

Le Interfacce Seriali

1

G. MARSELLA
UNIVERSITÀ DEL SALENTO



Introduzione

2

- Generalità
- RS232
- USB
- SPI
- I²C

Generalità

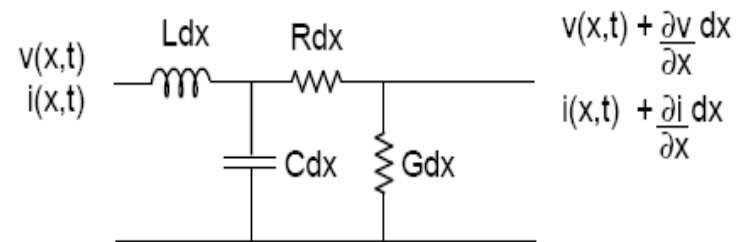
3

- La trasmissione delle informazioni tra i componenti di un medesimo elaboratore o fra più elaboratori avviene tramite un mezzo fisico (cavi di rame, fibra ottica, radio, etc.) mediante il quale si ottiene una connessione (link).
- Il modo in cui i segnali ed i dati sono organizzati e si propagano attraverso il link definisce un **protocollo** di comunicazione.
- I protocolli possono sommarsi e dar vita a protocolli più complessi (pile di protocollo) e permettere all'utente finale di trasmettere dati nello stesso modo su differenti link fisici (un esempio di pila di protocolli è TCP/IP su Ethernet, o su Fibre Channel)

Le linee di trasmissione



- Per linea di trasmissione si intende in genere una coppia di conduttori attraverso cui si propaga un segnale. I due conduttori possono essere paralleli (come in una piattina), coassiali, o composti da un conduttore ed un piano di massa (come nei circuiti stampati).
- Un tratto infinitesimo di linea si può schematizzare come in figura, in cui i parametri L , R , C e G sono intesi per unità di lunghezza



Dette $Z=R+i\omega L$ e $Y=G+i\omega C$ ed applicando le leggi di Kirkhoff otteniamo una coppia di equazioni delle onde:

$$\frac{\partial^2 V}{\partial x^2} = \gamma_0^2 V; \quad \frac{\partial^2 I}{\partial x^2} = \gamma_0^2 I; \quad \gamma_0 = \sqrt{ZY}$$

Linee di trasmissione II



- Se R e G sono trascurabili (linea non dissipativa):

$$\frac{\partial^2 v}{\partial x^2} = LC \frac{\partial^2 v}{\partial t^2}$$

Quindi un'equazione delle onde con velocità $u=1/\sqrt{LC}$ con soluzioni del tipo:

$$v(x, t) = f_1(x-ut) + f_2(x+ut)$$

$$i(x, t) = 1/R_0[f_1(x-ut) - f_2(x+ut)]$$

Ove $R_0 = \sqrt{L/C}$ è l'impedenza caratteristica della linea.

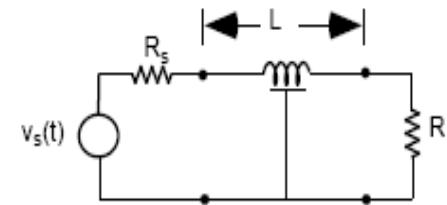
Se chiudiamo la linea su una resistenza R , nel punto alla fine della linea, la corrente sarà $v/R \Rightarrow i(L, t) = 1/R_0[v_s - \rho v_s] = 1/R[v_s + \rho v_s]$

Da cui si ricava il coefficiente di riflessione:

$$\rho = (R - R_0)/(R + R_0).$$

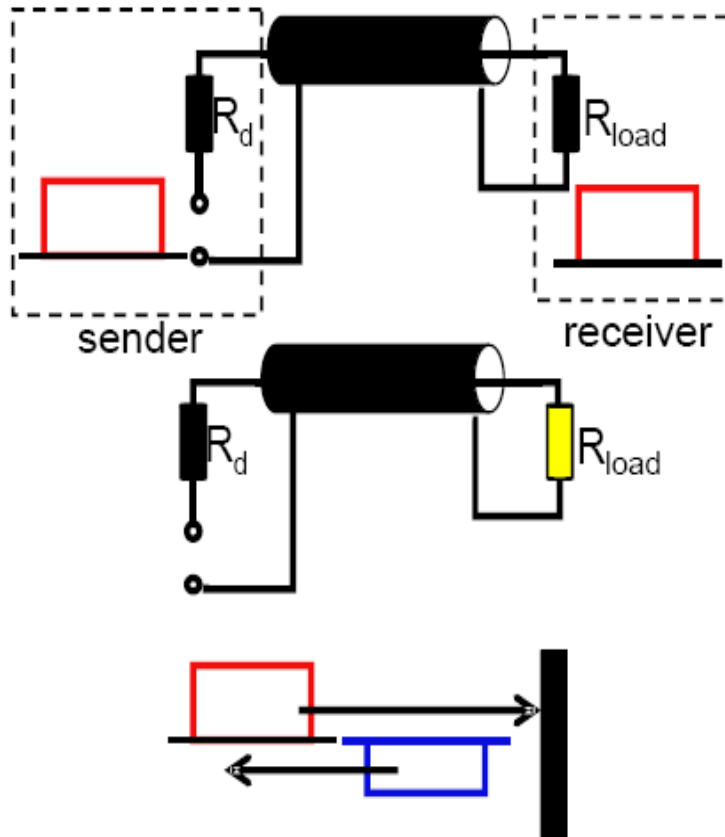
Se $R = R_0$ non c'è riflessione: **Linea adattata**

R è la **terminazione**



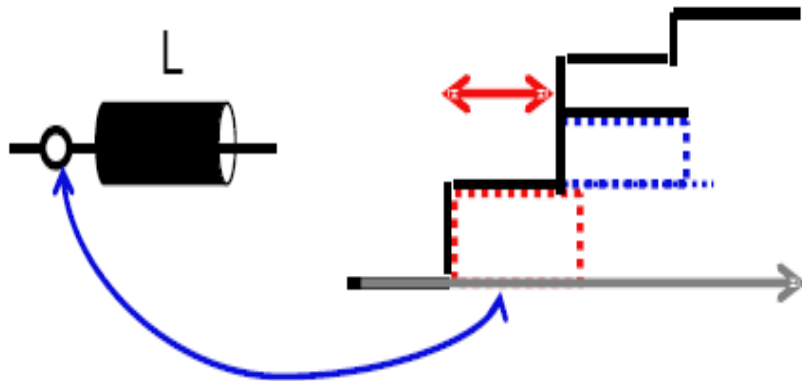
Linea di lunghezza L : carico R

Adattamento di impedenza

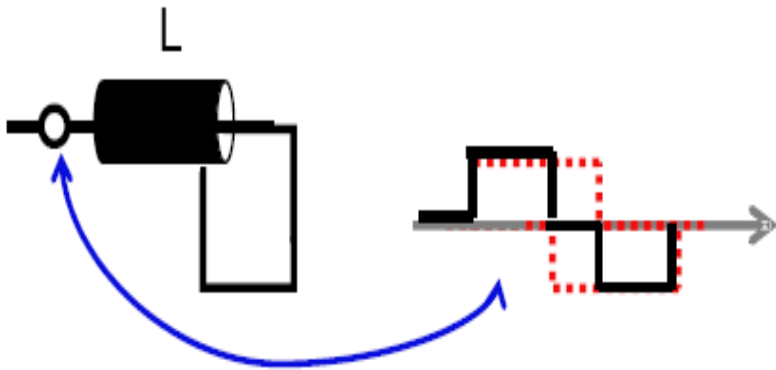


- Se l'impedenza è adattata il cavo appare infinitamente lungo: nessun segnale riflesso
- Se l'impedenza è disadattata un segnale riflesso al termine della linea si sovrappone al segnale inviato.

Ancora adattamento



- Linea aperta: $R_{load} = \infty$
l'ingresso e la riflessione hanno uguale polarità.
- Si sovrappongono per $t = 2T = 2l/c$

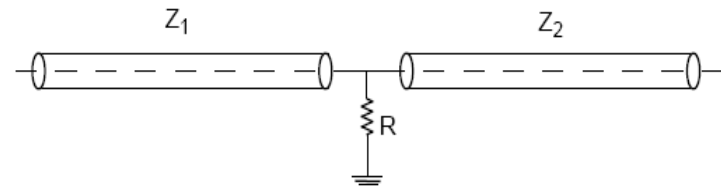


- Linea cortocircuitata: $R_{load} = 0$
l'ingresso e la riflessione hanno opposta polarità.
- Si sovrappongono per $t = 2T = 2l/c$
- Segnale bipolare

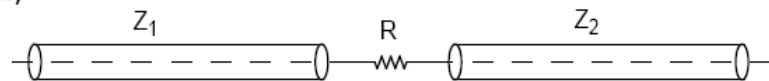
Giunzioni e divisioni di linee



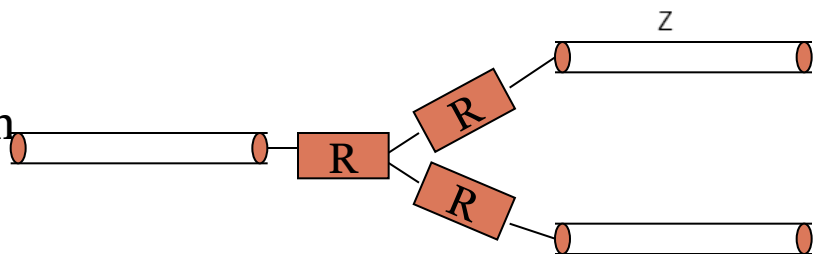
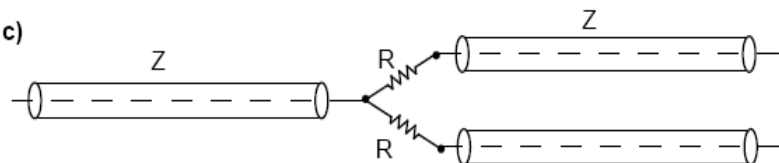
a)



b)



c)



a) $Z_1 < Z_2 \quad R = Z_1 Z_2 / (Z_1 + Z_2)$

b) $Z_1 > Z_2 \quad R = Z_2 - Z_1$

c) Se la divisione è in n rami bisogna imporre che $(R+Z)/n = Z \Rightarrow R = (n - 1)Z$.

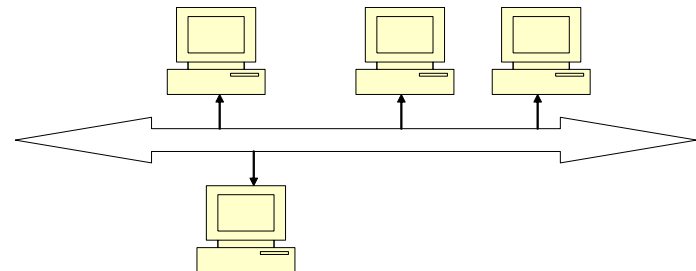
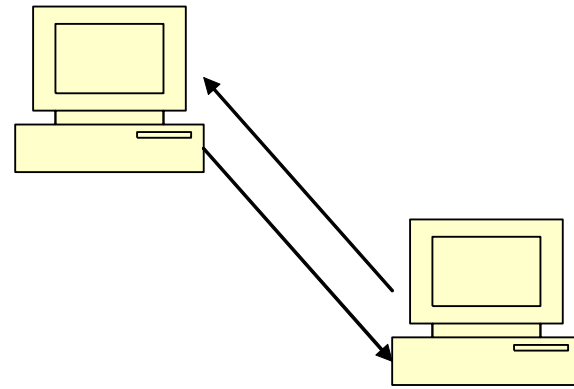
Per $n=2$; $R=Z$

Però in questo caso le due linee aggiunte non vedranno un'impedenza Z . Va usato uno splitter simmetrico con 3 resistenze uguali: $R=Z/3$ (per $Z=50\Omega$, $R=17\Omega$)

In generale $R = Z(N-1)/(N+1)$

Trasmissione punto-punto e bus

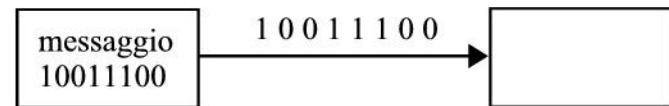
- Nella trasmissione punto-punto due dispositivi sono connessi fra loro e con nessun altro attraverso il link (esempio: un monitor con la scheda grafica attraverso il cavo VGA)
- I bus connettono più dispositivi tra loro mediante il medesimo link (esempio: più dischi con il computer e fra loro tramite il bus SCSI o IDE).



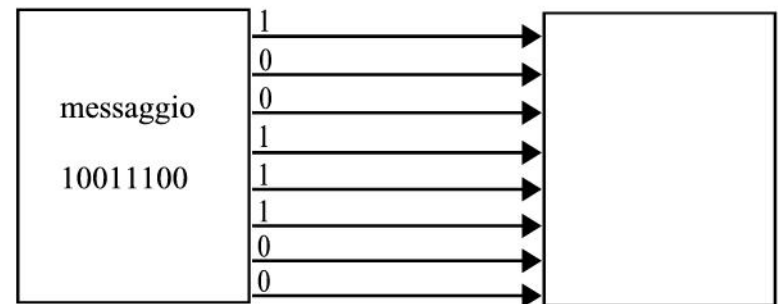
Trasmissione seriale e parallela



- La trasmissione dei dati si dice seriale, se questi sono trasmessi attraverso il link un bit alla volta, in sequenza. È questo il caso, per esempio, quando è presente un solo cavo per la trasmissione.
- La trasmissione si dice parallela quando tutti i bit di una parola sono trasmessi contemporaneamente, in parallelo. Per ottenere ciò, è necessario avere a disposizione tanti cavi (o canali) quanti sono i bit della parola.



(a) TRASMISSIONE SERIALE



(b) TRASMISSIONE PARALLELA

Figura 2 - Trasmissione seriale e parallela

Full duplex/Half duplex

11

- Una linea di comunicazione si dice *full duplex* se è in grado di ricevere e trasmettere contemporaneamente – ci sono due canali: uno in ingresso l'altro in uscita.
- Si dice *half duplex* se non può trasmettere e ricevere allo stesso tempo. C'è un solo canale su cui i dati possono viaggiare in un senso o nell'altro.

Rivelazione degli errori – bit di parità

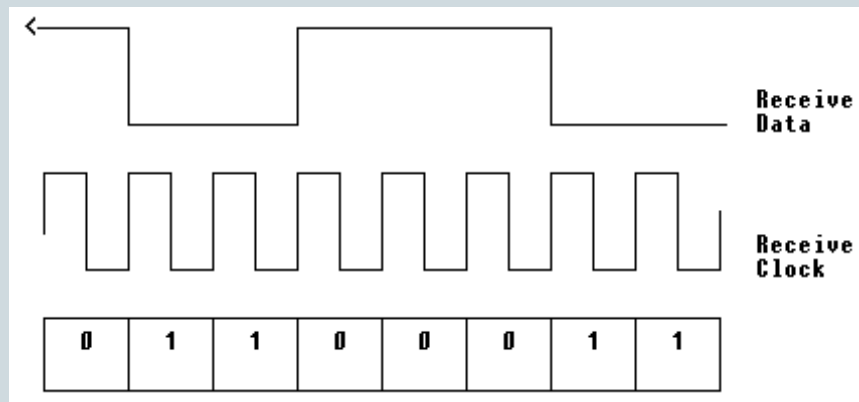
12

- Può capitare che i bit trasmessi risultino alterati al momento della ricezione, a causa di interferenze o errori nella trasmissione.
- C'è bisogno di un metodo per permettere la rivelazione di errori.
- Si possono trasmettere assieme ai dati uno o più bit di controllo.
- Un metodo comunemente usato è il bit di parità: esso vale 1 se il numero di bit con valore 1 compreso il bit di parità è pari (parity even), o se dispari (parity odd).
- Se in ricezione il bit di parità calcolato è diverso da quello ricevuto, un bit deve essere cambiato nel tragitto o erroneamente trasmesso .
- Non funziona se cambia più di un bit ma la probabilità di due errori nella trasmissione di un singolo byte è bassa.

Sincronizzazione

13

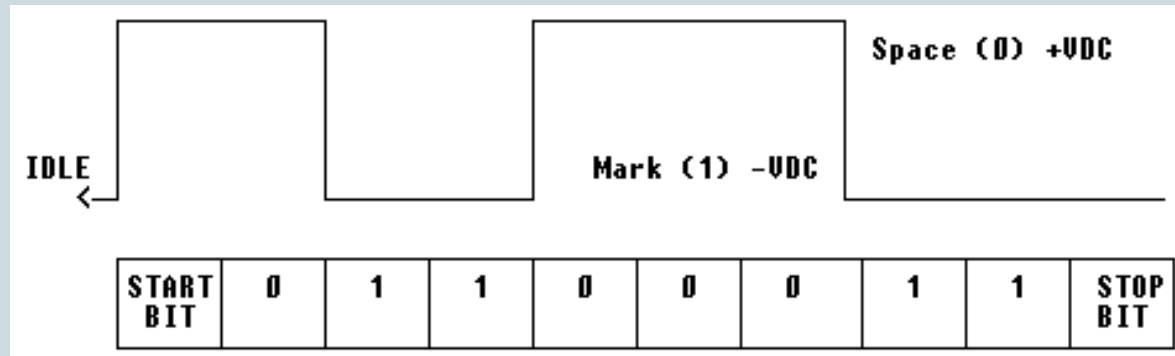
- Sia nella trasmissione seriale sia in quella parallela, spesso viene trasmesso un segnale di sincronizzazione su un canale dedicato (clock).
- Il clock consente al ricevitore di decodificare i segnali nello stesso tempo in cui vengono inviati. Il ricevitore ed il trasmettitore sono, così, sincronizzati nel tempo di clock.
- La comunicazione, in questo, caso si dice **sincrona**.
- La frequenza del clock definisce anche la velocità della comunicazione che si misura in bit x s=**bps** o **baud** (in realtà baud indica la frequenza con cui la linea cambia stato che non è necessariamente pari a bps).
- Il vantaggio di questo tipo di trasmissione è che qualsiasi drift nella frequenza di trasmissione è riconosciuto dal ricevitore. Sono, inoltre, possibili frequenze molto alte.



Comunicazione asincrona

14

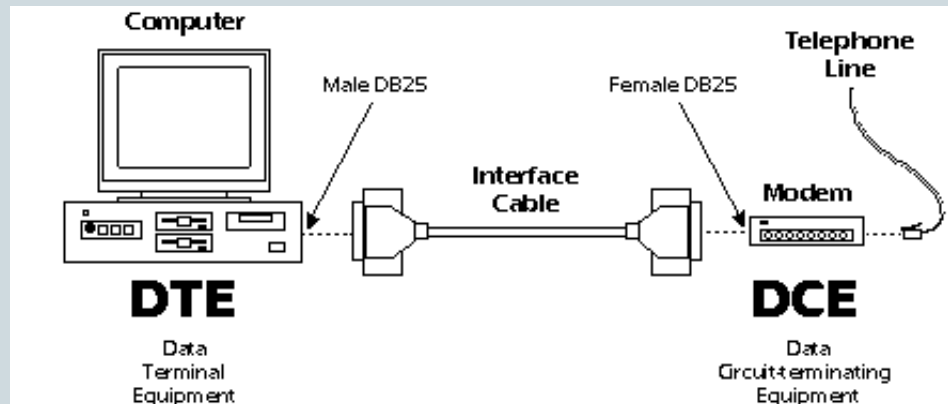
- Nella comunicazione asincrona, non viene trasmesso alcun segnale di clock.
- Ai dati vengono aggiunti in testa ed in coda una serie di bit, detti di start e stop, che istruiscono il ricevitore dell'inizio e della fine di una parola.
- Il vantaggio è che trasmettitore e ricevitore hanno clock indipendenti e le parole possono essere separate da un qualsiasi intervallo di tempo.



Linee seriali RS232

15

- RS-232 è uno standard per le comunicazioni seriali definito dalla Electronic Industries Association (EIA). RS sta per Recommended Standard.
- Esiste in tre diverse versioni A, B o C che definiscono i differenti voltaggi per i livelli *on* ed *off*.
- La versione più utilizzata è RS-232C che definisce un bit *on* (mark) come una tensione tra -3V e -12V e *off* (space) tra +3V e +12V.
- Lo standard definisce anche che la massima distanza lungo la quale questi segnali possono essere trasmessi è 8m (25 ft).
- Le porte o i dispositivi seriali sono etichettati come Data Communication Equipment (DCE) o Data Terminal Equipment (DTE). I segnali di trasmissione e ricezione fra queste due tipologie sono invertiti di posto.



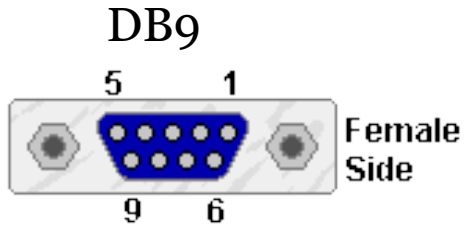
RS232 – schema del connettore



- Oltre ai cavi per la trasmissione e la ricezione dei dati, RS232 necessita di altri cavi per il controllo del flusso di dati.
- Il connettore utilizzato è un connettore a “D” a 25 poli su cui coesistono un canale primario ed uno secondario.
- Il segnale FG non è quasi mai utilizzato, per cui spesso è sufficiente un connettore a 9 poli.
- Eliminando il segnale DSR si può usare anche un connettore a 8 poli come quelli in uso per il networking (RJ45)

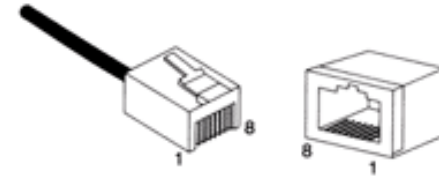
DB-25 Pin#	DB-9 Pin#	Nome comune	Direzione DTE-DCE	Nome formale
1		FG	-	Frame Ground
2	3	TD	→	Transmitted Data, TxD
3	2	RD	←	Received Data, RxD
4	7	RTS	→	Request to Send
5	8	CTS	←	Clear To Send
6	6	DSR	←	Data Set Ready
7	5	SG	-	Signal Ground, GND
8	1	DCD	←	Data Carrier Detect
9		--	-	+P
10		--	-	-P
11		--	-	Non assegnato
12		SDCD	←	Secondary Data Carrier Detect
13		SCTS	←	Secondary Clear To Send
14		STD	→	Secondary Transmitted Data
15		TC	←	Transmission Signal Element Timing
16		SRD	←	Secondary Received Data
17		RC	→	Receiver Signal Element Timing
18		--	-	Non assegnato
19		SRTS	→	Secondary Request To Send
20	4	DTR	→	Data Terminal Ready
21		SQ	←	Signal Quality detector
22	9	RI	←	Ring Indicator
23		--	→	Data Signal Rate Selector
24		--	←	Transmitter Signal Element Timing
25		--	-	Non assegnato

RS232 Connettori ridotti



1	DCD
2	RXD
3	TXD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

RJ45



1	RI
2	DCD
3	DTR
4	GND
5	RXD
6	TXD
7	CTS
8	RTS

RS232 Segnali di controllo I

18

- **GND - Logic Ground:** Riferimento di tensione per tutti gli altri segnali.
- **TXD - Transmitted Data:** Linea di trasmissione dei bit di informazione dal DTE (periferica) a DCE (computer). Il DTE mantiene tale linea al valore logico 1 quando *non* ci sono dati da trasmettere; la trasmissione del dato su questa linea è possibile solo se i segnali *Request To Send*, *Clear To Send*, *Data Set Ready* e *Data Terminal Ready*, quando presenti, assumono valore logico 0.
- **RXD - Received Data:** Linea di trasmissione dei bit di informazione dal DCE (computer) a DTE (periferica). Il dato (bit) primario viene inviato su questa linea dal DCE al DTE. Questo segnale viene mantenuto ad un valore logico 1 quando DCE non trasmette dati e viene portato a 0 per un breve intervallo di tempo dopo una transizione della linea *Request To Send* da 1 a 0, per consentire il completamento della trasmissione.
- **DCD - Data Carrier Detect:** Il segnale DCD indica che il computer o il dispositivo sono connessi o on line. DCD non è sempre usato o disponibile.
- **DSR – Data set Ready:** Su questa linea il DCE dice al DTE che il canale di comunicazione è disponibile, ma non indica che effettivamente sia stato stabilito un link con un dispositivo remoto.

RS232 Segnali di controllo II

19

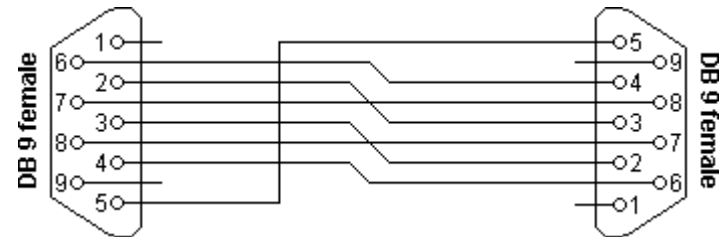
- **DTR - Data Terminal Ready:** Se questo segnale è a livello logico 1, DCE viene informato che DTE è pronto per la ricezione. Il segnale DTR deve essere attivo prima che DCE attivi il segnale Data Set Ready, indicando così di essere connesso al canale di comunicazione. Se il segnale DTR assume il valore logico 0, DCE interrompe la trasmissione in corso.
- **CTS - Clear To Send:** Segnale di risposta a DTE. Quando attivo, indica a DTE che può dare inizio alla trasmissione (linea TXD)
- **RTS - Request To Send:** dal DTE al DCE, disponibilità a trasmettere; quando attivo questo segnale informa il DCE che il DTE è pronto a spedire un byte.
- **Transmitter Signal Element Timing:** Linea usata da DTE per inviare a DCE un segnale di clock. La transizione da 1 a 0 indica il punto centrale del tratto di segnale corrispondente ad un bit sul *Transmitted Data*.
- **Receiver Signal Element Timing:** Segnale di clock inviato da DCE a DTE in modo che DTE sia in grado di sincronizzare il proprio circuito di ricezione che pilota la linea Received Data. La frequenza del segnale di clock dipende dal bit-rate della trasmissione sulla linea Received Data. La transizione da 1 a 0 indica il punto centrale del tratto di segnale corrispondente ad un bit sulla Received Data.

Collegare fra loro due DTE

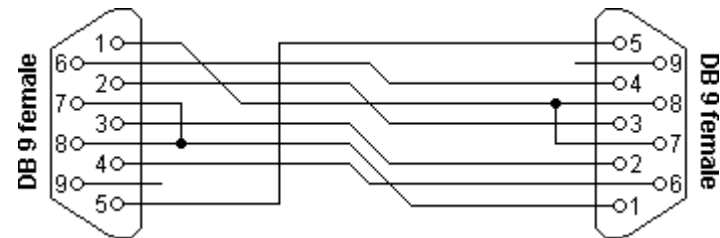


- Serve un collegamento con trasmissione e ricezione incrociate (null modem).
- Bisogna tener conto anche dei segnali di controllo (handshaking) o collegare mutuamente DTR con DSR e RTS con CTS o tra loro gli RTS e CTS di ciascuna porta ed entrambi al DCD dell'altra
- Alternativamente si possono mettere "a loop" i segnali di handshaking su entrambe le porte, ma a questo punto il controllo di flusso non potrà più essere hardware.

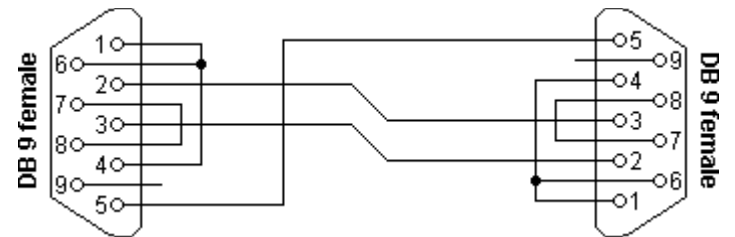
Full Handshaking



partial Handshaking



Loop Handshaking



Controllo di flusso software

21

- Per la trasmissione asincrona su RS232 è possibile non fare uso dei segnali di controllo per il controllo di flusso.
- A tale scopo vengono trasmessi byte particolari di start e stop. Questi sono definiti dal codice ASCII (American Standard Code for Information Exchange) come XON (binario:10001) e XOFF (10011)
- Questo metodo è utile quando si trasferiscono dati in formato testo (codice ASCII), meno quando si trasferiscono altri formati.
- Il controllo di flusso software è più lento di quello hardware (handshaking).

UART

22

- UART (**U**niversal **A**synchronous **R**eceiver **T**ransmitter) - ricevitore/trasmittitore asincrono universale - sono chip seriali. Scopo della UART è convertire i byte dal bus parallelo del PC in un flusso seriale di bit e viceversa nel senso inverso.
- Le UART trattano dati divisi in pezzi della dimensione di un byte, che per convenienza è anche la dimensione dei caratteri ASCII
- Ci sono due tipi principali di UART: UART "stupide" (dumb, per es 8250) e UART FIFO (per es. 16550A), . La differenza tra dumb e FIFO (First In, First Out - Il primo che entra è il primo che esce) è che le seconde sono dotate di un buffer (FIFO, per l'appunto) di >16 byte. Nelle prime ad ogni carattere ricevuto veniva generato un interrupt, nelle seconde, l'interrupt viene generato solo a buffer pieno (o quasi pieno – parametro "Trigger level" (TL) aggiustabile), e, a quel punto viene svuotato il buffer.
- Se ci sono meno di TL byte nel buffer, un interrupt viene comunque generato se non arrivano altri caratteri entro un tempo di "timeout".
- Si veda per esempio: <http://www.lammertbies.nl/comm/info/serial-uart.html>

Universal Serial Bus

23

- **A representative peripheral interface**
- **Universal Serial Bus (USB)** provides a serial bus standard for connecting devices, usually to a computer, but it also is in use on other devices such as set-top boxes, game consoles and PDAs. (wikipedia.org)



What USB Can Do

24

- USB is a likely solution any time you want to use a computer to communicate with devices outside the computer.
- The interface is suitable for one-of-kind and small-scale designs as well as mass-produced, standard peripheral types.
- Device, male connector
- Computer, female connector
- Hub



Comparison

Interface	Format	Number of Devices (maximum)	Length (maximum, feet)	Speed (maximum, bits/sec.)	Typical Use
USB	asynchronous serial	127	16 (or up to 96 ft. with 5 hubs)	1.5M, 12M, 480M	Mouse, keyboard, disk drive, modem, audio
RS-232 (EIA/TIA-232)	asynchronous serial	2	50-100	20k (115k with some hardware)	Modem, mouse, instrumentation
Parallel Printer Port	parallel	2 (8 with daisy-chain support)	10-30	8M	Printers, scanners, disk drives
IEEE-1394 (FireWire)	serial	64	15	400M (increasing to 3.2G with IEEE-1394b)	Video, mass storage

Benefits for Users

26

- **Ease of Use**
Ease of use was a major design goal for USB, and the result is an interface that's a pleasure to use for many reasons:
- **One interface for many devices.**
USB is versatile enough to be usable with many kinds of peripherals. Instead of having a different connector type and supporting hardware for each peripheral, one interface serves many.
- **Automatic configuration.**
When a user connects a USB peripheral to a computer, its OS automatically detects the peripheral and loads the appropriate software driver.
- **Hot pluggable**
We can connect and disconnect a peripheral whenever you want, whether or not the system and peripheral are powered, without damaging the PC or peripheral. The operating system detects when a device is attached and readies it for use.
- **No power supply required** (sometimes).
A peripheral that requires up to 500 milliamperes can draw all of its power from the bus instead of having its own supply..

- USB 1.0 specification introduced in 1994
- USB 2.0 specification finalized in 2001
- Became popular due to cost/benefit advantage
 - Eg. IEEE 1394 – high bandwidth, high cost
- Three generations of USB
 - USB 1.0
 - USB 2.0
 - USB 3.0 and WUSB

USB 2.0

28

- A big step in USB's evolution was version 2.0
 - support for *much* faster transfers.
 - a 40-times increase was found to be feasible, for a bus speed of 480 Megabits per second.
- How to use it
 - USB 2.0 is backwards compatible with USB 1.1.
 - Version 2.0 peripherals can use the same connectors and cables as 1.x peripherals.
 - To use the new, higher speed, peripherals must connect to 2.0-compliant hosts and hubs. 2.0 hosts and hubs can also communicate with 1.x peripherals.
 - A 2.0-compliant hub with a slower peripheral attached will translate as needed between the peripheral's speed and high speed.
 - This increases the hub's complexity but makes good use of the bus time without requiring different hubs for different speeds.

USB 3.0 Overview



29

- Also referred to as SuperSpeed USB
- Speeds 10x faster than 2.0 (5 Gbps in controlled test environment)
 - Transfer of 25 GB file in approx 70 seconds (see chart)
- Extensible – Designed to scale > 25Gbps
- Optimized power efficiency
 - No device polling (asynchronous notifications)
 - Lower active and idle power requirements
- Backward compatible with USB 2.0
 - USB 2.0 device will work with USB 3.0 host
 - USB 3.0 device will work with USB 2.0 host

	Song / Pic	256 Flash	USB Flash	SD-Movie	USB Flash	HD-Movie
	4 MB	256 MB	1 GB	6 GB	16 GB	25 GB
USB 1.0	5.3 sec	5.7 min	22 min	2.2 hr	5.9 hr	9.3 hr
USB 2.0	0.1 sec	8.5 sec	33 sec	3.3 min	8.9 min	13.9 min
USB 3.0	0.01 sec	0.8 sec	3.3 sec	20 sec	53.3 sec	70 sec

USB 3.0

30

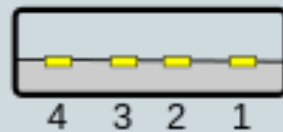
- "SuperSpeed" bus provides a fourth transfer mode at 5.0 Gbit/s
- Communication is full-duplex during SuperSpeed; (in the modes supported previously, by 1.x and 2.0, communication is half-duplex, with direction controlled by the host.)

USB versus IEEE-1394

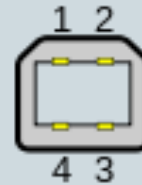
- IEEE-1394 is best suited for video and other links where speed is essential or a host PC isn't available.
- USB is best suited for typical peripherals such as keyboards, printers, scanners, and disk drives as well as low- to moderate-speed, cost-sensitive applications. For many devices, either interface would work.
- With USB, a single host controls communications with many peripherals.
- IEEE-1394 uses a peer-to-peer model, where peripherals can communicate with each other directly.
- A single communication can also be directed to multiple receivers.
- IEEE-1394's 400 Megabits per second is more than 30 times faster than USB 1.x's 12 Megabits per second.
- As USB is getting faster with version 2.0, IEEE-1394 is getting faster with the proposed IEEE-1394.b.

Physical Appearances

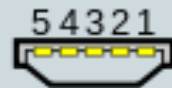
32



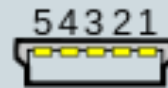
Type A



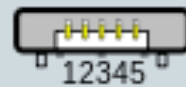
Type B



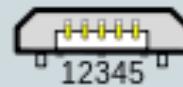
Mini-A



Mini-B



Micro-A

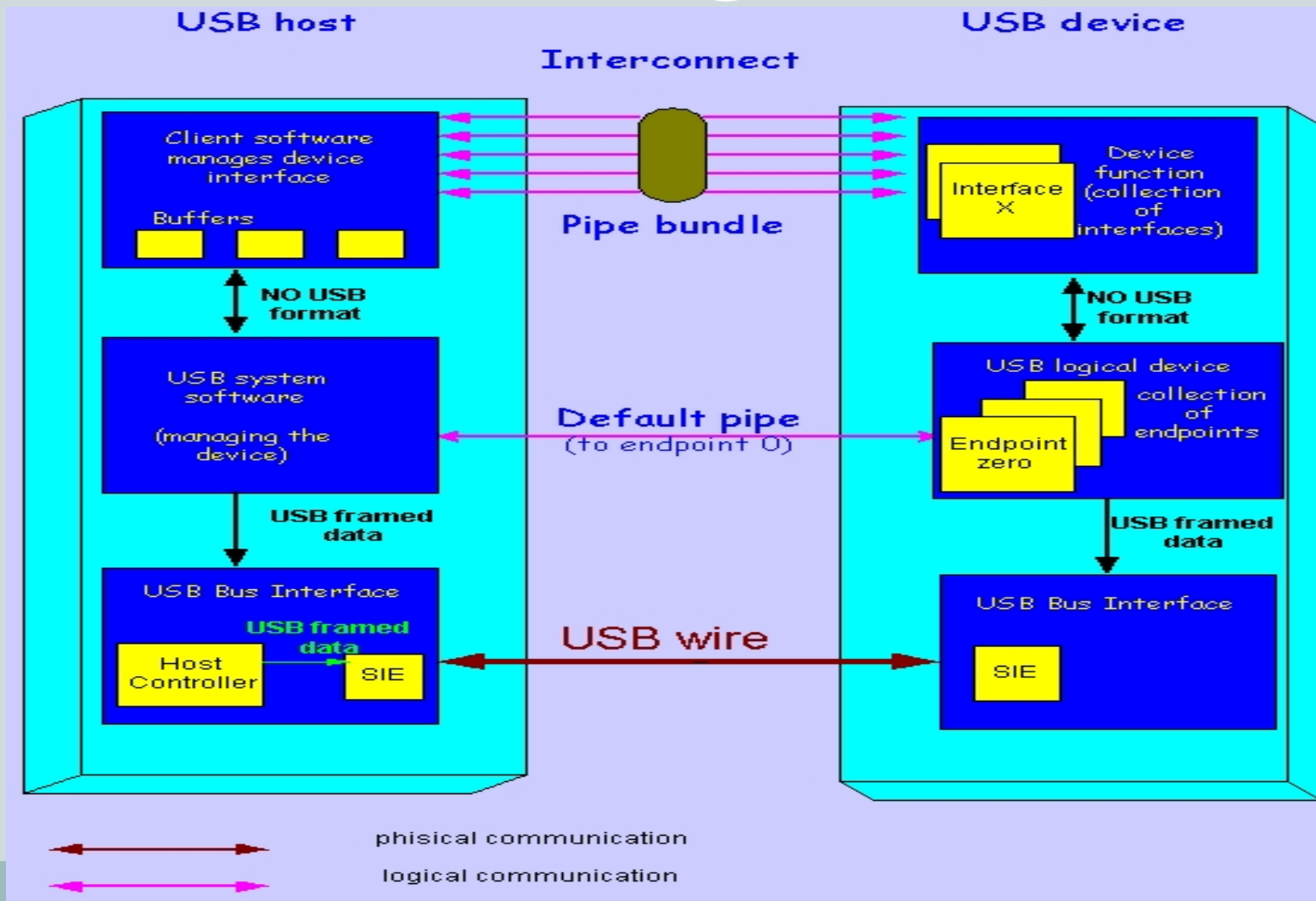


Micro-B

- Type A connectors on host devices that supply power
- Type B connectors on target devices that receive power.

Communication Flow

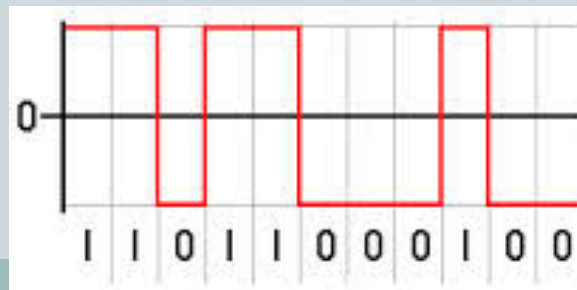
33



Signaling On The Bus

34

- The USB cable is 4 wire cable
- Signal on the bus is done by signaling over two wires (D_+ and D_-)
- Data encoding and decoding is done using NRZI (Non Return to Zero Inverted)
 - a 0 bit is transmitted by toggling the data lines
 - a 1 bit is transmitted by leaving the data lines as-is.



SIE

35

- SIE : Serial Interface Engine

It is part of both the host's and the device's physical layer

=>Serialization and De-serialization

=>Encoding and Decoding

=>Generate(for out) and Verify(for in) CRC

=>Detect PID

- HC : Host Controller

=>It is an additional hardware to ensure that everything which is transmitted on the bus is correct

=>It serves both the USB and the host and has the same functionality in every USB system

Protocol Engine Layer

37

- Responsible for the translating the data between the application layer and the USB transactions protocol .
- Two Roles
 - =>USB System Software (in the USB host)
 - =>USB Logical Device (in the USB device)

The USB System SW

38

- Compose of The Host Controller Driver and The USB Driver
 - Responsible for
 - =>Bandwidth allocation
 - =>bus power management
- Two of above are in order to enable devices to access the bus

The USB Logical Device

39

- Compose of a collection of independent endpoints
- Each endpoint has an unique Endpoint Number and is unidirectional(except endpoint zero and has two type--In/Out)
- Default pipe is associated with endpoint zero

The Application Layer

40

The Application Layer

41

- Host end : Client Software
 - = >Manages the appropriate interface by transferring data from its buffers to the endpoint with the appropriate interface
- Device end : Function
 - = >Composed of interfaces and controls the functionality of the device

Pipes

42

- The logic communication between the client software on the host and the function on the device is done through **pipes**
- It is a association between a specific endpoint on the device and the appropriate software in the host

Pipes (continued)

43

- An endpoint is the source or destination of the data that transmitted on the USB cable
- An endpoint of a pipe is addressable with a tuple (device_address, endpoint_number)
- Two direction

=>OUT :

data flows from the host to the device

=>IN :

data flows from the device to the host

USB Packet Types

USB PID bytes

Type	PID value (<i>msb</i> -first)	Transmitted byte (<i>lsb</i> -first)	Name	44	Description
<i>Reserved</i>	0000	0000 1111			
Token	1000	0001 1110	SPLIT		High-bandwidth (USB 2.0) split transaction
	0100	0010 1101	PING		Check if endpoint can accept data (USB 2.0)
Special	1100	0011 1100	PRE		Low-bandwidth USB preamble
			ERR		Split transaction error (USB 2.0)
Handshake	0010	0100 1011	ACK		Data packet accepted
	1010	0101 1010	NAK		Data packet not accepted; please retransmit
	0110	0110 1001	NYET		Data not ready yet (USB 2.0)
	1110	0111 1000	STALL		Transfer impossible; do error recovery
Token	0001	1000 0111	OUT		Address for host-to-device transfer
	1001	1001 0110	IN		Address for device-to-host transfer
	0101	1010 0101	SOF		Start of frame marker (sent each ms)
	1101	1011 0100	SETUP		Address for host-to-device control transfer
Data	0011	1100 0011	DATA0		Even-numbered data packet
	1011	1101 0010	DATA1		Odd-numbered data packet
	0111	1110 0001	DATA2		Data packet for high-bandwidth isochronous transfer (USB 2.0)
	1111	1111 0000	MDATA		Data packet for high-bandwidth isochronous transfer (USB 2.0)

Four Types of Data Transfers (Pipes)

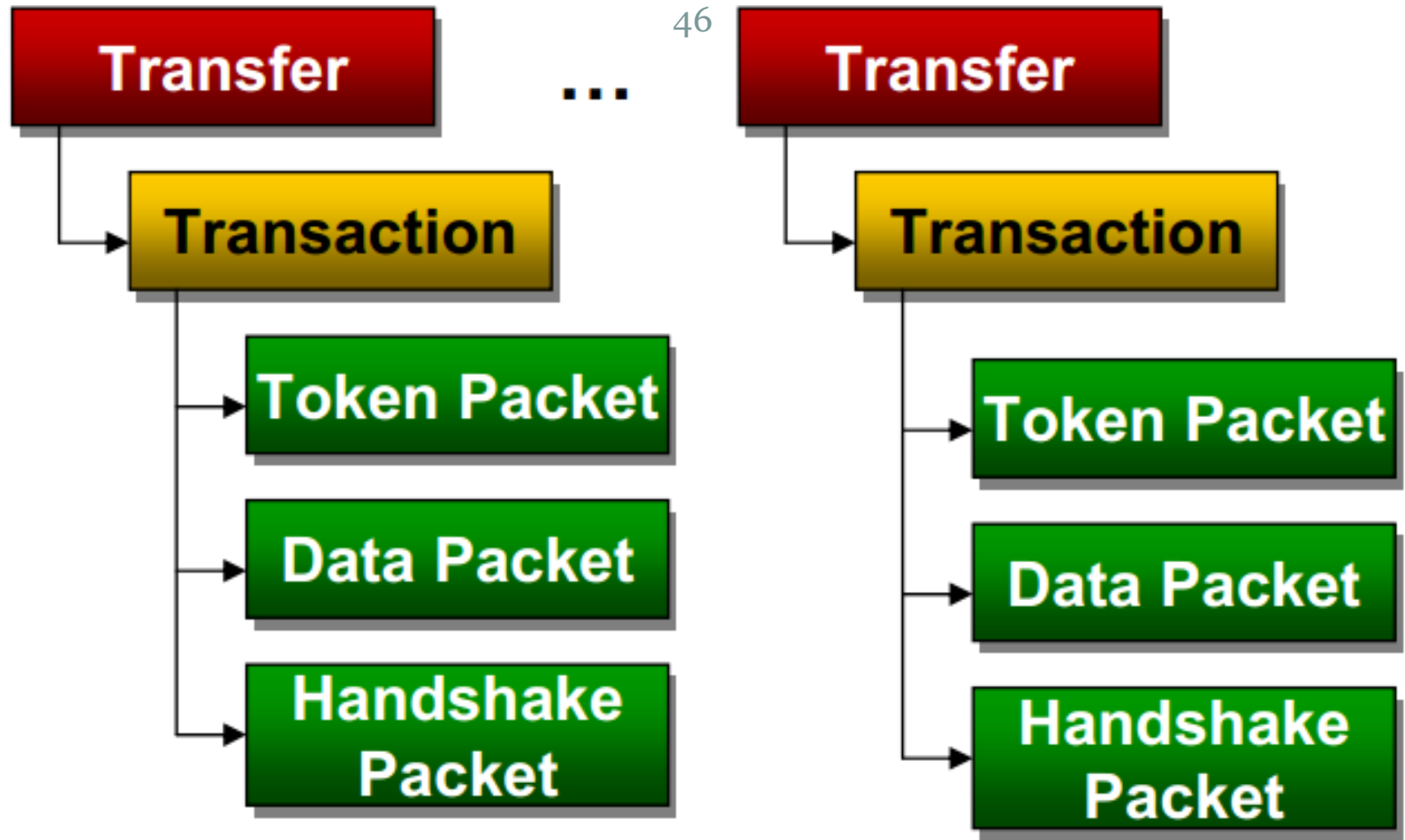
45

- **Message Pipe**

- **Control transfers** Control transfers are bidirectional transfers used by the USB system software mainly to query, configure, and issue certain generic commands to USB devices. Control transfers typically take place between the host computer and the USB device's endpoint 0, but your control transfers might use other endpoints.

- **Stream Pipe**

- **Interrupt transfers** are used when a peripheral wants to be “polled” by the host periodically to see if it has data to send to the host. Keyboards, mice, and joysticks are examples of devices that typically use interrupt transfers.
- **Bulk transfers** are used to move data between the host system and the peripheral when data integrity is more critical than data latency, and they also include error checking and retries if errors are detected. Printers, scanners, and storage devices are examples of devices that depend primarily on bulk transfers.
- **Isochronous transfers** are used for moving “real-time” data. In these transfers, the streaming of the data is more critical than the accuracy of the data. There is no error-checking or retries associated with isochronous transfers. Web-cams, speakers and microphones are examples of devices that utilize isochronous transfers.



Transfer Type	Stages (Transactions)	Phases (Packets)	Comments
Control	Setup	Token	<ul style="list-style-type: none"> ◆ Enables host to read configuration information, set addresses and select configurations ◆ Only transfer that is required to be supported by peripherals ◆ Has both IN and OUT transfers to a single endpoint
		Data	
		Handshake	
	Data (IN or OUT) (optional)	Token	
		Data	
		Handshake	
	Status (IN or OUT)	Token	
		Data	
		Handshake	
Bulk	Data (IN or OUT)	Token	<ul style="list-style-type: none"> ◆ Non-critical data transfers ◆ Bandwidth allocated to the host ◆ Good for file transfer where time critical data is not required
		Data	
		Handshake	
Interrupt	Data (IN or OUT)	Token	<ul style="list-style-type: none"> ◆ Periodic transfers on the time base conveyed during enumeration ◆ Host guarantees attention before this elapsed time
		Data	
		Handshake	
Isochronous	Data (IN or OUT)	Token	<ul style="list-style-type: none"> ◆ Guaranteed delivery time of packets for data streaming ◆ No-retransmitting of data allowed
		Data	

USB On-The-Go (OTG)

48

- Released in December 2006.
- USB On-The-Go makes it possible for two USB devices to communicate with each other without requiring a separate USB host.
- In practice, one of the USB devices acts as a host (device roles change) for the other device.
- For example:
 - A usb flash drive (peripheral) served by a printer (host);
 - A keyboard (peripheral) connected to a mobile phone (host)

USB OTG supported by Micro-AB

49

- An OTG product must have a single Micro-AB receptacle and no other USB receptacles.
- When attached to a PC, (host)
 - an OTG device requires a cable which has a USB Standard-A plug on one end
 - and a Micro-B plug on the other end.
- (Peripheral)

- In order to attach a peripheral to an OTG device (now it serves as a host)

50

- the peripheral either needs to have a cable ending in a Micro-A plug which is inserted into the OTG device's Micro-AB (same physical appearance as Micro-A receptacle) with receptacle
- or the OTG device itself needs an adapter cable which has a Micro-A plug on one end and a Standard-A receptacle on the other. The adapter cable enables any standard USB peripheral to be attached to an OTG device. (see below)



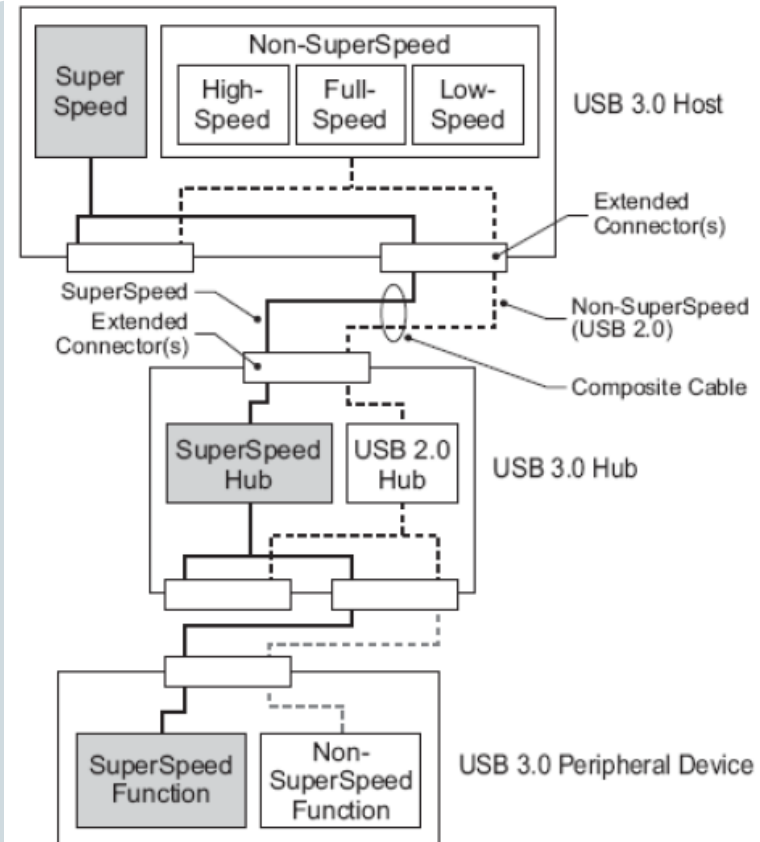
- In order to attach two OTG devices together requires either a cable with a Micro-B plug at one end and a Micro-A plug at the other or can be achieved using a combination of the PC cable and adapter cable.

USB 3.0 Bus Architecture



52

- Operates concurrently with USB 2.0 (Dual bus architecture)
 - Mechanically and electrically backward/forward compatible
 - Devices configured at fastest signaling rate
 - Hubs contain additional ports
- Speed and power efficiency
 - Non polling reduces power consumption
 - Additional data lines included to increase speed
 - Efficiency of bandwidth – simultaneous communication between host and device
 - Dedicated in and out lines allow communication between host and device



Note: Simultaneous operation of SuperSpeed and non-SuperSpeed modes is not allowed for peripheral devices.

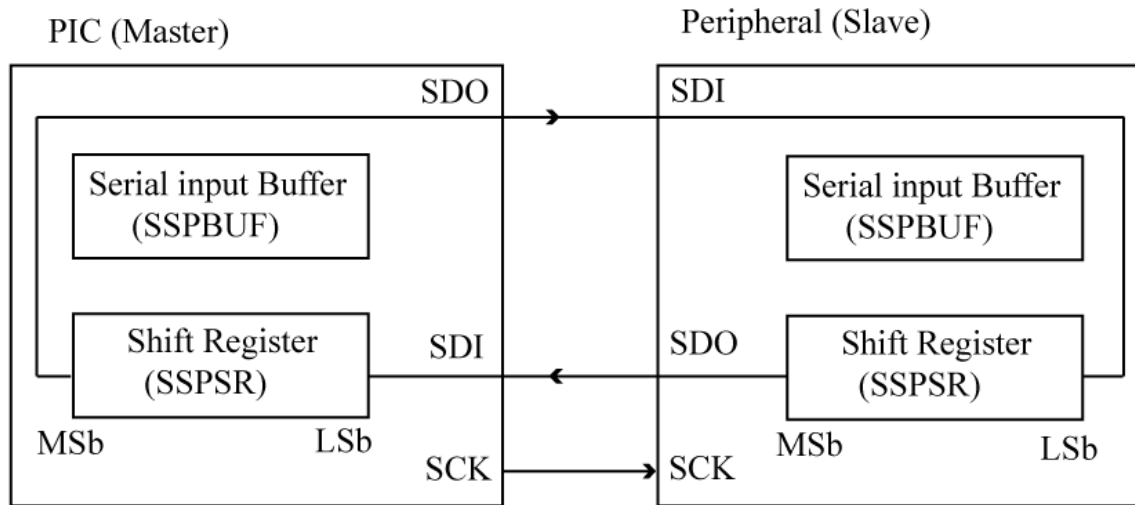
Synchronous Serial IO

53

- Send a separate clock line with data
 - SPI (serial peripheral interface) protocol
 - I²C (or I2C) protocol
- Encode a clock with data so that clock be extracted or data has guaranteed transition density with receiver clock via Phase-Locked-Loop (PLL)
 - IEEE Firewire (clock encoded in data)
 - USB (data has guaranteed transition density)

Serial Peripheral Interface (SPI)

54

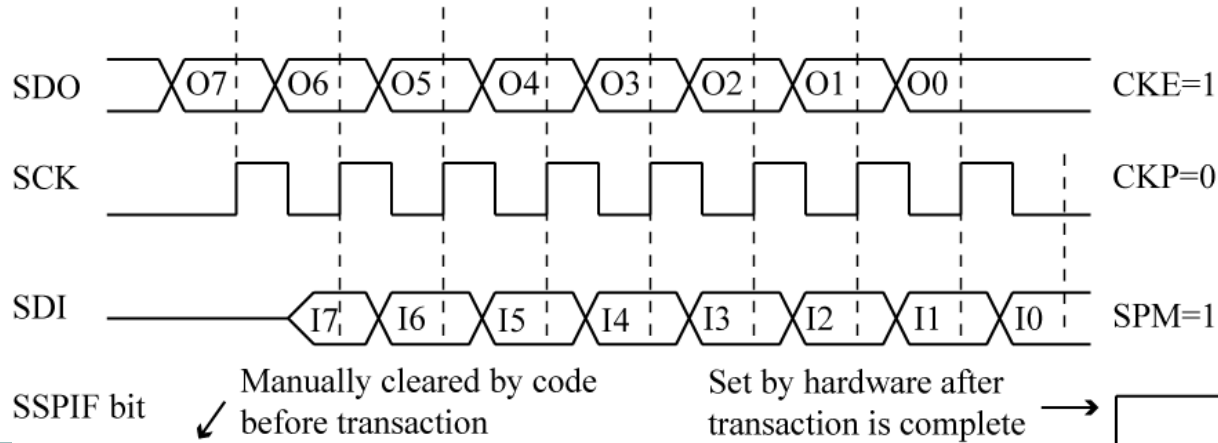


SDI: data in

SDO: data out

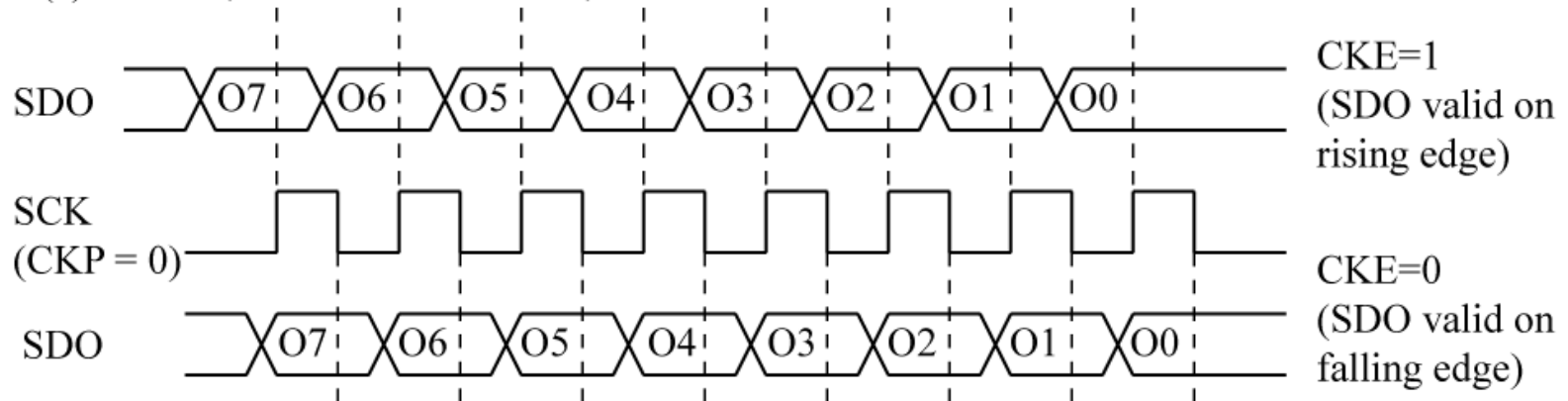
SCK: clock

Data sent MSb first; received data clocked in as transmitted data clocked out

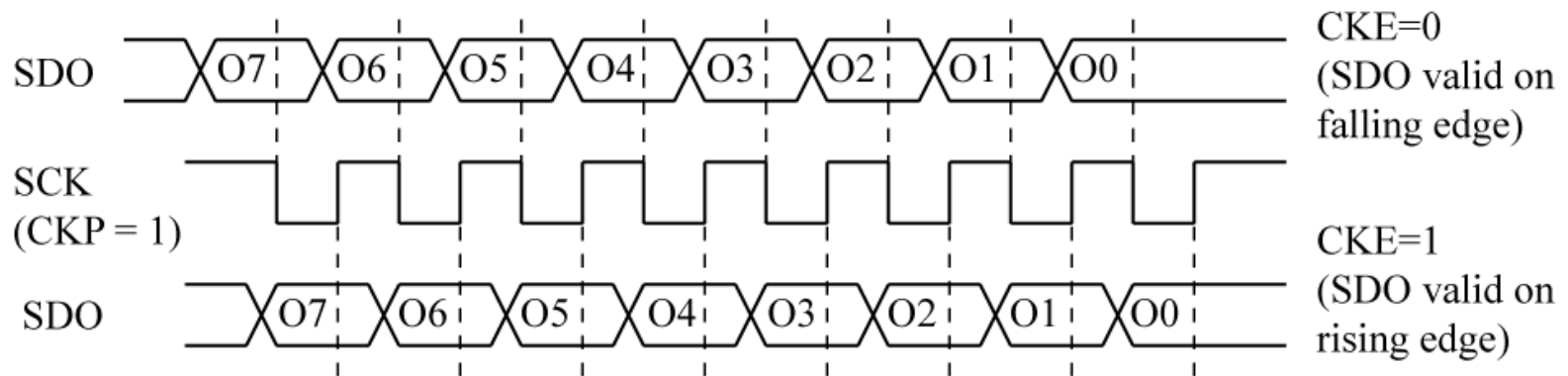


CKE configuration bit allows either falling or rising edge of clock to be used, while CKP selects clock polarity.

(a) CKP = 0, CKE = 1 and CKP = 0, CKE = 0 cases

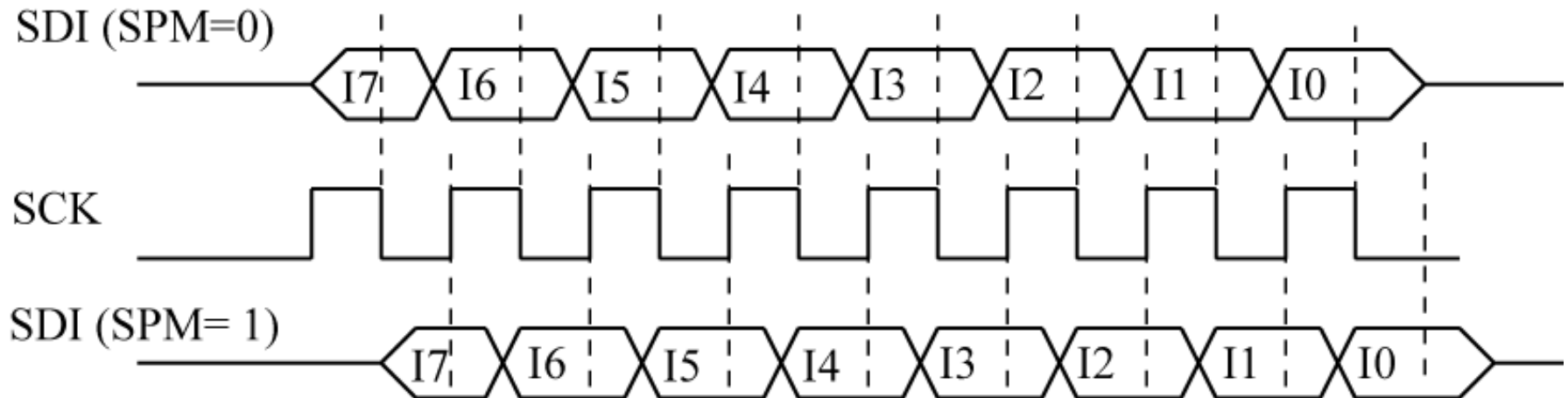


(b) CKP = 1, CKE = 0 and CKP = 1, CKE = 1 cases

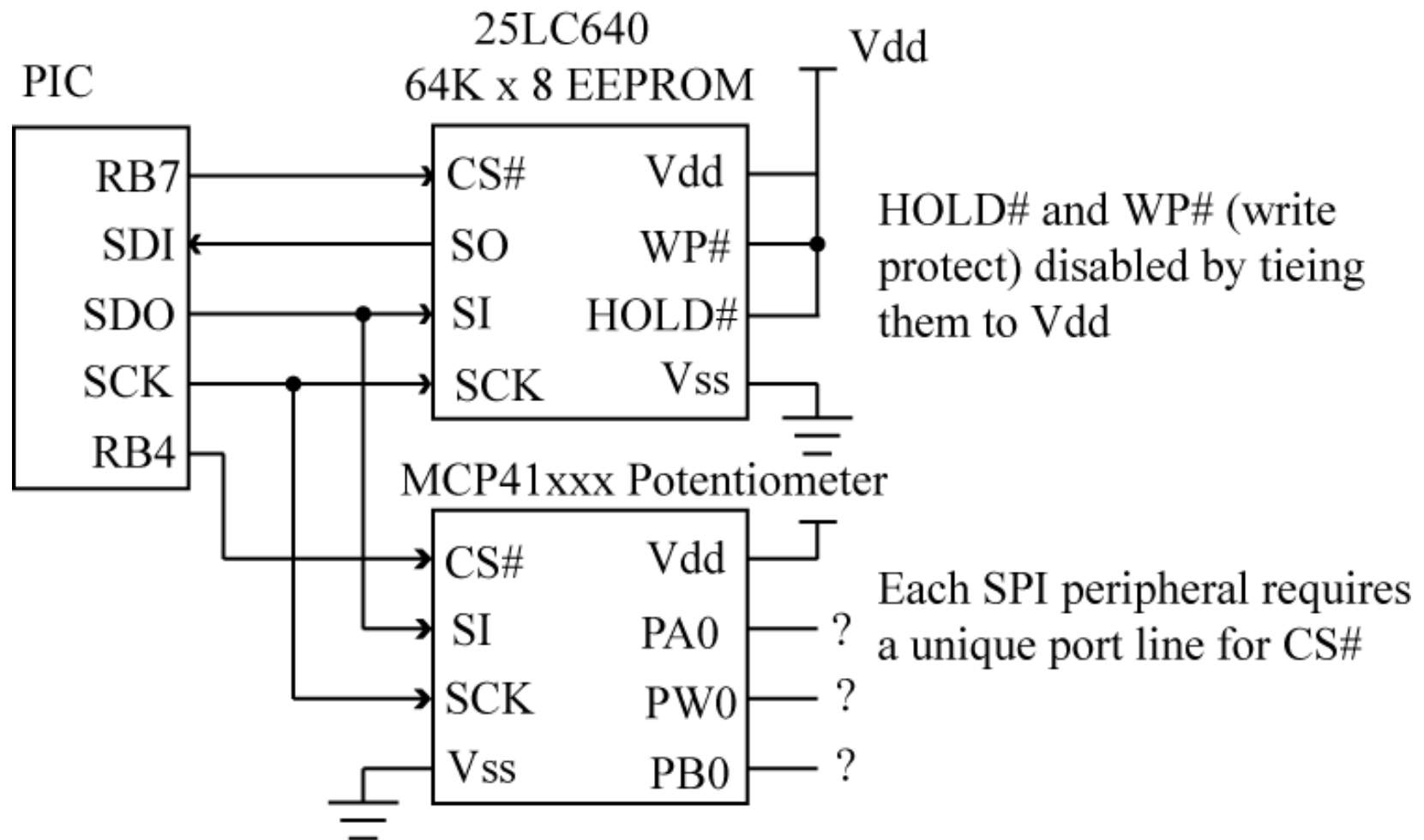


SPM bit selects whether data is sampled in middle of clock period or at end of clock period.

SPM = 0, sample SDI in SCK middle; SPM = 1, sample SDI at SCK end



Between the CKP, CKE, SPM bits there is a lot of flexibility in how data is clocked in. Can make the SPI protocol work with just about any serial device.

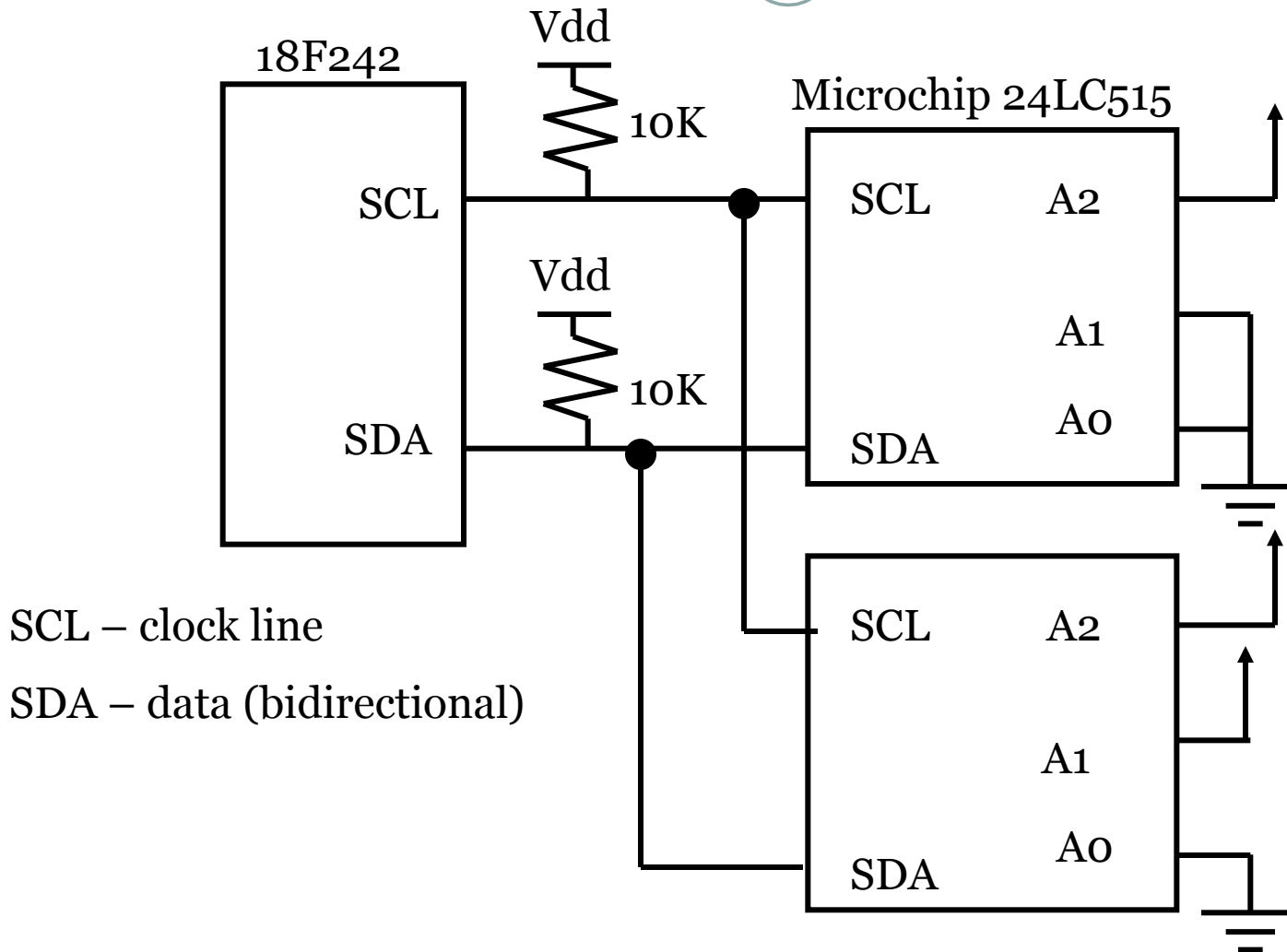


Multiple SPI peripherals each require a separate chip select line via parallel port line. We will concentrate on the I2C serial bus as it does not require use of chip selects.

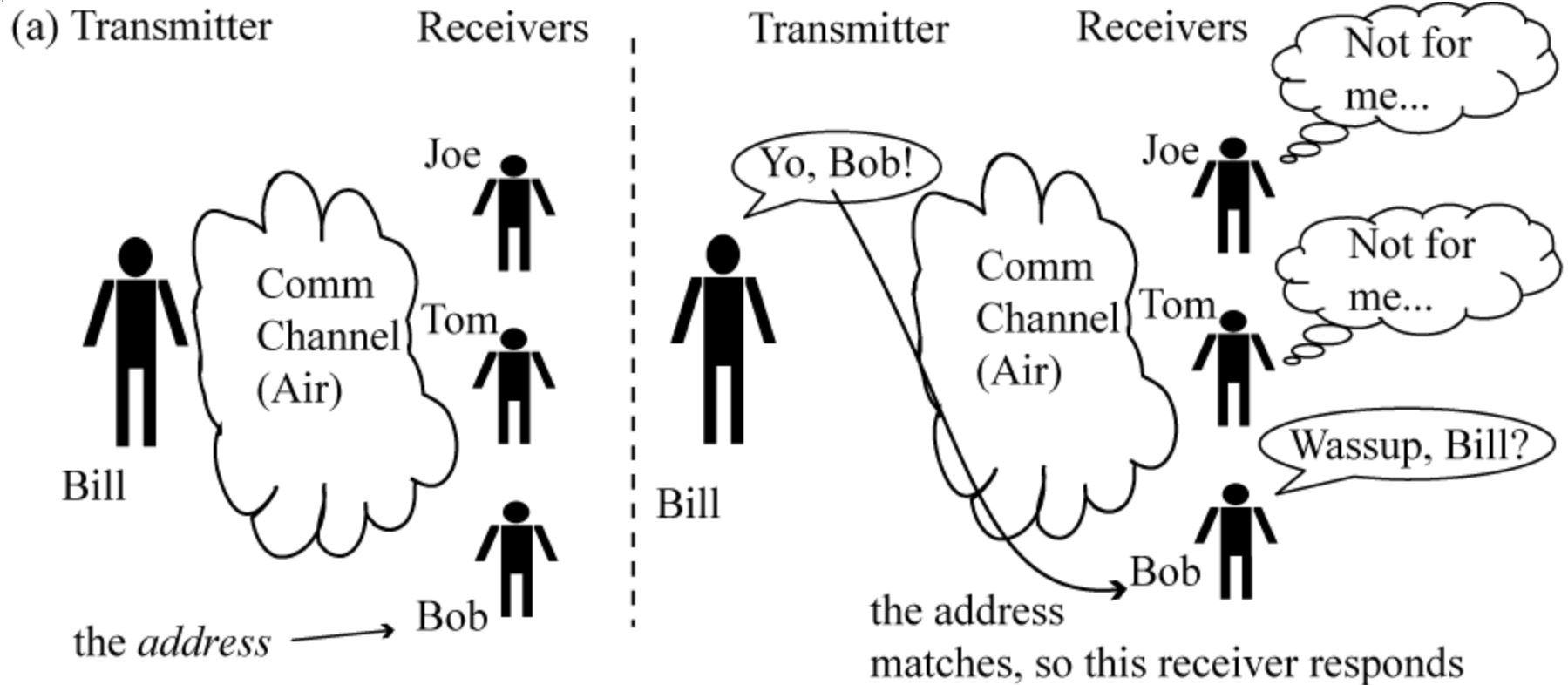
I²C (Inter-Integrated-Circuit) Bus

I²C is a two wire serial interface.

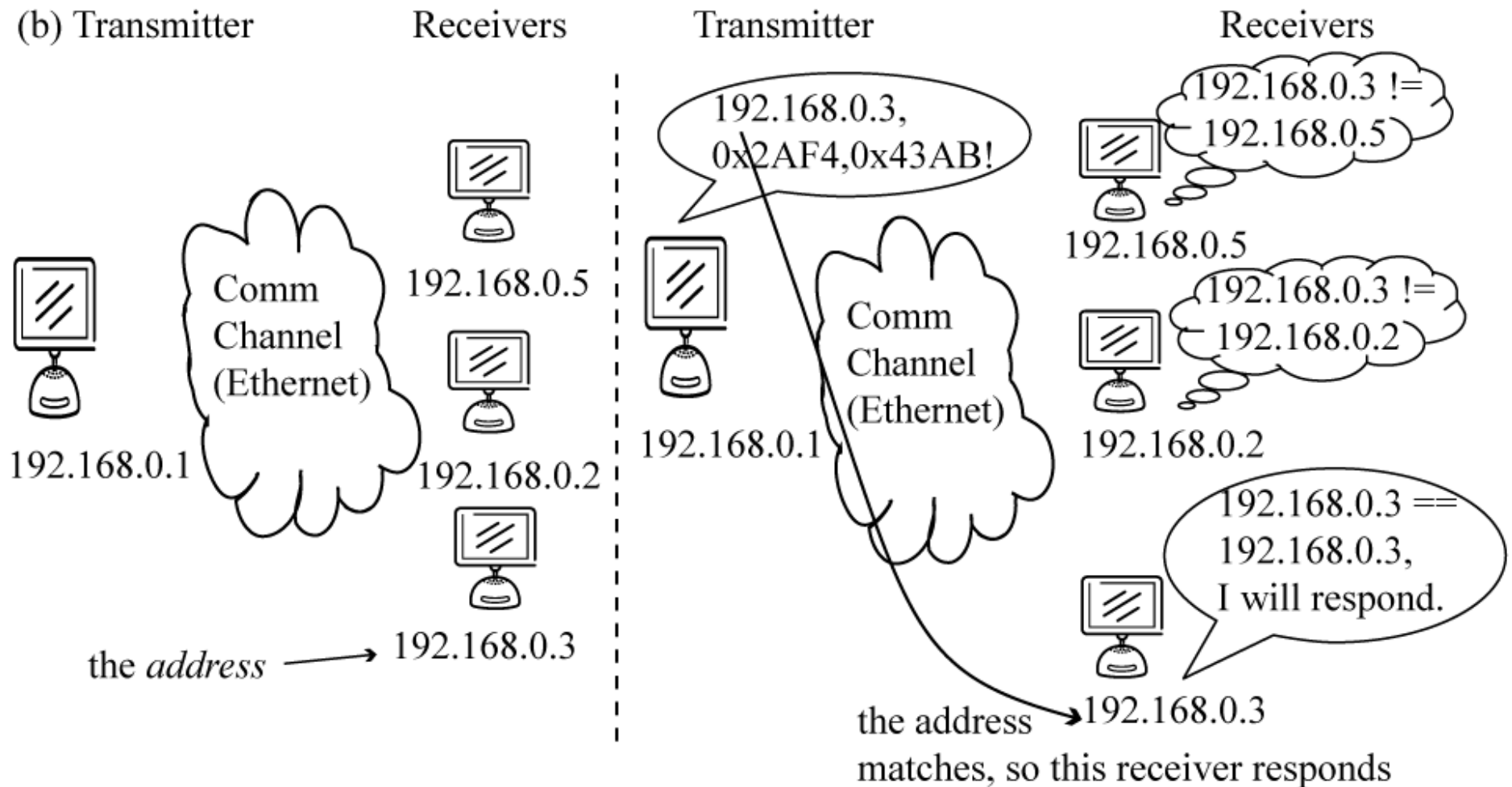
58



What is a bus??



Ethernet is an example of a bus



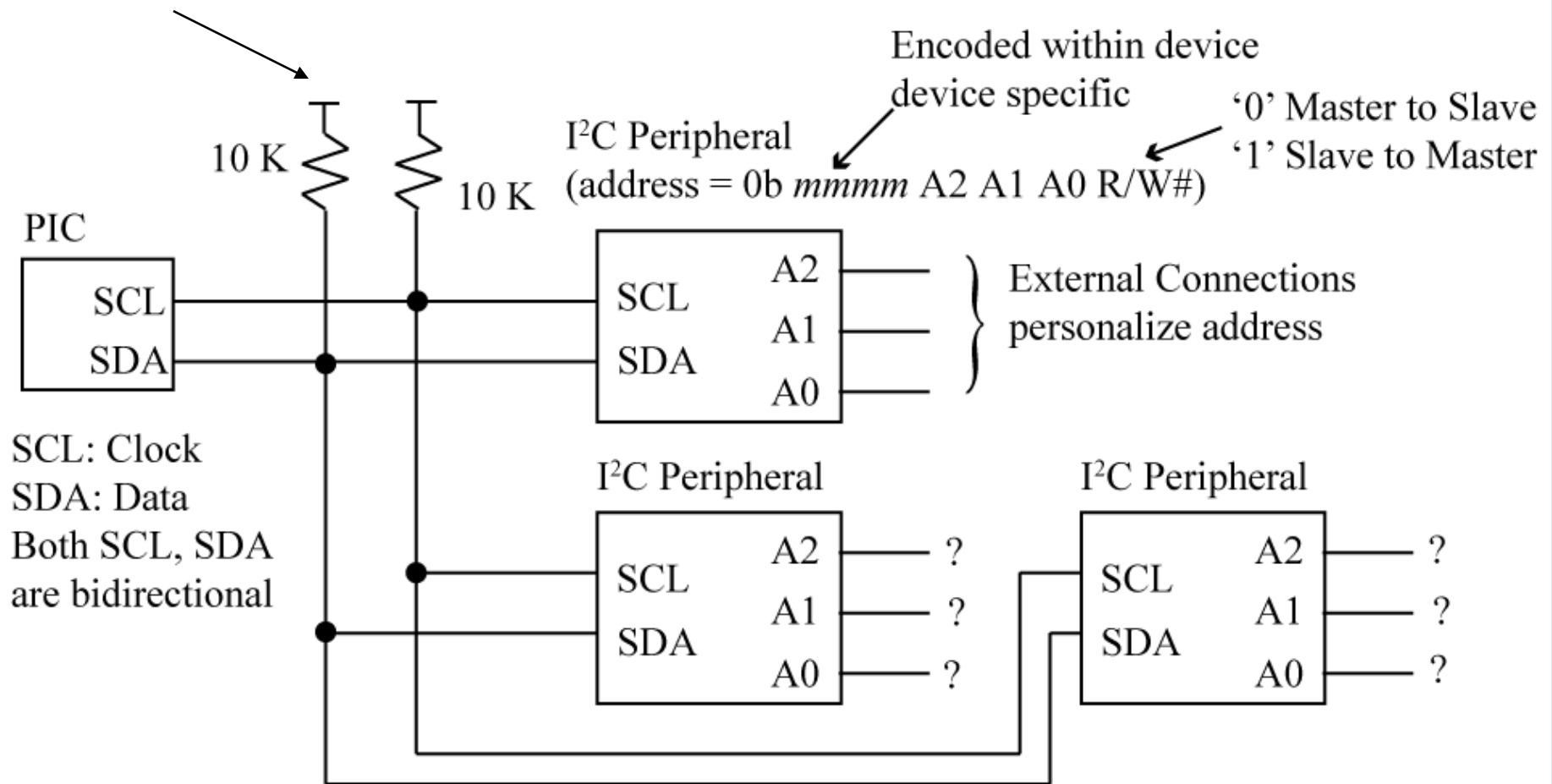
I²C Features

61

- Multiple receivers do not require separate select lines as in SPI
 - At start of each I²C transaction a 7-bit device address is sent
 - Each device listens – if device address matches internal address, then device responds
- SDA (data line) is bidirectional, communication is half duplex
- SDA, SCLK are open-drain, require external pullups
 - Allows multiple bus masters (will discuss this more later).

pullups are needed

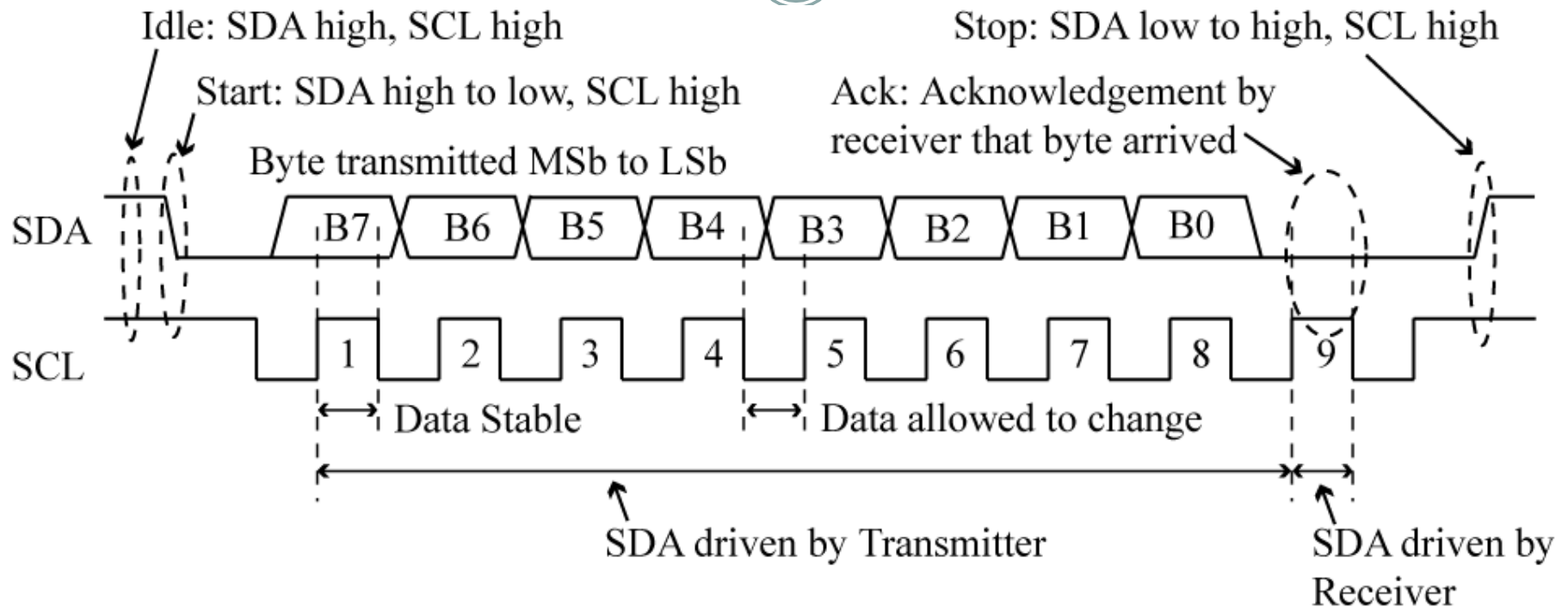
I2C Bus Addressing



No chip selects needed!!!!

I2C Bus Transfer

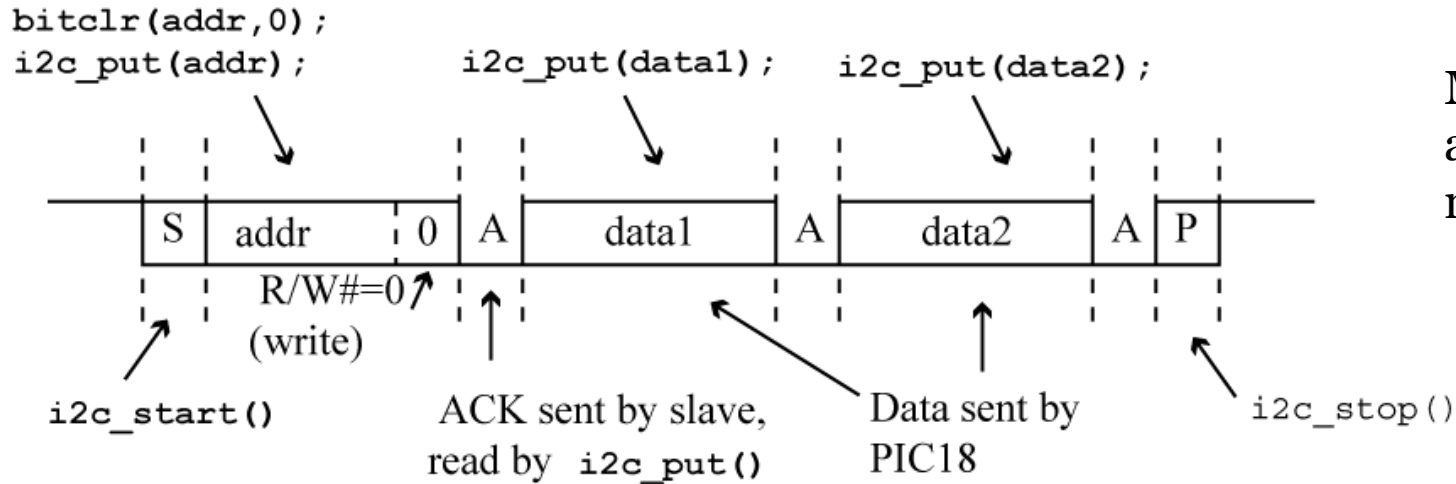
63



Multiple bytes sent in a transaction; every 8 bits has a 9th bit that is an acknowledge.

(a) Write two bytes to slave

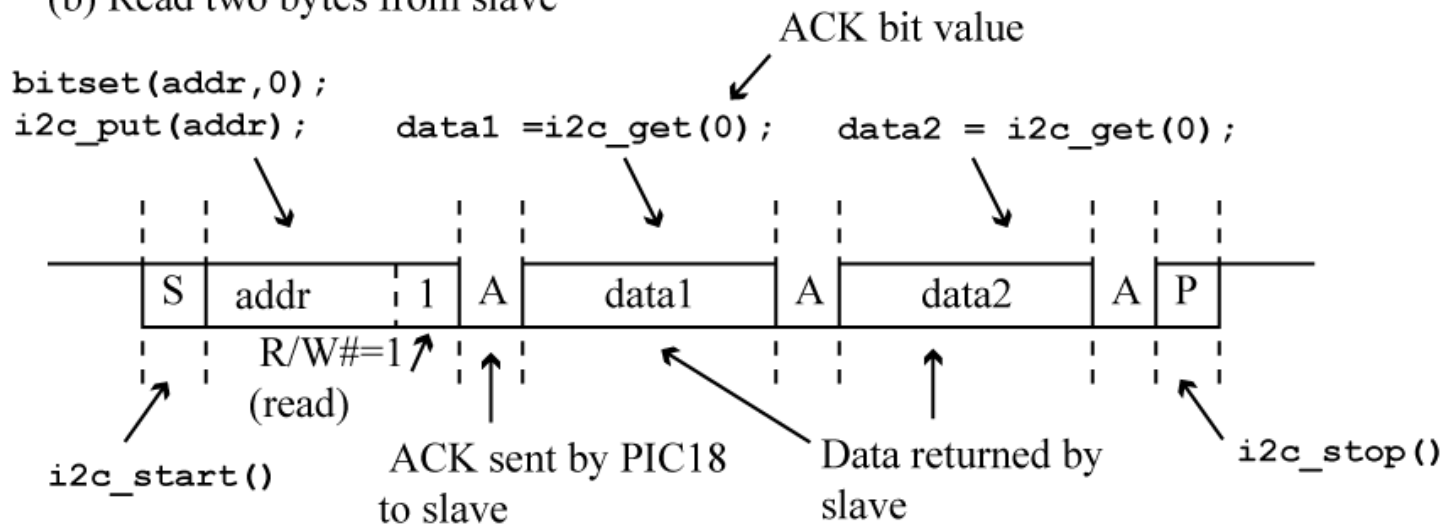
Write (master to slave)



Master initiates all transactions, read or write.

Read (master from slave)

(b) Read two bytes from slave



Example: I²C Serial EEPROM

65

Will use the Microchip 24LC515 Serial EEPROM to discuss I²C operation.

The 24LC515 is a 64K x 8 memory. This would require 16 address lines, and 8 data lines if a parallel interface was used, which would exceed the number of available IO pins our PIC18F242!!!

Putting a serial interface on a memory device lowers the required pin count.

Reduces speed since data has to be sent serially, but now possible to add significant external storage to a low pin-count micro controller.

I²C Device Addressing

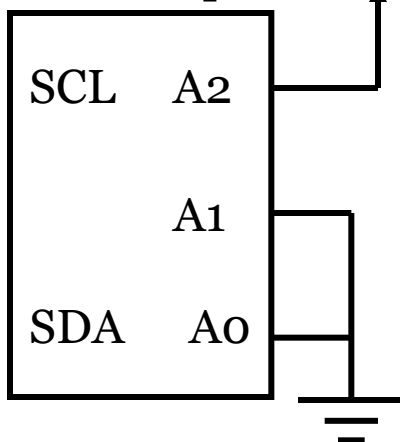
66

Each I²C device has either a 7-bit or 10-bit device address.

We will use an I²C EEPROM and an I²C DAC (Digital-to-Analog Converter, MAX517) in lab. Both of these devices have a 7-bit address.

Upper four bits are assigned by device manufacturer and are hardcoded in the device. Lower three bits are used in different ways by manufacturer.

Microchip 24LC515



LC515 control byte (contains slave address):

7	6	5	4	3	2	1	0
1	0	1	0	Bo	A1	A0	R/ \overline{W}

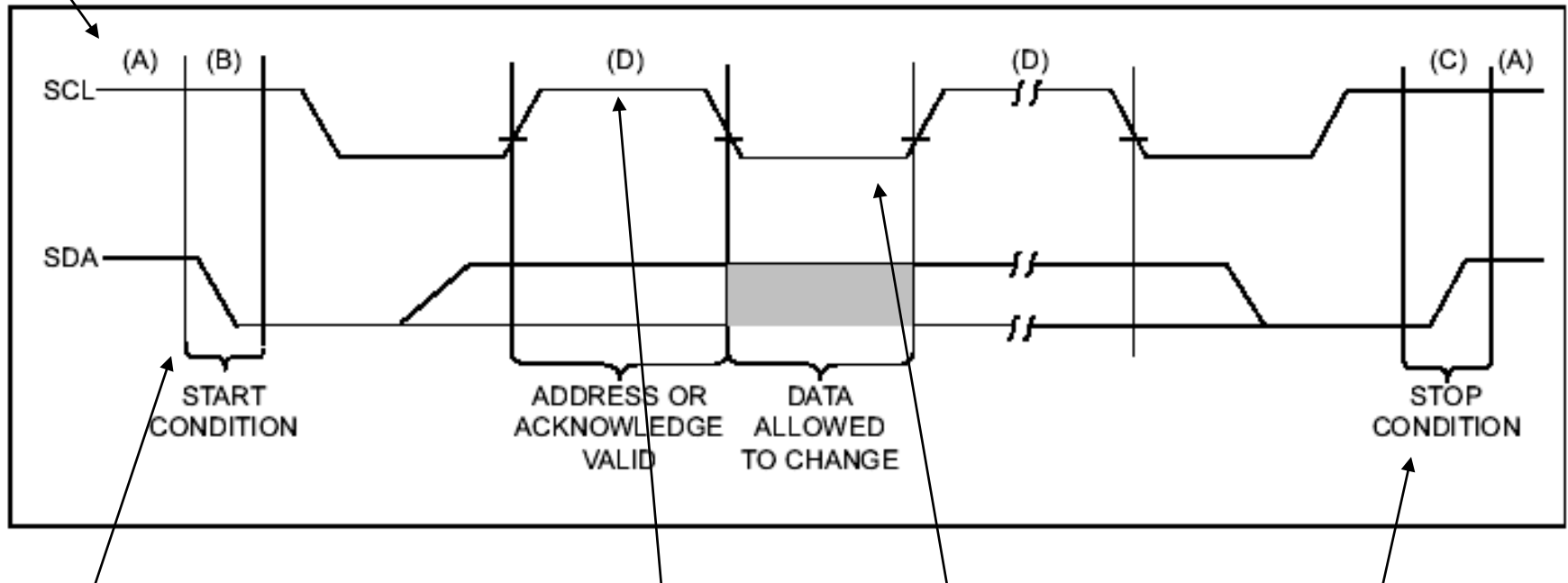
R/ \overline{W} = 1 for
read, 0 for
write.

‘Bo’ is block select (upper/lower 32K). A1, A0 are chip selects, four devices on one bus.

IDLE: SCL, SDA
high.

I²C Transmission

FIGURE 4-1: DATA TRANSFER SEQUENCE ON THE SERIAL BUS



START: high to
low transition on
SDA while SCL
high.

Valid data: While
clock is high, data
valid and stable.

Data changes
while clock is
low.

STOP: low to
high transition
on SDA while
SCL high.

Acknowledgement

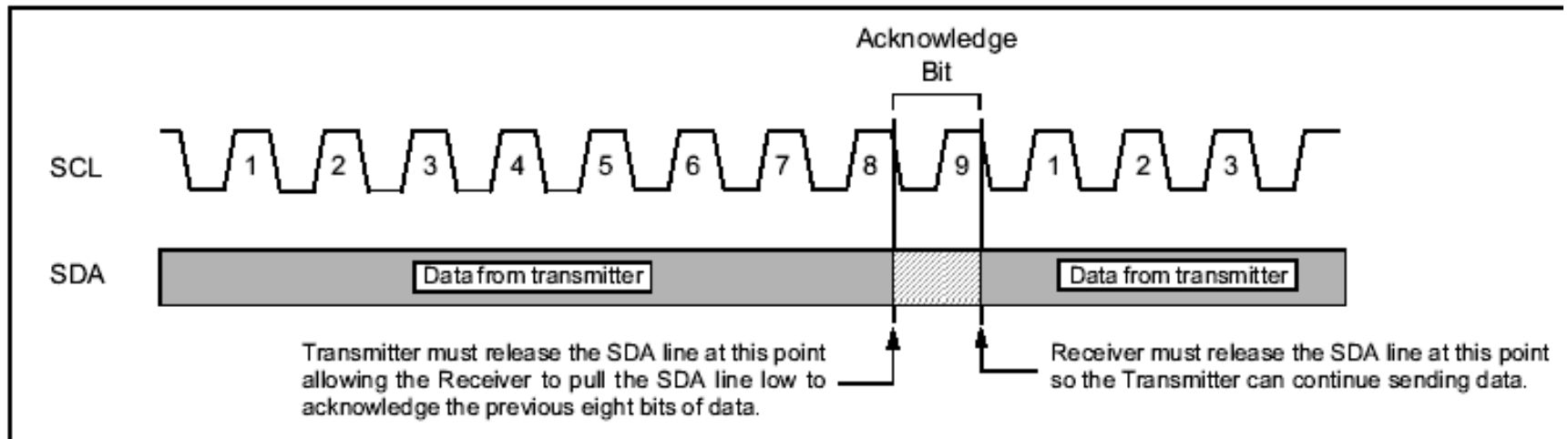
68

ACK sent by slave after every 8-bits received. Master releases line (stops driving), samples line on next clock.

Slave **MUST** pull line low. If Master does not detect ACK, then sets error bit. If Slave does not pull line low, the pullup resistors will pull the line low.

Most common cause of ACK error – incorrect device address.

FIGURE 4-2: ACKNOWLEDGE TIMING

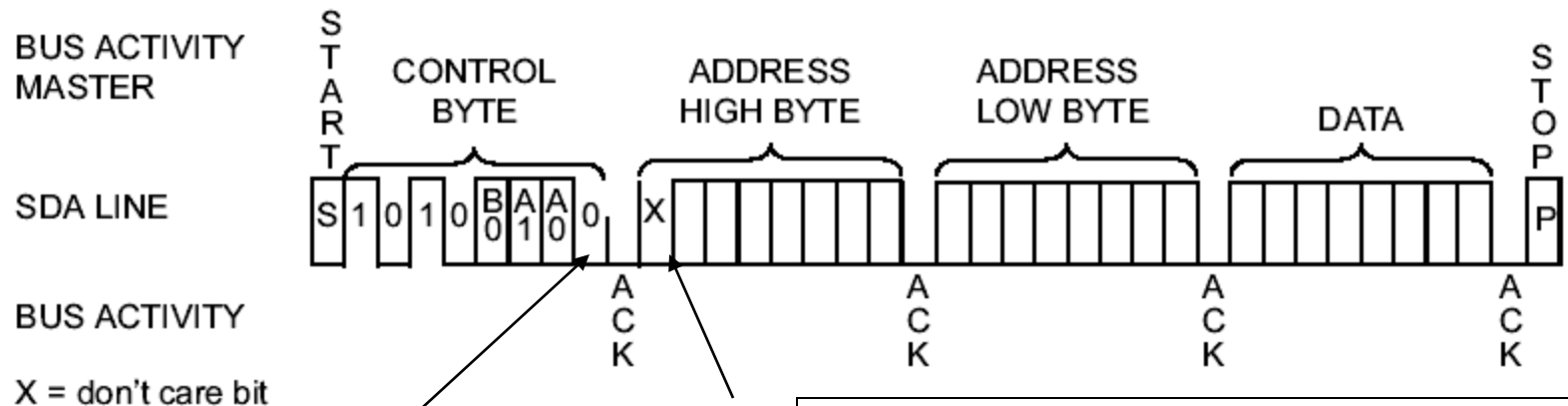


Byte Write Operation

69

- Byte Write: used to write one byte
 - Send Control Byte, High address byte, low address byte, data.
 - After data is sent, takes 5 ms for write to complete
 - SLOOOOOWWWWW....

FIGURE 6-1: BYTE WRITE



'o' indicates write mode.

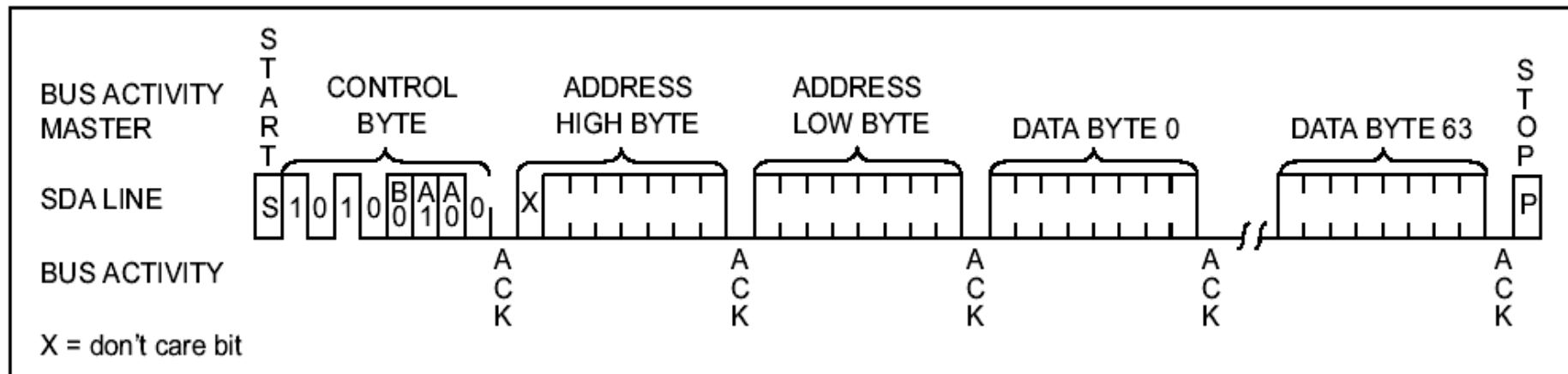
'X' because block select bit chooses high or low 32K.

Page Write Operation

70

- Send a block of 64 bytes, then perform write
 - Send starting address, followed by 64 bytes
 - After 64 bytes sent, wait 5 ms for write to complete
 - Much faster than individual writes

FIGURE 6-2: PAGE WRITE



Address should be on a page boundary. For page size = 64 = 0x40, starting address should be a multiple of 64.

Speed Comparison

71

- Assume a 400 Khz I²C bus, 2.5 us clock period (2.5 e-6)
- Random write:
 - 9 bit transmission = $2.5 \text{ us} * 9 = 22.5 \text{ us}$
 - $5 \text{ ms} + 22.5 \text{ us} * 4 \text{ (control, addhi, addlo, data)} = 5.09 \text{ ms}$
 - For 64 bytes = 325 ms approximately, not counting software overhead.
- Page Write
 - 67 bytes total (control, addhi, addlo, data)
 - $5 \text{ ms} + 67 * 22.5 \text{ us} = 6.5 \text{ ms!!!}$

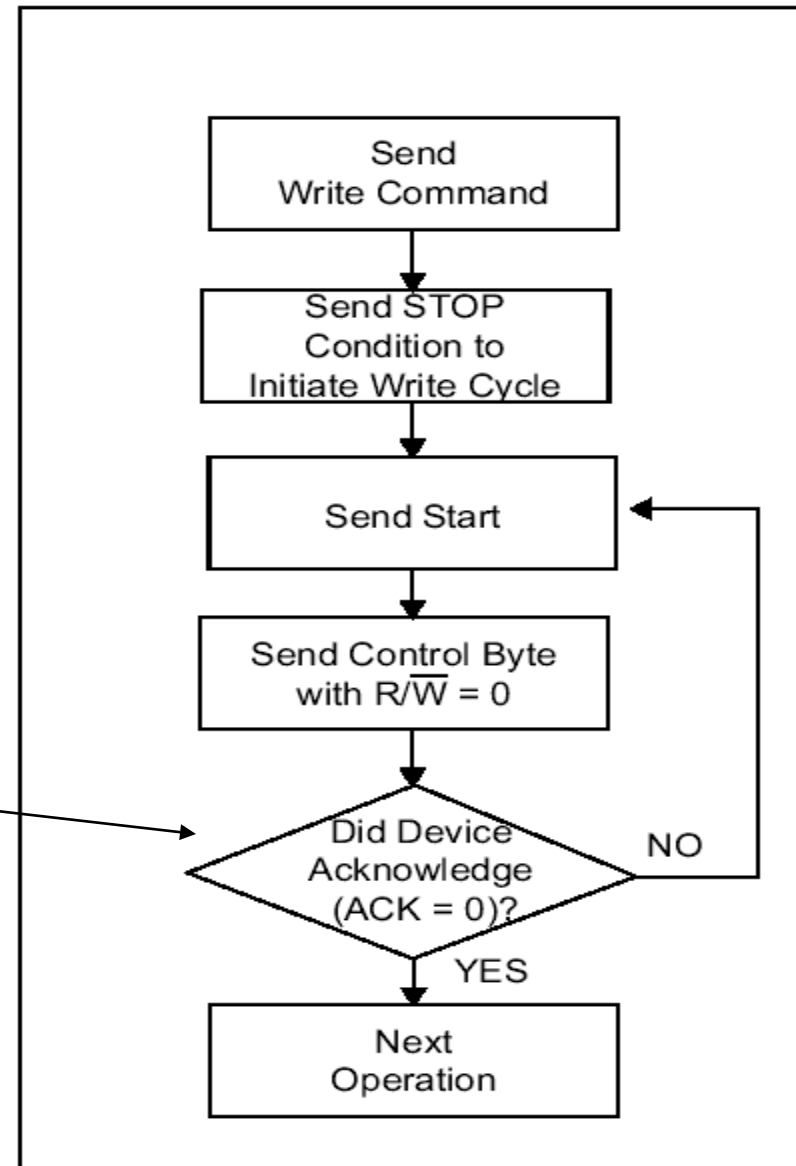
Checking for end-of-write

72

Timing on write is guaranteed to finish after 5 ms. But can end sooner; to determine if write finished use polling method.

No ACK means device is still busy with last write.

FIGURE 7-1: ACKNOWLEDGE POLLING FLOW

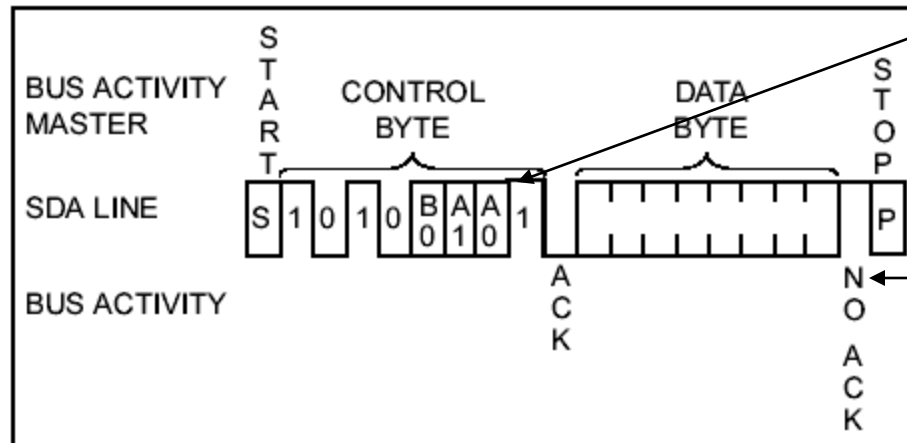


Read Operation: Current Address

73

- An internal counter is used to keep track of last address used
- A current address read uses this address, only sends the command byte
 - Internal counter incremented after read operation

FIGURE 8-1: CURRENT ADDRESS READ



‘1’ indicates read operation

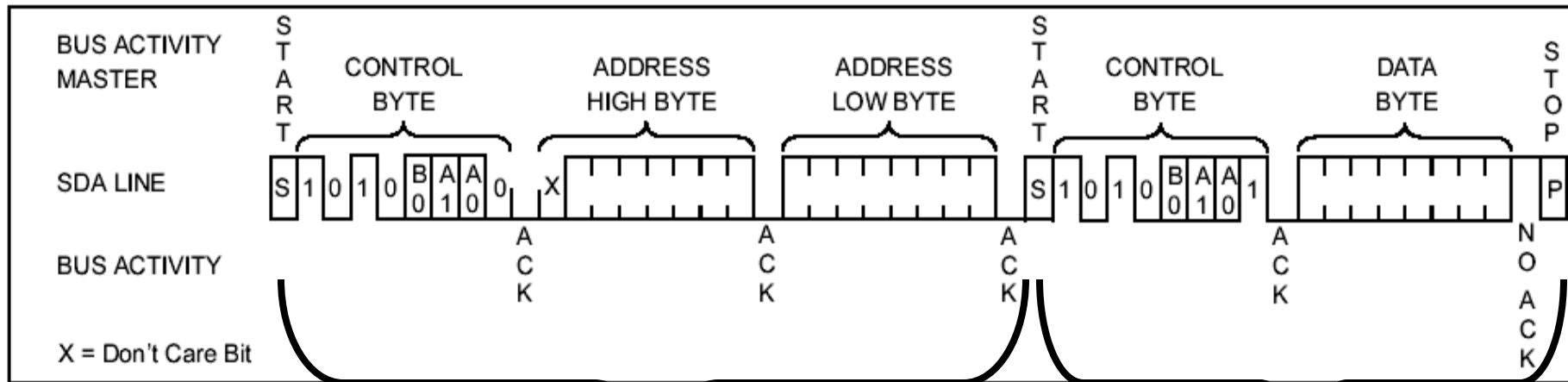
‘no ack’ because slave is driving data back to master.

Random Read Operation

74

- Must first set the internal address counter by starting a write operation, but aborting by not sending data byte

FIGURE 8-2: RANDOM READ



Aborted random write (address only, no data)

Current address read

Sequential Read

75

- Like a current address read, but after Slave sends data byte, Master sends ACK instead of STOP
 - Slave then sends next byte
 - Can do this from 0x0000h to 0x7FFF (lower 32K block). When 0x7FFF is reached, address rolls over to 0x0000
 - Upper block goes from 0x8000 to 0xFFFF; at 0xFFFF address rolls over to 0x8000
 - Internal address counter is only 15 bits wide.

FIGURE 8-3: SEQUENTIAL READ

