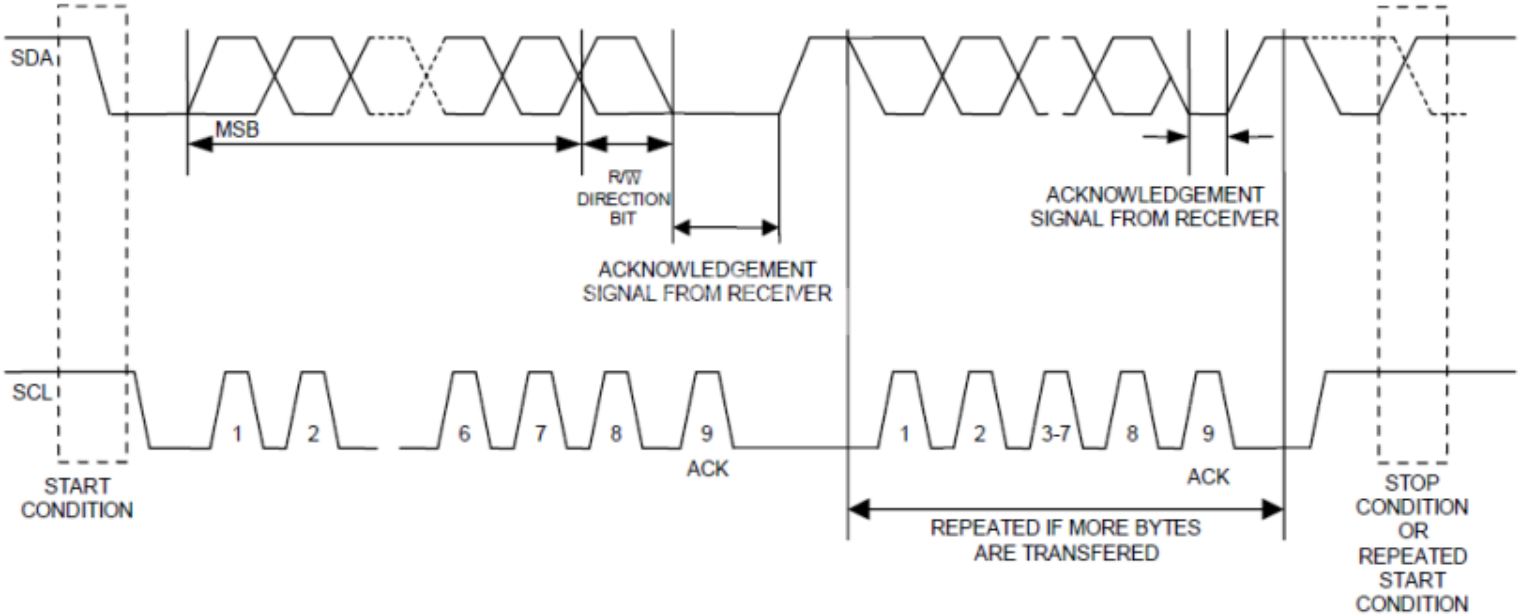
- bit START : SCL Hi oraz SDA $1 \rightarrow 0$;
- bit STOP : SCL Hi oraz SDA $0 \rightarrow 1$
- (przy transmisji danych) SDA jest ustalone przy SCL Hi
- powtórny START: bit START powtórzony po danych, bez STOP

I2C – timing



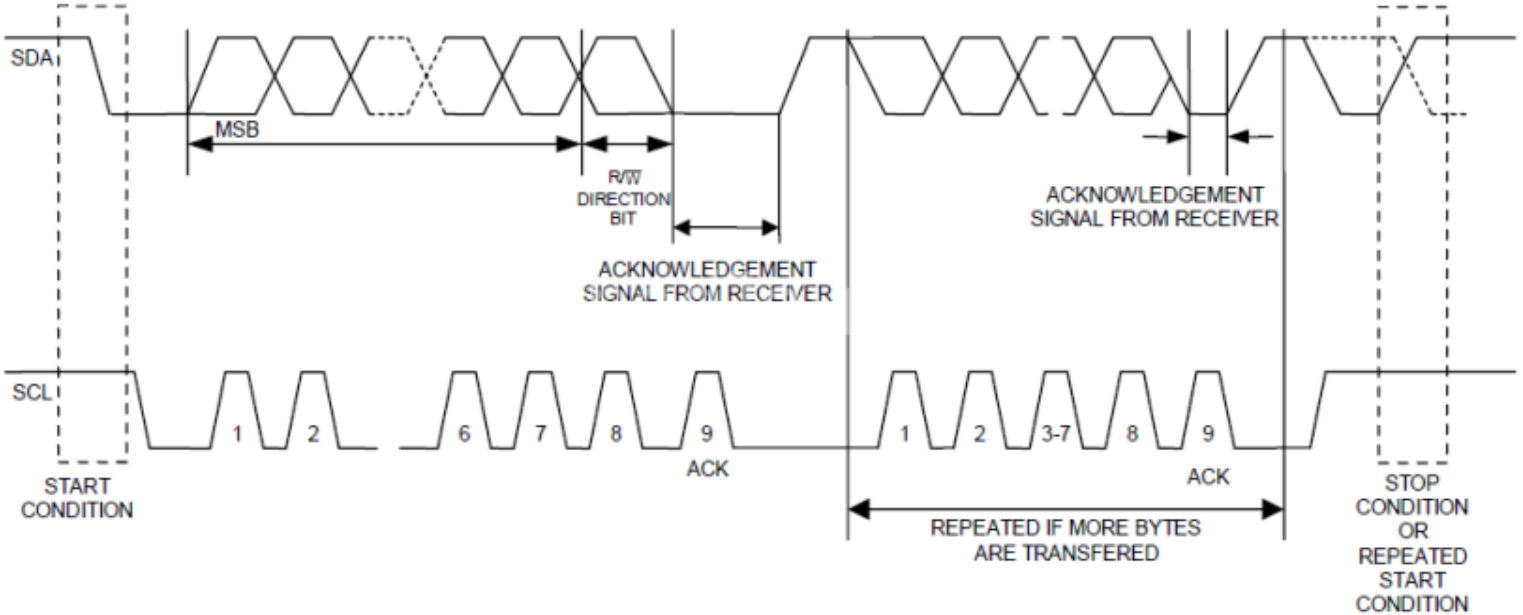
- bit START : SCL Hi oraz SDA $1 \rightarrow 0$;
- bit STOP : SCL Hi oraz SDA $0 \rightarrow 1$
- (przy transmisji danych) SDA jest ustalone przy SCL Hi
- powtórny START: bit START powtórzony po danych, bez STOP
- bit ACK : odbiorca wysyła bit 0

I2C – timing



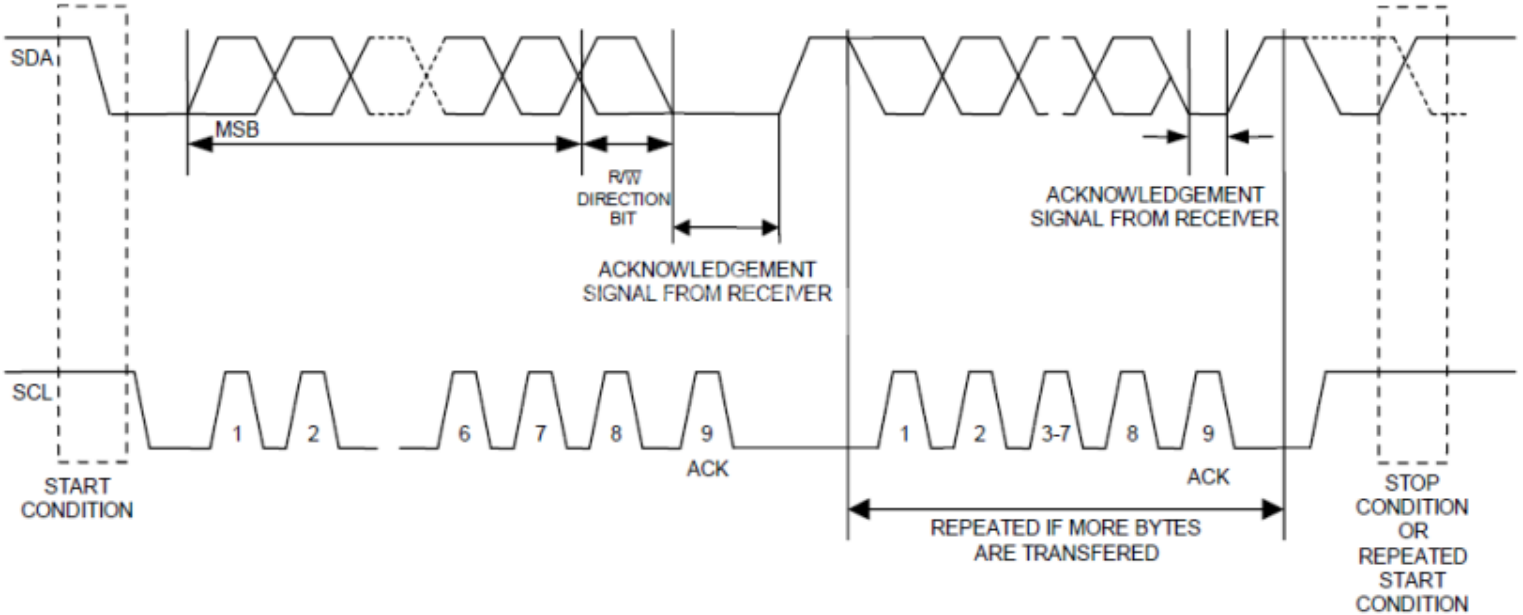
- bit START : SCL Hi oraz SDA $1 \rightarrow 0$;
- bit STOP : SCL Hi oraz SDA $0 \rightarrow 1$
- (przy transmisji danych) SDA jest ustalone przy SCL Hi
- powtórny START: bit START powtórzony po danych, bez STOP
- bit ACK : odbiorca wysyła bit 0
- bit NACK : odbiorca wysyła bit 1 (normalny stan na linii!)

I2C – timing



- bit START : SCL Hi oraz SDA $1 \rightarrow 0$;
- bit STOP : SCL Hi oraz SDA $0 \rightarrow 1$
- (przy transmisji danych) SDA jest ustalone przy SCL Hi
- powtórny START: bit START powtórzony po danych, bez STOP
- bit ACK : odbiorca wysyła bit 0
- bit NACK : odbiorca wysyła bit 1 (normalny stan na linii!)
  - gdy odbiorcą jest master: już nie chcę więcej czytać od slave

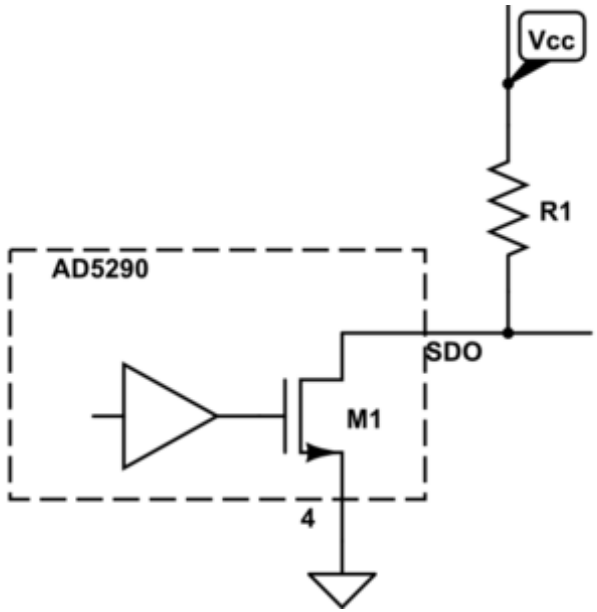
I2C – timing



- bit START : SCL Hi oraz SDA $1 \rightarrow 0$;
- bit STOP : SCL Hi oraz SDA $0 \rightarrow 1$
- (przy transmisji danych) SDA jest ustalone przy SCL Hi
- powtórny START: bit START powtórzony po danych, bez STOP
- bit ACK : odbiorca wysyła bit 0
- bit NACK : odbiorca wysyła bit 1 (normalny stan na linii!)
  - gdy odbiorcą jest master: już nie chcę więcej czytać od slave
  - gdy odbiorcą jest slave: nie mogę czytać, nie ma mnie, nie rozumiem;

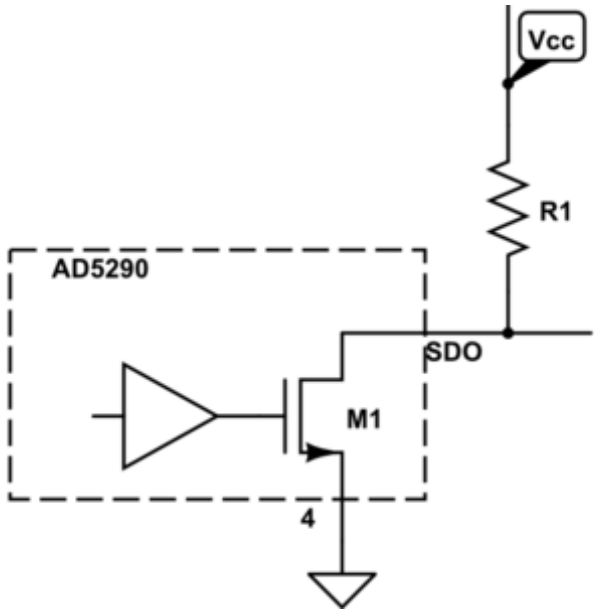
I2C - fizyczność

- na liniach SCL i SDA zastosowano rezystory pull-up



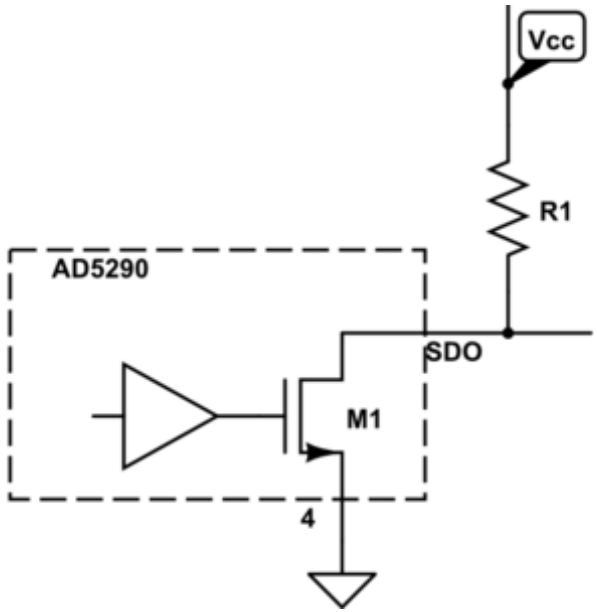
I2C - fizyczność

- na liniach SCL i SDA zastosowano rezystory pull-up
- V_{dd} to typowo 3.3 lub 5V



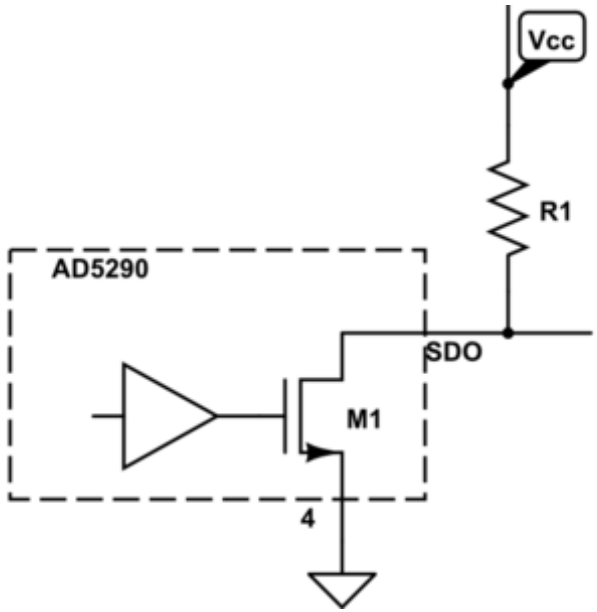
I2C - fizyczność

- na liniach SCL i SDA zastosowano rezystory pull-up
- V_{dd} to typowo 3.3 lub 5V
- linia, na której nikt nie nadaje (pływająca) ma V_{dd}



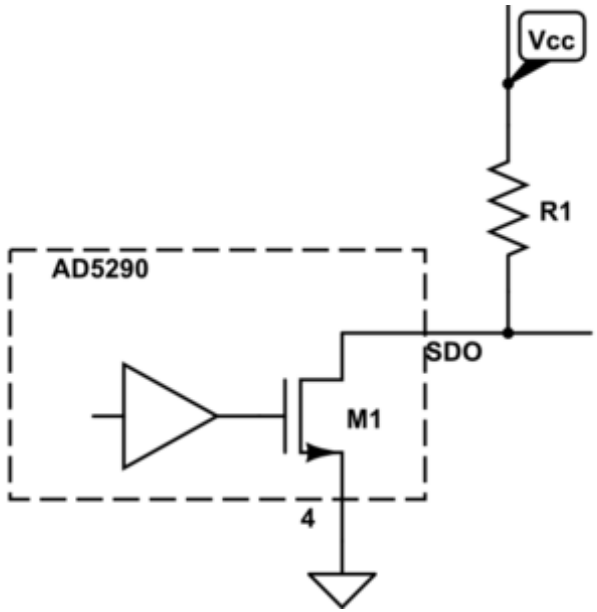
I2C - fizyczność

- na liniach SCL i SDA zastosowano rezystory pull-up
- V_{dd} to typowo 3.3 lub 5V
- linia, na której nikt nie nadaje (pływająca) ma V_{dd}
- dla SCL - *clock stretching*, dla SDA - *arbitraż*



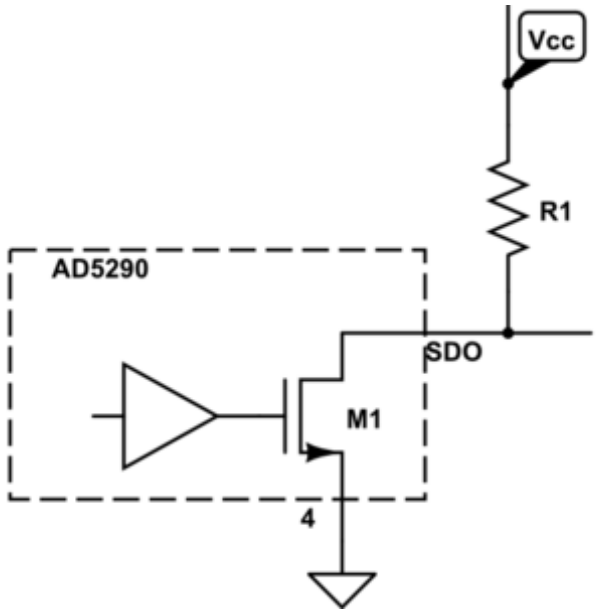
I2C - fizyczność

- na liniach SCL i SDA zastosowano rezystory pull-up
- V_{dd} to typowo 3.3 lub 5V
- linia, na której nikt nie nadaje (pływająca) ma V_{dd}
- dla SCL - *clock stretching*, dla SDA - *arbitraż*
  - slave może przedłużyć SCL na Lo



I2C - fizyczność

- na liniach SCL i SDA zastosowano rezystory pull-up
- V_{dd} to typowo 3.3 lub 5V
- linia, na której nikt nie nadaje (pływająca) ma V_{dd}
- dla SCL - *clock stretching*, dla SDA - *arbitraż*
  - slave może przedłużyć SCL na Lo
  - nadające urządzenia monitorują czy SDA jest takie, jak się spodziewają



PROFIBUS

- protokół szeregoweg połączenia wielu urządzeń

PROFIBUS

- protokół szeregoweg połączenia wielu urządzeń
- przemysłowy standard ISO 7498

PROFIBUS

- protokół szeregoweg połączenia wielu urządzeń
- przemysłowy standard ISO 7498
- określony dla warstw 1, 2 i 7 (PHY, MAC, APP)

PROFIBUS

- protokół szeregoweg połączenia wielu urządzeń
- przemysłowy standard ISO 7498
- określony dla warstw 1, 2 i 7 (PHY, MAC, APP)
- usługi przetwarzania danych i kontrolowania urządzeń

Urządzenia PROFIBUS

Master:

Urządzenia PROFIBUS

Master:

- kontroluje ruch magistrali

Urządzenia PROFIBUS

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing
- trzy klasy:

Urządzenia PROFIBUS

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing
- trzy klasy:
  1. kontrolery, sterowniki, PC-ty

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing
- trzy klasy:
  1. kontrolery, sterowniki, PC-ty
  2. ukł. narzędziowe (wdrażanie, kontrola, utrzymanie sieci)

Urządzenia PROFIBUS

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing
- trzy klasy:
  1. kontrolery, sterowniki, PC-ty
  2. ukł. narzędziowe (wdrażanie, kontrola, utrzymanie sieci)
  3. główny zegar, synchronizuje sieć

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing
- trzy klasy:
  1. kontrolery, sterowniki, PC-ty
  2. ukł. narzędziowe (wdrażanie, kontrola, utrzymanie sieci)
  3. główny zegar, synchronizuje sieć

Urządzenia PROFIBUS

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing
- trzy klasy:
  1. kontrolery, sterowniki, PC-ty
  2. ukł. narzędziowe (wdrażanie, kontrola, utrzymanie sieci)
  3. główny zegar, synchronizuje sieć

Slave:

Urządzenia PROFIBUS

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing
- trzy klasy:
  1. kontrolery, sterowniki, PC-ty
  2. ukł. narzędziowe (wdrażanie, kontrola, utrzymanie sieci)
  3. główny zegar, synchronizuje sieć

Slave:

- urządzenia I/O, czujniki, aktywatory

Urządzenia PROFIBUS

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing
- trzy klasy:
  1. kontrolery, sterowniki, PC-ty
  2. ukł. narzędziowe (wdrażanie, kontrola, utrzymanie sieci)
  3. główny zegar, synchronizuje sieć

Slave:

- urządzenia I/O, czujniki, aktywatory
- odpowiadają tylko na żądanie mastera

Urządzenia PROFIBUS

Master:

- kontroluje ruch magistrali
- może być kilka $\rightarrow$ token passing
- trzy klasy:
  1. kontrolery, sterowniki, PC-ty
  2. ukł. narzędziowe (wdrażanie, kontrola, utrzymanie sieci)
  3. główny zegar, synchronizuje sieć

Slave:

- urządzenia I/O, czujniki, aktywatory
- odpowiadają tylko na żądanie mastera
- prostsze w implementacji

Adresacja

- Każde urządzenie na magistrali posiada *adres* – 1-bajtową liczbę

Adresacja

- Każde urządzenie na magistrali posiada *adres* – 1-bajtową liczbę
- Urządzenia **master** mają niskie adresy (1 . . . n)

Adresacja

- Każde urządzenie na magistrali posiada *adres* – 1-bajtową liczbę
- Urządzenia **master** mają niskie adresy (1 . . . n)
- Urządzenia **slave** dysponują pozostałą pulą

Adresacja

- Każde urządzenie na magistrali posiada *adres* – 1-bajtową liczbę
- Urządzenia **master** mają niskie adresy (1 . . . n)
- Urządzenia **slave** dysponują pozostałą pulą
- Adres 126 (0x7E) – adres dla urządzeń o zmiennym adresie

Adresacja

- Każde urządzenie na magistrali posiada *adres* – 1-bajtową liczbę
- Urządzenia **master** mają niskie adresy (1 . . . n)
- Urządzenia **slave** dysponują pozostałą pulą
- Adres 126 (0x7E) – adres dla urządzeń o zmiennym adresie
- Adres 127 (0x7F) – adres rozgłoszeniowy (broadcast)

Adresacja

- Każde urządzenie na magistrali posiada *adres* – 1-bajtową liczbę
- Urządzenia **master** mają niskie adresy (1 . . . n)
- Urządzenia **slave** dysponują pozostałą pulą
- Adres 126 (0x7E) – adres dla urządzeń o zmiennym adresie
- Adres 127 (0x7F) – adres rozgłoszeniowy (broadcast)
- Urządzenia typu repeater, interfejsy FO są przezroczyste

Medium transportowe

- RS-485 (dwużyłowe, kodowanie różnicowe, +1 – +4V)

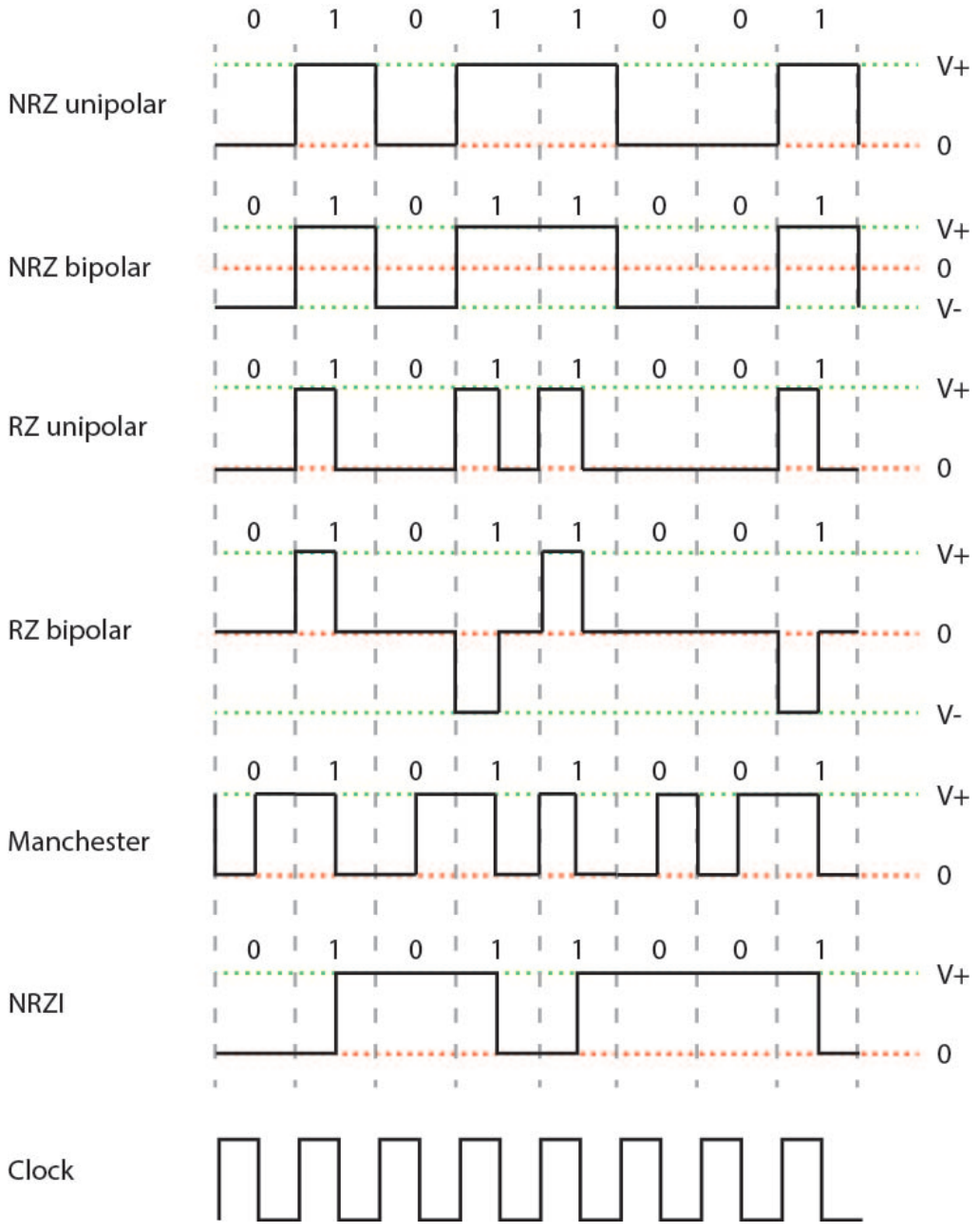
Medium transportowe

- RS-485 (dwużyłowe, kodowanie różnicowe, +1 – +4V)
- światłowodowe (szklane, syntetyczne)

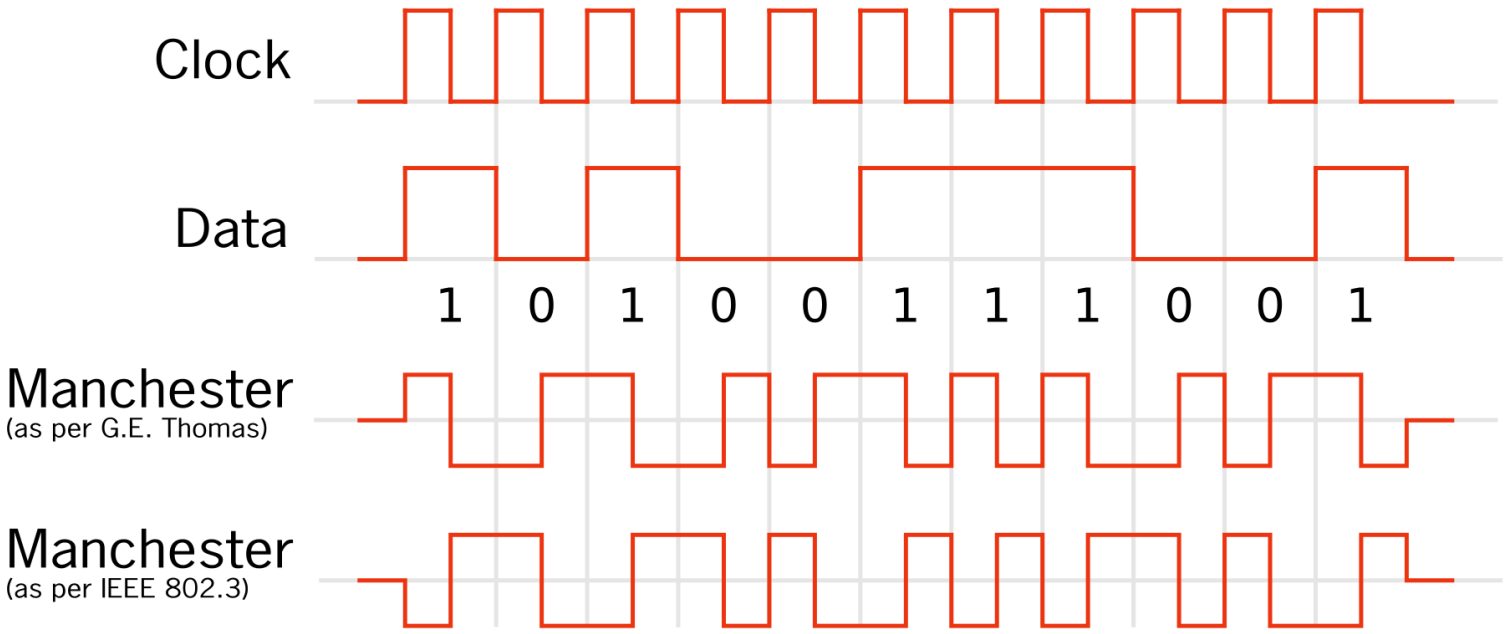
Medium transportowe

- RS-485 (dwużyłowe, kodowanie różnicowe, +1 – +4V)
- światłowodowe (szklane, syntetyczne)
- MBP (Manchester Bus Powered) (31,25 kbps) ...

Kodowanie, kodowanie ...



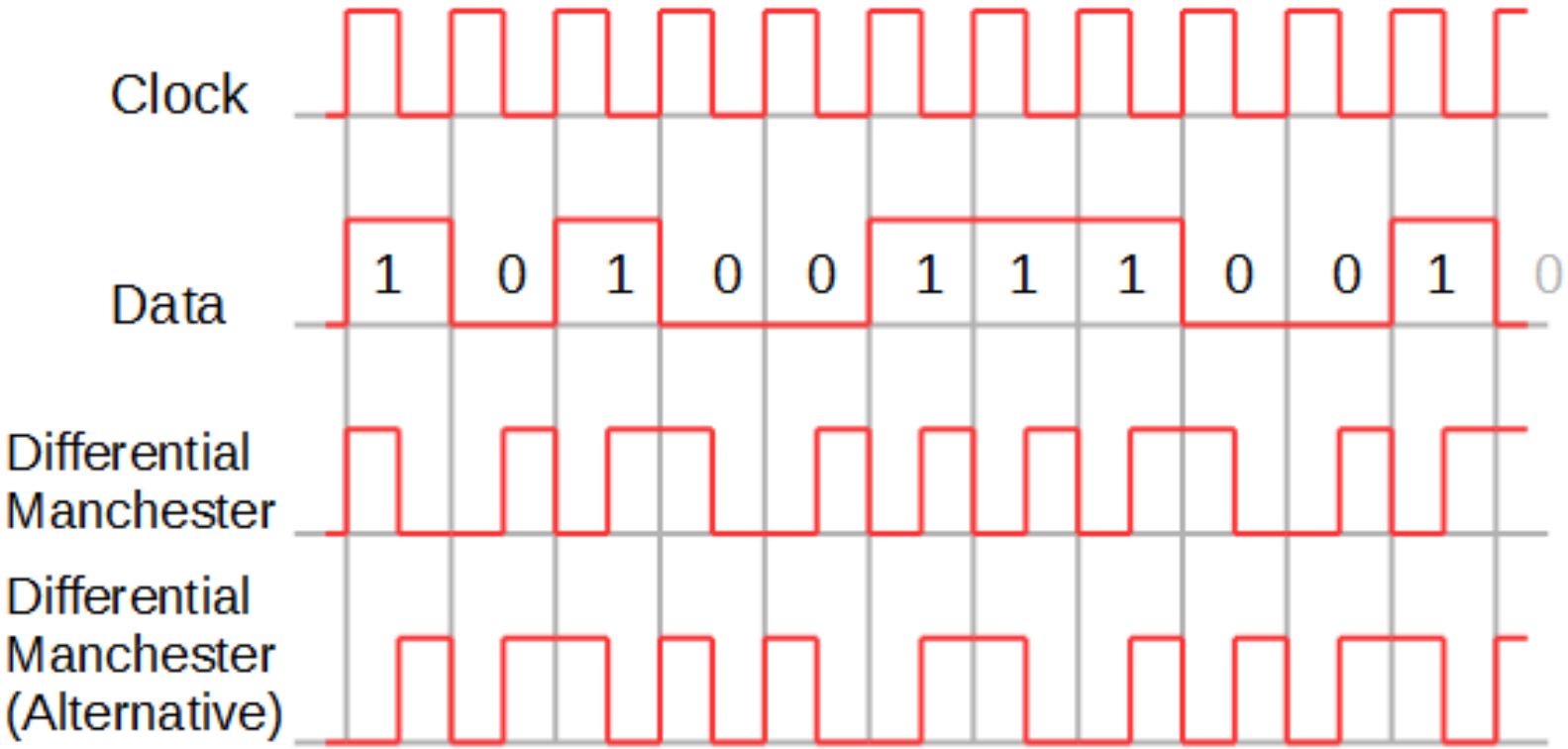
Kodowanie, kodowanie ...



źródło: WikiMedia

Kodowanie: przejście "w połowie bitu" $\mathbf{1}: 1 \searrow 0$, $\mathbf{0}: 0 \nearrow 1$ (lub odwrotnie...)

Kodowanie, kodowanie ...



źródło: WikiMedia

Kodowanie: zawsze zmiana "w połowie bitu" (opadające zbocze CLK), a "na początku bitu" zmiana 1 $\searrow$ 0 lub 0 $\nearrow$ 1 koduje **1**, a brak zmiany koduje **0** (lub odwrotnie...)

FDL – warstwa danych

Dostępne są następujące schematy przesyłania danych:

FDL – warstwa danych

Dostępne są następujące schematy przesyłania danych:

- Send Data, No ACK (zarządzanie)

FDL – warstwa danych

Dostępne są następujące schematy przesyłania danych:

- Send Data, No ACK (zarządzanie)
- Send Data, ACK (między masterami)

FDL – warstwa danych

Dostępne są następujące schematy przesyłania danych:

- Send Data, No ACK (zarządzanie)
- Send Data, ACK (między masterami)
- Send Data, Request Data (master $\leftrightarrow$ slave)

FDL – warstwa danych

Dostępne są następujące schematy przesyłania danych:

- Send Data, No ACK (zarządzanie)
- Send Data, ACK (między masterami)
- Send Data, Request Data (master $\leftrightarrow$ slave)
- Cyclic Send Data, Request Data

FDL – warstwa danych

Dostępne są następujące schematy przesyłania danych:

- Send Data, No ACK (zarządzanie)
- Send Data, ACK (między masterami)
- Send Data, Request Data (master $\leftrightarrow$ slave)
- Cyclic Send Data, Request Data
- Send and Request Data, Broadcast Response

FDL – warstwa danych

Dostępne są następujące schematy przesyłania danych:

- Send Data, No ACK (zarządzanie)
- Send Data, ACK (między masterami)
- Send Data, Request Data (master $\leftrightarrow$ slave)
- Cyclic Send Data, Request Data
- Send and Request Data, Broadcast Response
  - paradygmat publish-subscribe

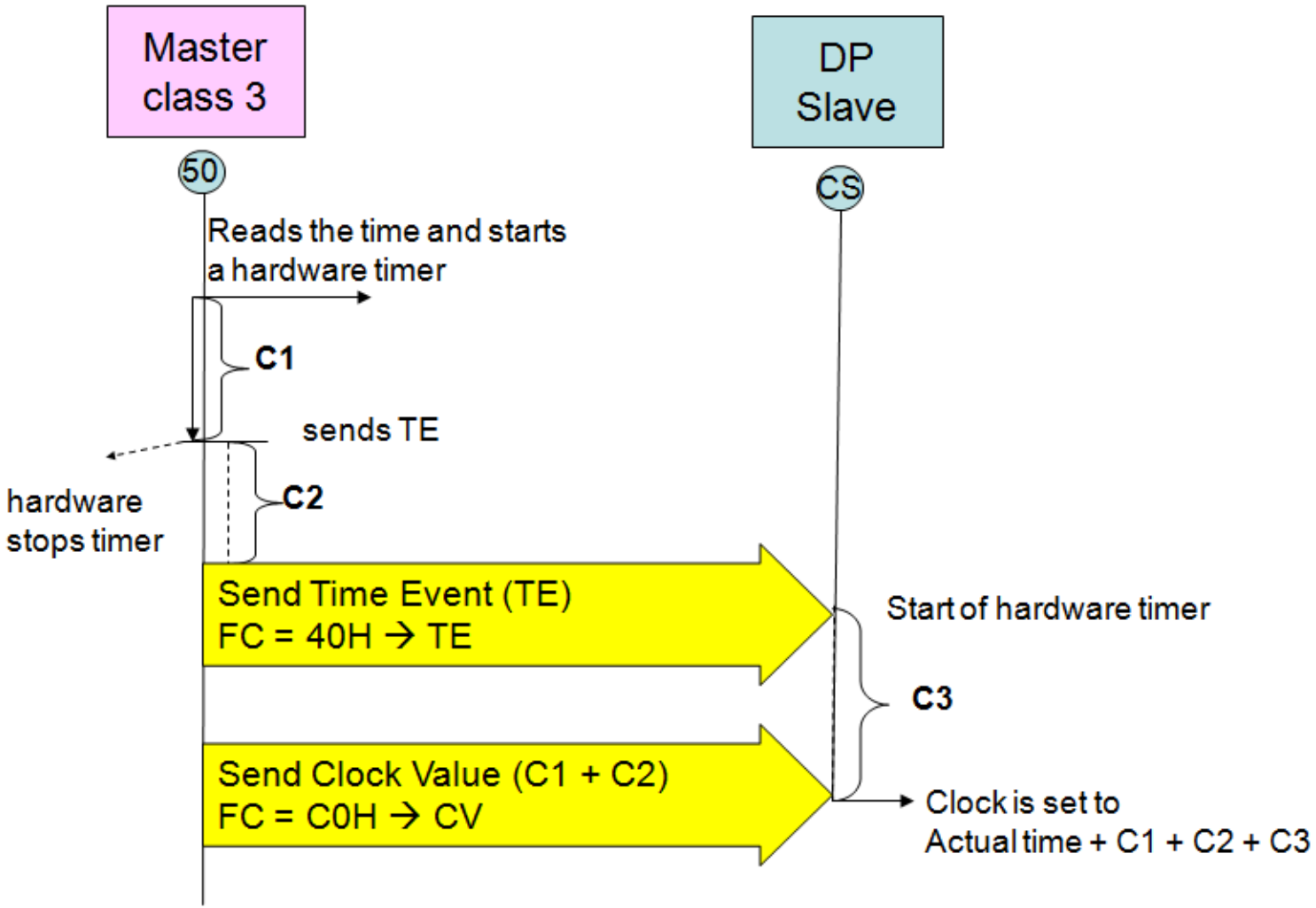
FDL – warstwa danych

Dostępne są następujące schematy przesyłania danych:

- Send Data, No ACK (zarządzanie)
- Send Data, ACK (między masterami)
- Send Data, Request Data (master $\leftrightarrow$ slave)
- Cyclic Send Data, Request Data
- Send and Request Data, Broadcast Response
  - paradygmat publish-subscribe
- Clock Sync

Synchronizacja zegara

Synchronizacja zegara



www.felser.ch/profibus-manual/

MAC – warstwa dostępowa

- Tylko **master** posiadający żeton (token) inicjuje komunikację

MAC – warstwa dostępowa

- Tylko **master** posiadający żeton (token) inicjuje komunikację
- Żeton jest przekazywany w kierunku wyższych adresów po określonym czasie

MAC – warstwa dostępowa

- Tylko **master** posiadający żeton (token) inicjuje komunikację
- Żeton jest przekazywany w kierunku wyższych adresów po określonym czasie
  - Każdy master cyklicznie odpytuje urządzenia do następnego mastera

MAC – warstwa dostępowa

- Tylko **master** posiadający żeton (token) inicjuje komunikację
- Żeton jest przekazywany w kierunku wyższych adresów po określonym czasie
  - Każdy master cyklicznie odpytuje urządzenia do następnego mastera
  - Nowy master może wtedy się zgłosić, stary potwierdza swój status online

MAC – warstwa dostępowa

- Tylko **master** posiadający żeton (token) inicjuje komunikację
- Żeton jest przekazywany w kierunku wyższych adresów po określonym czasie
  - Każdy master cyklicznie odpytuje urządzenia do następnego mastera
  - Nowy master może wtedy się zgłosić, stary potwierdza swój status online
  - Jeśli minie ustalony czas i żeton się nie pojawi w sieci, każdy master ustawia timer: $6 \cdot T_{sl} + 2n \cdot T_{sl}$ i pierwszy, który "odpali" rozpoczyna proces od nowa
    $\rightarrow$ programatory mają adres $n = 0$

MAC – warstwa dostępowa

- Tylko **master** posiadający żeton (token) inicjuje komunikację
- Żeton jest przekazywany w kierunku wyższych adresów po określonym czasie
  - Każdy master cyklicznie odpytuje urządzenia do następnego mastera
  - Nowy master może wtedy się zgłosić, stary potwierdza swój status online
  - Jeśli minie ustalony czas i żeton się nie pojawi w sieci, każdy master ustawia timer: $6 \cdot T_{sl} + 2n \cdot T_{sl}$ i pierwszy, który "odpali" rozpoczyna proces od nowa
    $\rightarrow$ programatory mają adres $n = 0$
- Każda stacja liczy czas do powrotu żetonu (jeśli jest mniejszy, niż ustalony próg - może nadawać)

Ramki danych

no data	SD1	dst	src	FC	crc	ED				
data1	SD2	len	len	SD2	dst	src	FC	data	crc	ED
data2	SD3	dst	src	FC	data	crc	ED			
żeton	SD4	dst	src							
ACK	SC									

- SDx (start delim), SC (short conf.), ED (end delim) – odległość Hamminga = 4
- FC - *function code*

Ramki danych

no data	SD1	dst	src	FC	crc	ED				
data1	SD2	len	len	SD2	dst	src	FC	data	crc	ED
data2	SD3	dst	src	FC	data	crc	ED			
żeton	SD4	dst	src							
ACK	SC									

- SDx (start delim), SC (short conf.), ED (end delim) – odległość Hamminga = 4
- FC - *function code*
  - zawiera opis schematu przesyłania danych lub (zwrotnie) kod błędu;

Ramki danych

no data	SD1	dst	src	FC	crc	ED				
data1	SD2	len	len	SD2	dst	src	FC	data	crc	ED
data2	SD3	dst	src	FC	data	crc	ED			
żeton	SD4	dst	src							
ACK	SC									

- SDx (start delim), SC (short conf.), ED (end delim) – odległość Hamminga = 4
- FC - *function code*
  - zawiera opis schematu przesyłania danych lub (zwrotnie) kod błędu;
  - zawiera bity FCV i FCB (frame count bit) – odwracany po każdej udanej transmisji)

Do zapamiętania

- I2C: linie, sygnały, arbitraż, clock-stretching

Do zapamiętania

- I2C: linie, sygnały, arbitraż, clock-stretching
- PROFIBUS: master/slave vs. token-ring, mechanizmy dołączania, komunikacji...

Do zapamiętania

- I2C: linie, sygnały, arbitraż, clock-stretching
- PROFIBUS: master/slave vs. token-ring, mechanizmy dołączania, komunikacji...
- kodowania Manchester i inne...