# Serial Data Communications II

This lecture note was adapted from the following materials:

1) **Fast and Effective Embedded System Design Applying the ARM mbed** by Rob Toulson and Tim Wilmshurt
2) **AN10216-01 $I^2C$ manual**, Philips Semiconductor
3) Lecture note: **Design of Microprocessor-Based Systems**
   by Dr. Prabal Dutta
4) http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html
5) **LPC1769 User Manual**, NXP

# Synchronous Serial Interface of LPC1769

- SSP 0/1 2 ports for Serial bus digital data + DMA
- SPI 0 1 port for Serial bus digital data
- I2C 0/1/2 3 ports for Serial bus digital data
- I2S 1 port for Serial bus for digital Audio +DMA
- SSP 0/1 can be configured to comply with
  - Motorolla SPI
  - NS Microwire
  - TI SSI
- SSP 0 is intended to used as an alternative for SPI 0. Thus they can not use at the same time (may be they share some hardware)

# LPC1769 SPI : Registers

- SPI Control Register (S0SPCR)
  - control the function of the SPI block and interrupt
- SPI Status Register (S0SPSR)
  - read-only bits to detect completion of a data transfer.
- SPI Data Register (S0SPDR)
  - provide the transmit and receive data bytes.
  - a write to the data register **goes directly** into the internal shift register (no buffering thus cannot write during transmitting)
  - a read of the SPI Data Register returns the value of the **read data buffer** ( this can read at any time)
- SPI Clock Counter Register (S0SPCCR)
  - controls the clock rate ; use when it is a master
- SPI Interrupt Register (S0SPINT) : 1 bit interrupt flag

# LPC1769 SPI : Master Operation

- Make sure it was initialized by
  - Set the S0SPCCR to the desired clock rate.
  - Set the S0SPCR to the desired settings for master mode.
- Step
  - Optionally, verify the SPI setup before starting the transfer.

  1. Write the data to transmitted to the SPI Data Register. This write starts the SPI data transfer.
  2. Wait for the SPIF bit in the SPI Status Register to be set to 1. The SPIF bit will be set after the last cycle of the SPI data transfer.
  3. Read the SPI Status Register.
  4. Read the received data from the SPI Data Register (optional).
  5. Go to step 1 if more data is to be transmitted.

# LPC1769 SPI : Slave Operation

- Make sure it was initialized by
  - Set the S0SPCR to the desired settings for slave mode.
- Step
  - Optionally, verify the SPI setup before starting the transfer.

1. Write the data to transmitted to the SPI Data Register (optional). Can be done when a slave SPI transfer is not in progress.
2. Wait for the SPIF bit in the SPI Status Register to be set to 1. The SPIF bit will be set after the last cycle of the SPI data transfer.
3. Read the SPI Status Register.
4. Read the received data from the SPI Data Register (optional).
5. Go to step 1 if more data is to be transmitted.

# LPC1769 SPI : Exception Conditions

Can occur during operation, check by read Status Register

- Read Overrun:
  - New data is lost because the received data have not been read, i.e. read data is full
- Write Collision:
  - Write data to data register during transfer in progress. The write data will be lost
- Mode Fault
  - SSEL of the Master goes active. Master will change to be Slave
- Slave aboard
  - SSEL goes inactive before the transfer is complete. Data will be lost.

# LPC1769 SPI : Default Set up

- Control Register
  - Use 8 bit data transfer
  - Use Clock idles high
  - Sampling at first falling clock edge (mode 1)
  - Start in Slave mode
  - Transfer MSB first
  - Interrupt disable
- Clock Counter Register
  - NOT use (use in Master mode only); set to x00

# LPC1769 SPI : How to Set up

Step for setting up SPI0 (both Master and Slave)

1. Turn on power control for SPI interface (PCONP)
2. Set SPI Clock speed
    1. Set PCLK_SPI in PLCKSEL0
    2. Set divider number in S0SPCCR (even number ≥ 8)
3. Set I/O pin for CLK, SSEL, MISO and MOSI
4. If we want to use interrupt from SPI interface
    1. Set SPI interface to generate interrupt (be an **interrupt source** ) by setting SPIE bit of S0SPCR = 1
    2. Connect **interrupt source** (SPI interrupt) to **interrupt controller** (NVIC #13) by setting ISE_SPI = 1

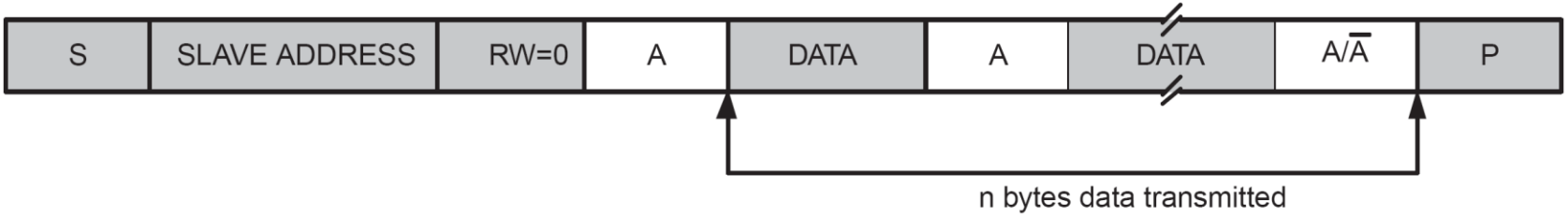
# LPC1769 I2C

- There are 3 I2C ports
  - I2C 0 :
    - supports standard (100kHz), fast mode(400kHz) and fast mode plus (1MHz)
    - Use specialized I2C I/O pad with analog spike suppress filter
  - I2C 1/2 :
    - supports standard and fast mode
    - Use standard I/O pad
  - Each recognizes up to 4 distinct addresses
  - All interface blocks are the identical and include digital filter.

# LPC1769 I2C: operation modes

- In a given application it may operate as a master, slave, or both.
- It can be in one of four modes
  - Master Transmit
  - Master Receive
  - Slave Transmit
  - Slave Receive

# LPC1769 I2C : Master Transmit




from Master to Slave

from Slave to Master

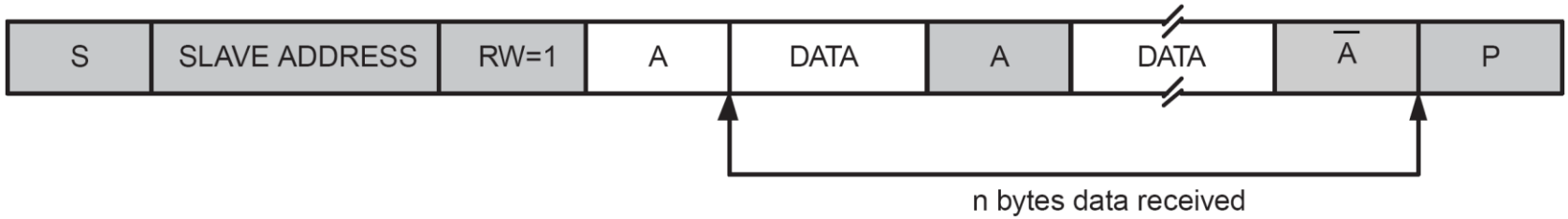
A = Acknowledge (SDA low)

$\overline{A}$ = Not acknowledge (SDA high)

S = START condition

P = STOP condition

# LPC1769 I2C : Master Receive




from Master to Slave

from Slave to Master

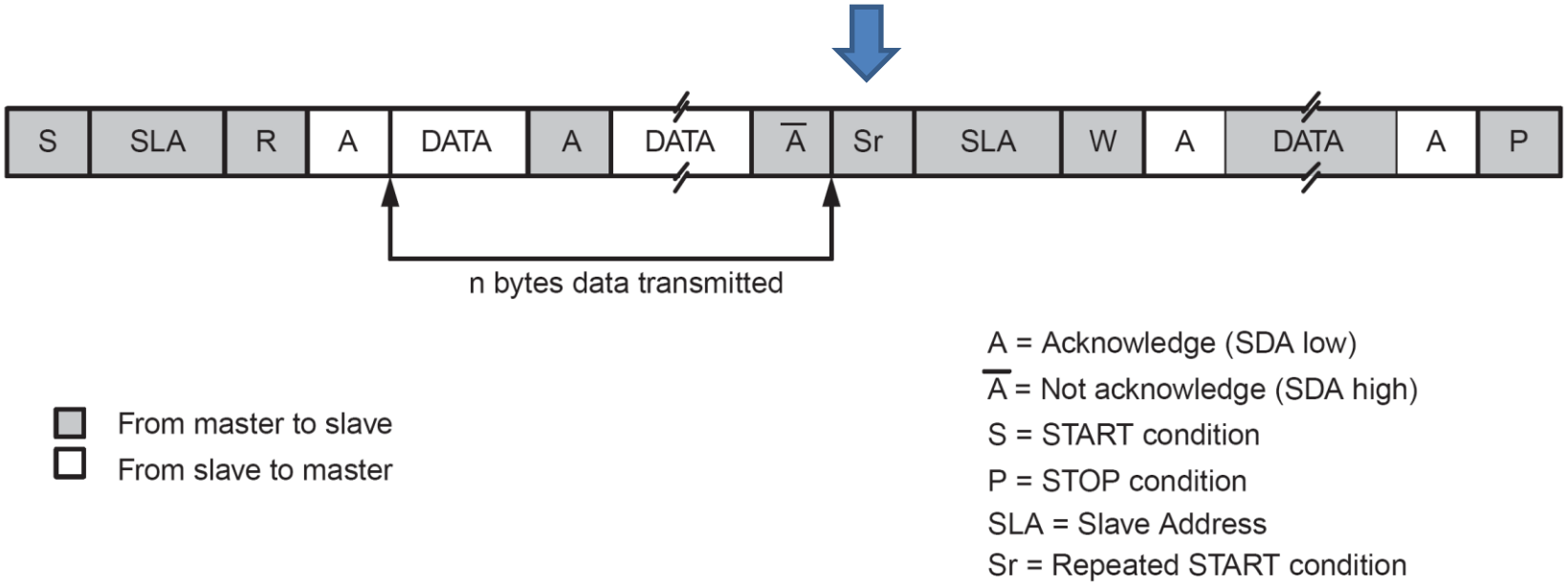
A = Acknowledge (SDA low)

$\overline{A}$ = Not acknowledge (SDA high)

S = START condition

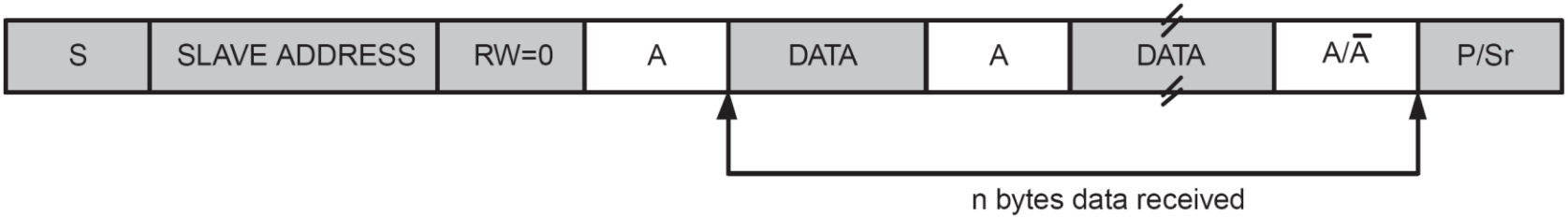
P = STOP condition

# LPC1769 I2C : Master Receive




- When Master Receive follows by Master Transmit to the slave, it does not need to send Stop Condition but set Repeat Start (Sr) instead

# LPC1769 I2C : Slave Receive




from Master to Slave

from Slave to Master

A = Acknowledge (SDA low)

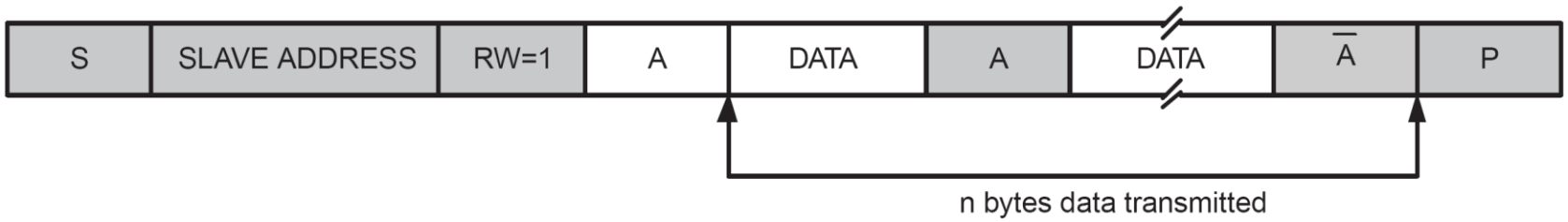
$\overline{A}$ = Not acknowledge (SDA high)

S = START condition

P = STOP condition

Sr = Repeated START condition

# LPC1769 I2C : Slave Transmit




from Master to Slave

from Slave to Master

A = Acknowledge (SDA low)

$\overline{A}$ = Not acknowledge (SDA high)

S = START condition

P = STOP condition

# LPC1769 I2C: Interrupt of I2C

- I2C generate interrupt by setting **SI bit** when I2C state changes
  - In Master mode
    - After Start Condition is transmitted
    - When Acknowledge bit (SDA=L) has been received
    - After sending Not Acknowledge bit (SDA=H)
  - In Slave mode
    - After the address and direction bit have been received
- **SI bit** is the **I2C interrupt Flag**. It is bit[3] of **I2C Control Set Register** (I2xCONSET[3])
  - While it set, the low period of SCL is stretch and the transfer is suspended
  - After it is set, an interrupt is requested and **I2C Status Register** (I2STAT) is loaded with the appropriate state code (one of 26 codes) of a state service routine to be executed
- Clear by software by writing a 1 to **SIC bit** (I2xCONCLR[3])

# LPC1769 I2C : Clock Rate I2C

$$I^2C_{bitfrequency} = \frac{PCLKI2C}{I2CSCLH + I2CSCLL}$$

- I2SCLH defines the number of PCLK_I2C cycles for the SCL HIGH time
- I2SCLL defines the number of PCLK_I2C cycles for the SCL low time
- Each register value must be greater than or equal to 4

Ex PCLK_I2C = 10 MHz,
I2CSCLH + I2CSCLL = 10,
I2C_bit_freq=1 MHz
which is the Fast Mode Plus
(See LPC user's manual)

# LPC 1769 I2C : Basic Configuration

- Set power : PCONP
- Set clock speed of PCLK for use with I2C block
  - For I2C 0, in PCLKSEL0 select PCLK_I2C0
  - For I2C 1/2, in PCLKSEL1 select PCLK_I2C1/PCLK_I2C2
- Select Pin : PINSEL, PINMODE, PINMODE_OD (open drain)
- Interrupt are enable in NVIC: ISE_I2Cx (#10,11,12)
- Initialization
  - Set address
  - Enable interrupt
  - Set serial clock frequency

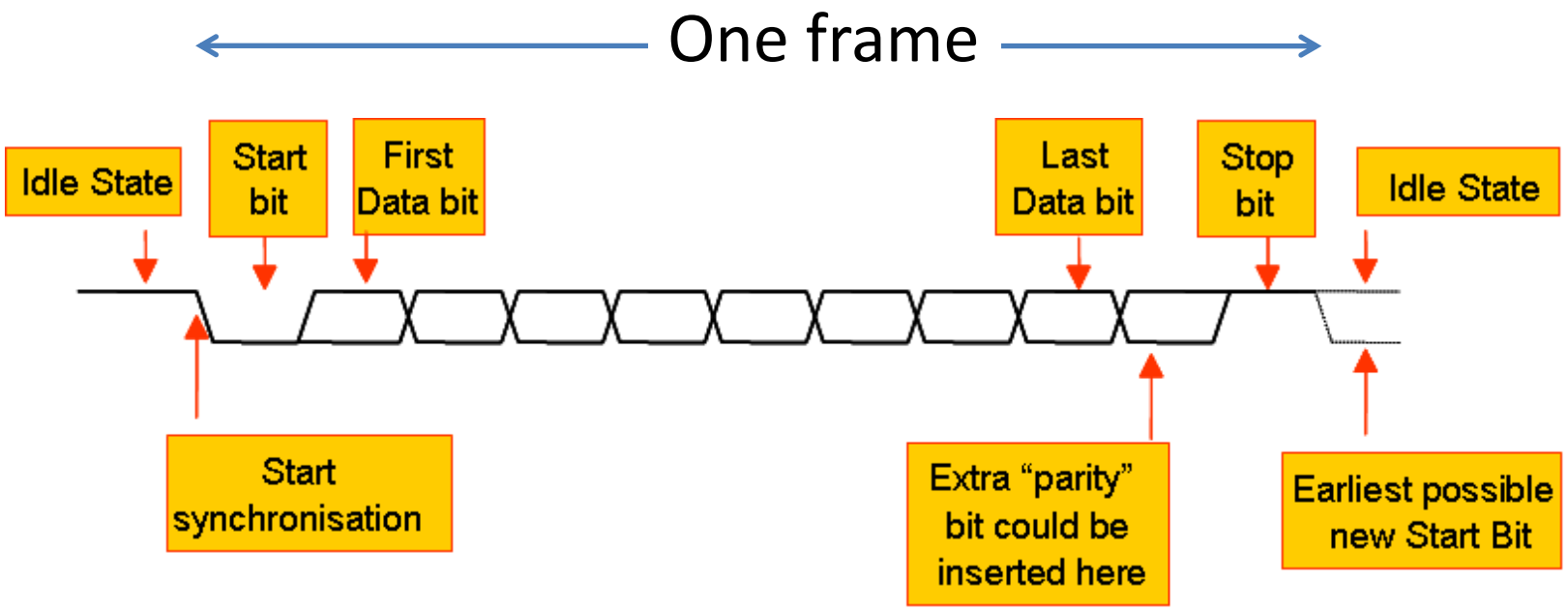
# Outline

- Serial vs Parallel
- Synchronous/Asynchronous
- Commonly used Serial: SPI, I2C, UART, USB, CAN
  - SPI
  - I2C
  - UART

# Asynchronous Serial Data

- Do not require a clock signal to be sent with the data
- Receiver extracts all timing information directly from the signal itself.
  - Transmitter (Tx) and receiver (Rx) nodes somewhat more complex than comparable synchronous nodes
- Data rate is predetermined
  - Transmitter and receiver are preset to recognize the same data rate
- Transmit and receive data in a frame of a byte or a word with a Start and Stop bit.

# UART




- Universal Asynchronous Receiver/Transmitter
  - Idle state = Logic H
  - Frame = Start bit(L) + Data 5–8 Bits + Parity + Stop bit (H)
  - LSB first

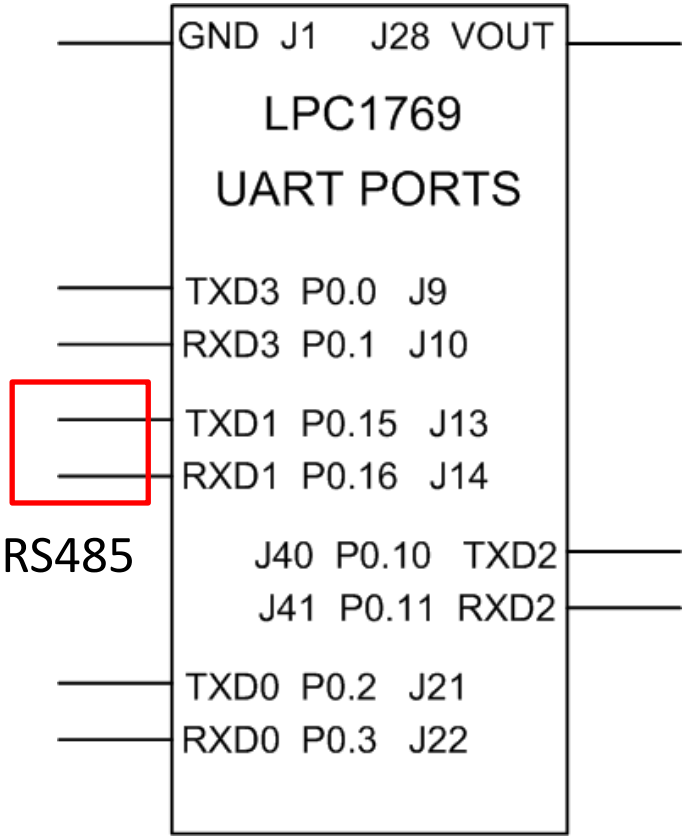
# UART

- Receiver Operation
  - Usually its operation controlled by a clock running at multiples of the data rate such as 8x, 16x oversample clock
  - It operate as
    - The receiver check the state of the incoming signal on each clock pulse, looking for the beginning of the start bit.
    - After start bit is found (usually last for a half-bit time), it waits for another one bit time then begins sampling and shift the result into a shift register.
    - Once completes a frame (5-8 bit data) , converts to parallel and then generates a flag or an interrupt to request CPU to take the received data

# UART

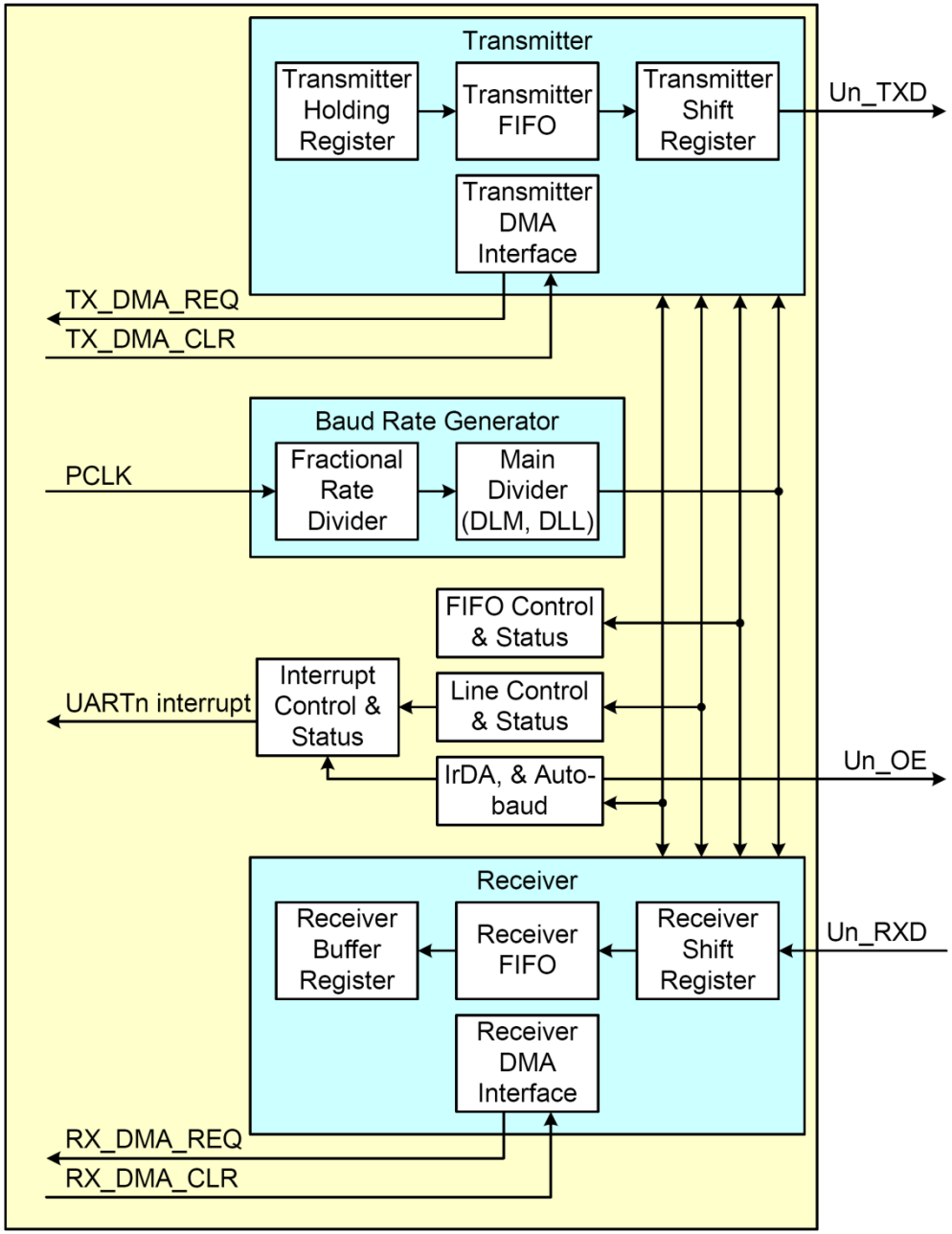
- Transmitter Operation
  - Place data to the shift register
  - Generates a start bit, shifts the required number of data bits out to the line
  - Generates and appends the parity bit (if used), and appends the stop bits.
  - Shows busy status to CPU during transmission
  - Generate interrupt when done and ready for a new data

# UART of LPC1769




RS485

Four serial ports



# UART Parameters

- Parity Bit is an additional bit append to the data to tell how many bit “1” totally in the data
  - Part of simple error detecting code
- Two Types
  - Odd parity : # of “1” + parity bit = odd number
  - Even parity: # of “1” + parity bit = even number

# Parity Check

| Parity Type | Data | Compute parity | Transmit: Data + parity | Receive : Data +parity | Compute parity | Result |
|---|---|---|---|---|---|---|
| Even | 1001 | 1+0+0+1 (mod 2) =0 | 10010 | 10010 | 1+0+0+1+0 (mod2) = 0 | correct |
| Odd | 1001 | 1+0+0+1 +1 (mod 2) =1 | 10011 | 10011 | 1+0+0+1 +1 (mod 2) =1 | correct |
| Even | 1001 | 1+0+0+1 (mod 2) =0 | 10010 | 11010 | 1+1+0+1+0 (mod 2) = 1 | error detected |
| Odd | 1001 | 1+0+0+1 +1 (mod 2) =1 | 10011 | 11001 | 1+1+0+1 +1 (mod 2) =0 | error detected |
| Even | 1001 | 1+0+0+1 (mod 2) =0 | 10010 | 11110 | 1+1+1+1+0 (mod2) = 0 | fail to catch |
| Odd | 1001 | 1+0+0+1 +1 (mod 2) =1 | 10011 | 11111 | 1+1+1+1 +1 (mod 2) =1 | fail to catch |

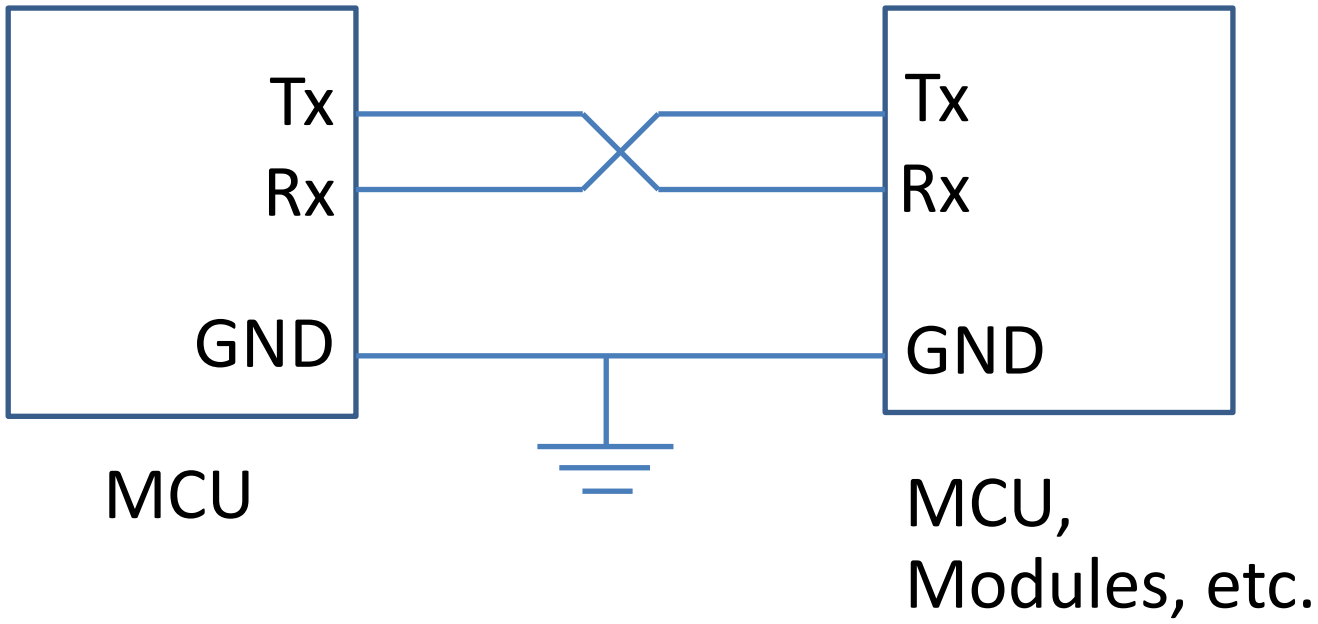
# UART Parameters

- A parity bit is only guaranteed to detect an odd number of bit errors
  - Failing to catch even number of  bit errors)
- Baud rate is total number of bits per unit time
  - Baud rate = 1 / bit-time
- Bandwidth is data per unit time
  - Bandwidth = (data-bits / frame-bits) * baud rate
- LPC1769 Baud rate

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UARTn_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

# UART Physical Connection

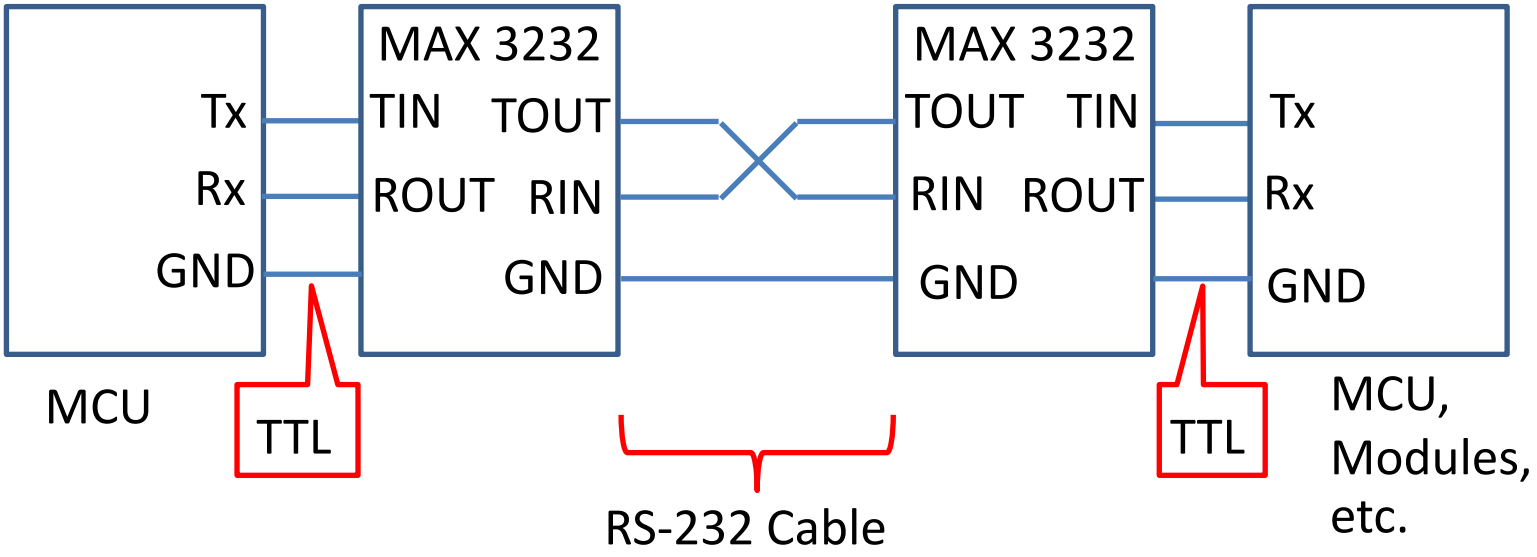
- Directly cross connection with 2 wires and shared GND
- For use in short distance



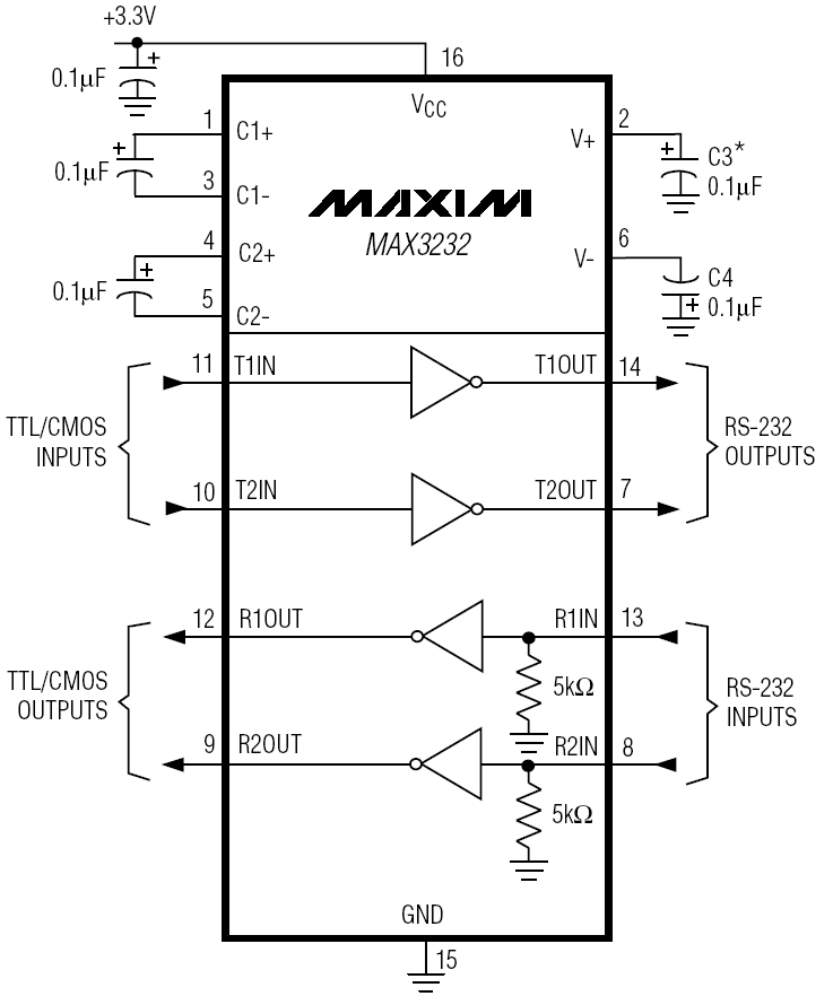

- Logic level : $V_H$=3.3V, $V_L$=0V or TTL level $V_H$=5V, $V_L$=0V

# UART Physical Connection

- UART using RS-232 Standard
  - RS-232 Standard defines different voltage levels of logical H and L to support a long distance link and provide a better noise immunity.
  - Need a device to convert such as IC MAX 3232

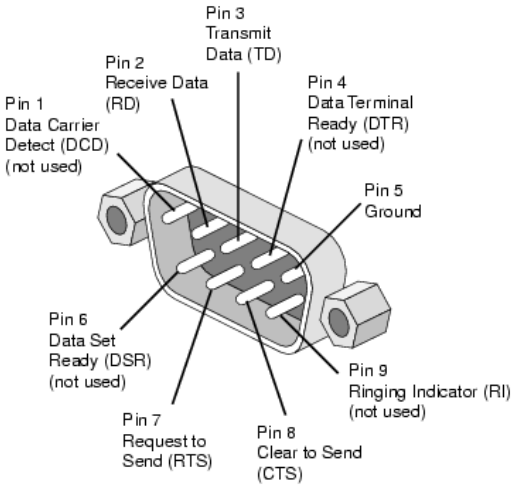


# TTL - RS232 Transceiver




* C3 CAN BE RETURNED TO EITHER $V_{CC}$ OR GROUND.

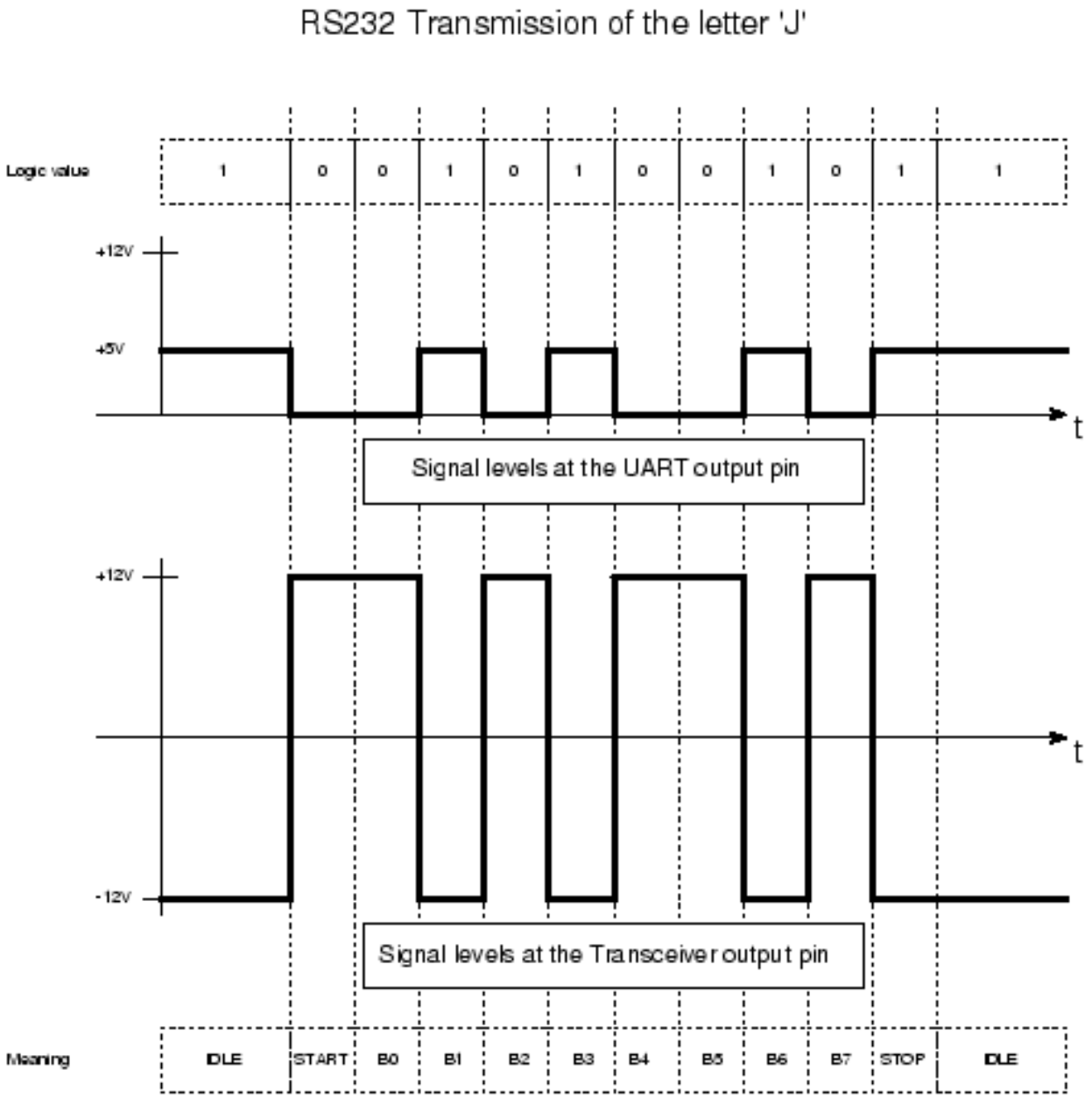
250 kbps – 1Mbps

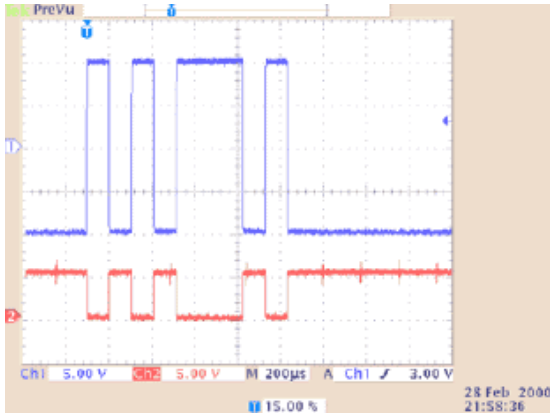




Cable and DB-9Connector

# RS-232 Signal Level



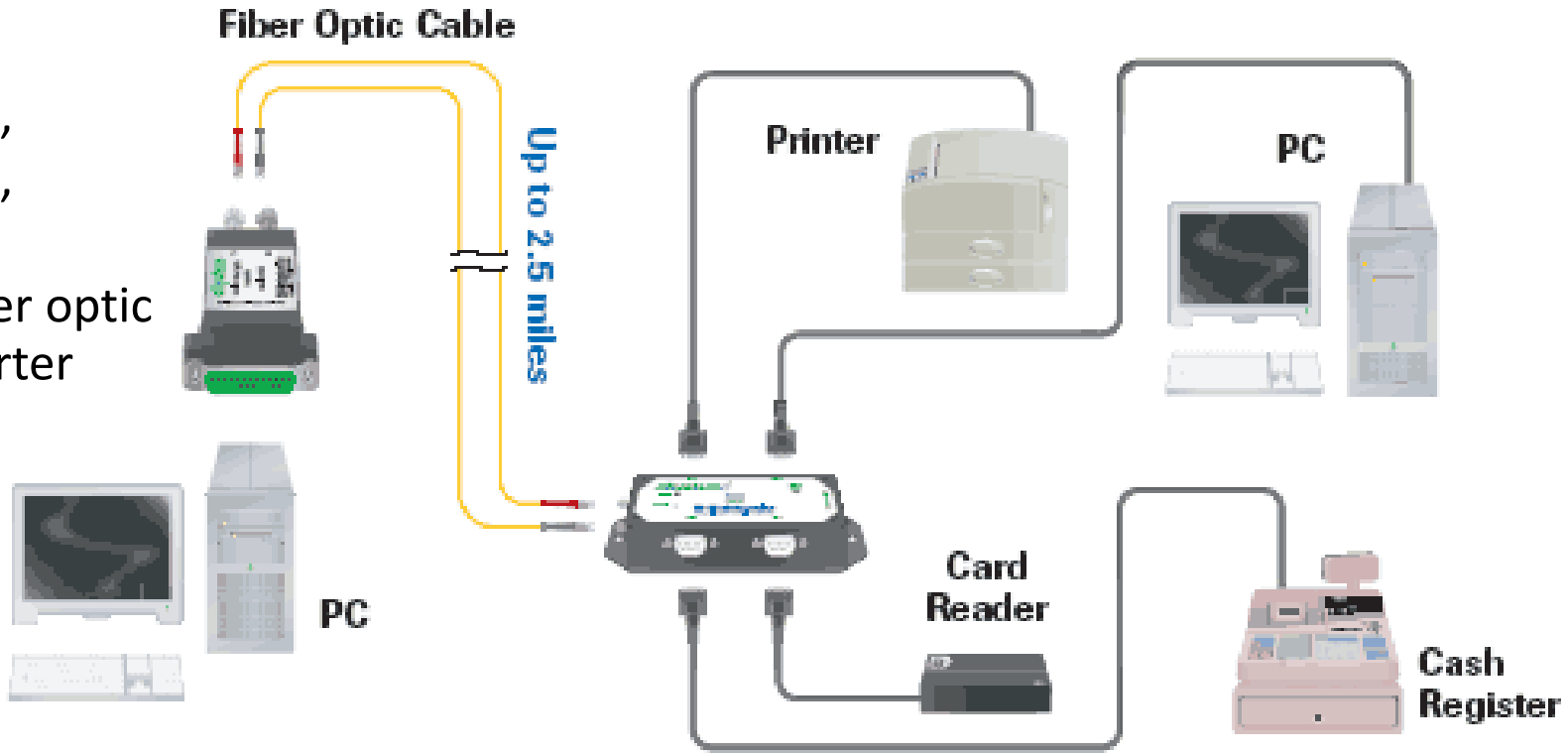



# UART Physical Connection

- Connecting via other means that not use electrical wire
  - Light such as fiber optic using  serial to fiber converter
  - Infrared beam through the air such as infrared port of cell phone, printer
  - Radio Frequency (RF) such as serial port profile (SPP) of Bluetooth
- Usually requires  a conversion  from UART signal into other IEA serial link standards  such as RS232, RS422, RS485, then uses adapter  to convert  to light or RF signals

# Example: Serial link with fiber optic

RS232,
RS422,
RS485
to Fiber optic converter




RS232 to Fiber optic converter

# UART: Applications

- UART is a standard port in MCU to communicate to another MCU, peripheral modules such as sensor, GPS module, Zigbee module, wi-fi modules,  etc.
- Used to be a standard serial port of PC with RS-232 (now most PC use USB instead)
  - Connect to peripheral such as printer, modem, cash register, card reader, etc.
  - Today, to connect PC to UART devices uses a USB-to-UART converter
- Infrared link
- Serial Port Profile (SPP) of Bluetooth