Sec 5. Serial Communication & Interfaces

Connecting an Embedded Computer to Other Devices

1

Concepts of Serial Communication

- Limitations of Parallel Bus
  - Clock skew becomes a serious issue for high speed and long distance
  - Cost of wire, fewer wires cost less and occupies less space
  - Cross talk between multiple conductors, affecting signal quality
- Serial Communication
  - Sending data a bit at a time, sequentially, over a communication channel
  - Un-clocked, no clock skew problem
  - Fewer wires → low cost, more space for better isolation from surroundings
- Simplex, half-duplex, and full-duplex
  - Simplex: data can be transferred in only one direction
  - Half-duplex: data can be transferred in two directions, but one at a time
  - Full-duplex: data can be transferred in two directions concurrently

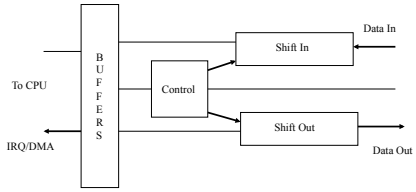
2

General Procedure and Logic of Serial Communication

- Special signal marks
  - Start bit and stop bit that mark the begin and end of one comm.
- Parity bit
  - Error checking and correction
    - Even parity, odd parity, CRC, Hamming code, etc.
- Baud rate
  - Specifies how fast bit sequence is transmitted
- Buffering for transmitter and receiver

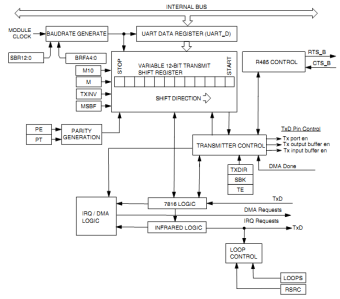
3

Logic Diagram of Most Serial Interfaces



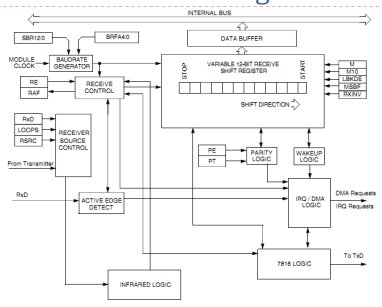
4

UART Transmitter Block Diagram



5

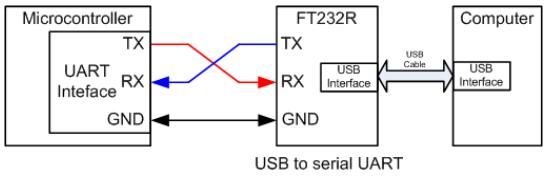
UART Receiver Block Diagram



6

Connecting to PC

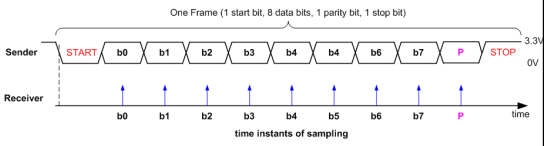
- FT232R converts the UART port to a standard USB interface



USB to serial UART

7

Data Frame



Tolerate 10% clock shift during transmission

- Sender and receiver uses the same transmission speed
- Data frame
  - One start bit
  - Data (LSB first or MSB, and size of 7, 8, 9 bits)
  - Optional parity bit
  - One or two stop bit

8

Baud Rate

- Historically used in telecommunication to represent the number of pulses physically transferred per second
- In digital communication, baud rate is the number of bits physically transferred per second
- Example:
  - Baud rate is 9600
  - each frame: a start bit, 8 data bits, a stop bit, and no parity bit.
  - Transmission rate of actual data

    ~~9600/8 = 1200 bytes/second~~

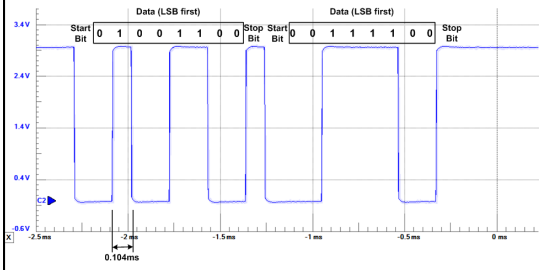
    9600/(1 + 8 + 1) = 960 bytes/second
  - The start and stop bits are the protocol overhead

9

Error Detection

- Even Parity: total number of "1" bits in data and parity is even
- Odd Parity: total number of "1" bits in data and parity is odd
- Example: Data = 10101011 (five "1" bits)
  - The parity bit should be 0 for odd parity and 1 for even parity
- This can detect single-bit data corruption

Transmitting 0x32 and 0x3C



1 start bit, 1 stop bit, 8 data bits, no parity, baud rate = 9600

11

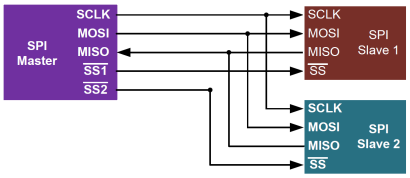
Voltage Levels

Standard	Voltage signal	Max distance	Max speed	Number of devices supported per port
RS-232	Single end (logic 1: +5 to +15V, logic 0: -5 to -15V)	100 feet	115Kbit/s	1 master, 1 receiver
RS-422	Differential (-6V to +6V)	4000 feet	10Mbit/s	1 master, 10 receivers
RS-485	Differential (-7V to +12V)	4000 feet	10Mbit/s	32 masters, 32 receivers

12

Serial Peripheral Interface (SPI)

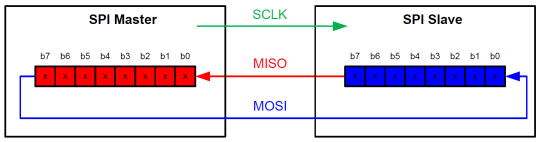
- Synchronous full-duplex communication
- Can have multiple slave devices
- No flow control or acknowledgment
- Slave cannot communicate with slave directly.



SCLK: serial clock
SS: slave select (active low)
MOSI: master out slave in
MISO: master in slave out

13

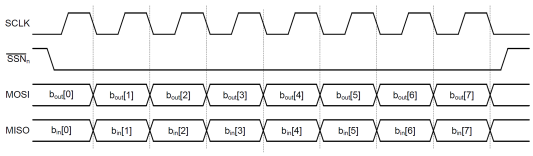
Data Exchange



- Master has to provide clock to slave
- Synchronous exchange: for each clock pulse, a bit is shifted out and another bit is shifted in at the same time. This process stops when all bits are swapped.
- Only master can start the data transfer

14

Clock



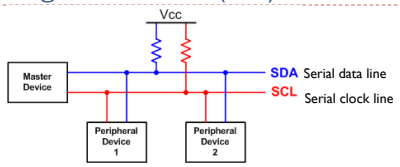
15

Inter-Integrated Circuit (I^2C)

- Designed for low-cost, medium data rate applications by Philips in the early 1980's
  - Original purpose: connect a CPU to peripheral chips in a TV-set
  - Today: a de-facto standard for 2-wire communications
  - Since October 10, 2006, no licensing fees are required to implement the I^2C protocol. However, fees are still required to obtain I^2C slave addresses allocated by NXP (acquired Philips).
- Characteristics
  - Serial, byte-oriented
  - Multi-master, multi-slave
  - Two bidirectional open-drain lines, plus ground
    - Serial Data Line (SDA)
    - Serial Clock Line (SCL)
    - SDA and SCL need to pull up with resistors

16

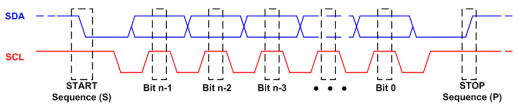
Inter-Integrated Circuit (I2C)



- Up to 100 kbit/s in the standard mode, up to 400 kbit/s in the fast mode, and up to 3.4 Mbit/s in the high-speed mode.
- SDA and SCL have to be open-drain
  - Connected to positive if the output is 1
  - In high impedance state if the output is 0
- Each Device has an unique address (7, 10 or 16 bits). Address 0 used for broadcast
- External pull-up (4.7 kΩ for low speed, 3 kΩ for standard mode, and 1 kΩ for fast mode).

17

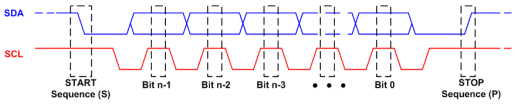
Timing Diagram



- A **START** condition is a high-to-low transition on SDA when SCL is high.
- A **STOP** condition is a low to high transition on SDA when SCL is high.
- The address and the data bytes are sent most significant bit first.
- Master generates the clock signal and sends it to the slave during data transfer

18

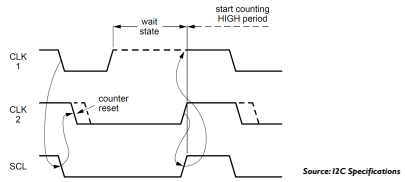
Multiple Masters



- "Wired-AND" bus: A sender can pull the lines to low, even if other senders are trying to drive the lines to high
- In single master systems, arbitration is not needed.
- Arbitration for multiple masters:
  - During data transfer, the master constantly checks whether the SDA voltage level matches what it has sent.
  - When two masters generate a START setting concurrently, the first master which detects SDA low while it has actually intended to set SDA high will lose the arbitration and let the other master complete the data transfer.

19

Clock Synchronization



- Clock synchronization is needed when there are multiple masters.
- **Wired-AND** connection for clock synchronization
  - Each master has a counter. Counter resets if SCL goes LOW. When the counter counts down to zero, the master releases SCL and thus SCL goes high.
  - SCL remains LOW if any master pulls it LOW.
  - When all masters concerned have counted off their LOW period, the clock line is released and goes HIGH.
  - After going high, all masters start counting their HIGH periods. The first master to complete its HIGH period pulls the SCL line LOW again.

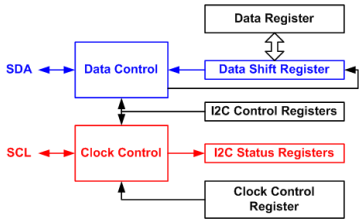
20

Working Modes

- Master-sender}
  - Master issues START and ADDRESS, and then transmits data to the addressed slave device
- Master-receiver}
  - Master issues START and ADDRESS, and receives data from the addressed slave device
- Slave-sender}
  - Master issues START and the ADDRESS of the slave, and then the slave sends data to the master
- Slave-receiver}
  - Master issues START and the ADDRESS of the slave, and then the slave receives data from the master.

21

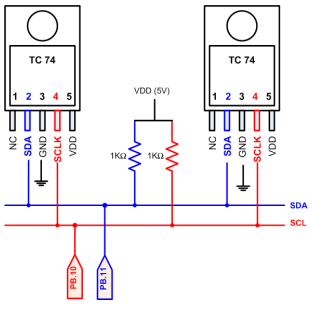
STM32L I2C Module



- During sending, the I²C hardware automatically sets the *TxE* flag in the status register if an acknowledge pulse is received from the slave.
- During receiving, the I²C hardware then automatically sets the *RxNE* flag in the status register if a byte has been successfully received.

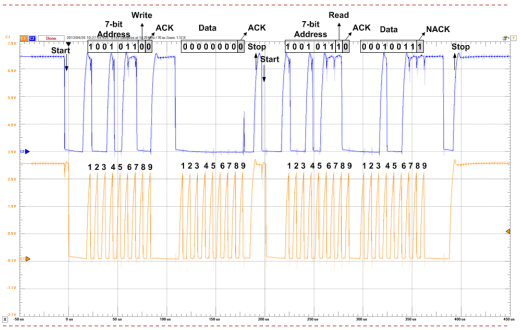
22

Interfacing Serial Digital Thermal Sensor



23

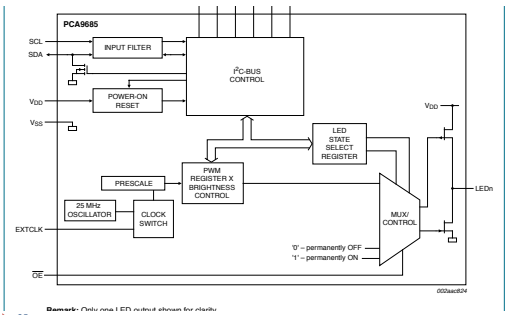
I2C Data



24

PCA9685

16-channel, 12-bit PWM Fm+ I2C-bus LED controller



25

- The PCA9685 is an I2C-bus controlled 16-channel LED controller optimized for LCD Red/Green/Blue/Amber (RGBA) color backlighting applications. Each LED output has its own 12-bit resolution (4096 steps) fixed frequency individual PWM controller that operates at a programmable frequency from a typical of 40 Hz to 1000 Hz with a duty cycle that is adjustable from 0 % to 100 % to allow the LED to be set to a specific brightness value. All outputs are set to the same PWM frequency.
- Each LED output can be off or on (no PWM control), or set at its individual PWM controller value. The LED output driver is programmed to be either open-drain with a 25 mA current sink capability at 5 V or totem pole with a 25 mA sink, 10 mA source capability at 5 V. The PCA9685 operates with a supply voltage range of 2.3 V to 5.5 V and the inputs and outputs are 5.5 V tolerant. LEDs can be directly connected to the LED output (up to
- 25 mA, 5.5 V) or controlled with external drivers and a minimum amount of discrete components for larger current or higher voltage LEDs.
- The PCA9685 is in the new Fast-mode Plus (Fm+) family. Fm+ devices offer higher frequency (up to 1 MHz) and more densely populated bus operation (up to 4000 pF).

26

Error Control Coding

Define binary addition:
0 + 0 = 0;
0 + 1 = 1;
1 + 0 = 1;
1 + 1 = 0.
Exclusive-OR

Define Vector addition:
(0101) + (1100) = (1001)
Bit-wise Exclusive-OR

N-bit Binary Vectors:
N-tuple of 0's and 1's
Also referred to as N-bit word
i.e. 4-bit vector or 4-bit word:
(0100), (1001), & (1101)

Scalar Multiplication:
Two scalars: 0 and 1:
1*(0110) = (0110)
0*(0110) = (0000)

27

Error Control Coding

Goal: detect "errors" (e.g., flipped bits) in transmitted segment

Block code
a set of code words of fixed length n, with each code word being an n-tuple over a finite field: ***S, V*** *= all n-tuples*

Linear code
If ***S*** *forms a subspace of* ***V****, then* ***S*** *is called linear code*

codeword: a word in ***S*** is called codeword and otherwise noncodeword

Hamming* weight (*w*)**: # of nonzero components of ***X *= (x1, x2, ... xn)*

Hamming distance (d): # of positions in which the two words differ

Minimum Distance:

- the minimum of the distances between all pairs of code ***C***, it is also the distance of the code.

examples:
x=(10011), y=(01010),
w(x)=3, w(y)=2, d(x,y)=3
C= (001, 010, 100,110, 101, 011)
distance of ***C*** is 1

28

Coding Theory

Theorem 1:

It is necessary and sufficient that the distance of a code is at least d in order to detect any error pattern of weight d-1 or less

How do we design a code that has distance *d*?

A linear code can be represented by a matrix G, or H:

$$G = \begin{pmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ \cdots & & & \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{pmatrix}$$

Where n is the code length, k is number of information bits, and n-k is number of parity bits

29

Representation of Linear Code

Matrix G is called generating matrix

Matrix H, defined as a null space of G, is called parity matrix

G or H uniquely defines a linear code

Code words of the linear code are generated by multiplying all possible information words to G

If $G = \begin{pmatrix} I_k & P_{n-k} \end{pmatrix}$

then $H = \begin{pmatrix} P^T & I_{n-k} \end{pmatrix}$

codewords =

$$u * \begin{pmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ \cdots & & & \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{pmatrix}$$

For u = 0……00, 0……01,…,1……1

30

Two Orthogonal Matrices

$$G = \begin{pmatrix} 1\,0\,\ldots\ldots 0\,p_{1,1}\;p_{1,2}\;\ldots\ldots\;p_{1,n-k} \\ 0\,1\,\ldots\ldots 0\,p_{2,1}\;p_{2,2}\;\ldots\ldots\;p_{2,n-k} \\ \ldots\ldots \\ 0\,0\,\ldots\ldots 1\,p_{k,1}\;p_{k,2}\;\ldots\ldots\;p_{k,n-k} \end{pmatrix}$$

$$H = \begin{pmatrix} p_{1,1}\;p_{2,1}\;\ldots\ldots\;p_{k,1}\;1\,0\,\ldots\ldots 0 \\ p_{1,2}\;p_{2,2}\;\ldots\ldots\;p_{k,2}\;0\,1\,\ldots\ldots 0 \\ \ldots\ldots \\ p_{1,n-k}\;p_{2,n-k}\;\ldots\ldots\;p_{k,n-k}\;0\,0\,\ldots\ldots 1 \end{pmatrix}$$

For any i and j, we have

$$(0\,0\ldots1\ldots\ldots0\;p_{i,1}\;p_{i,2}\;\ldots\ldots\;p_{i,n-k}) *$$

$$*\;(p_{1,j}\;p_{2,j}\;\ldots\ldots\;p_{k,j}\;0\,0\ldots1\ldots\ldots0)^T$$

$$= p_{i,j} + p_{i,j} = \underline{\boldsymbol{0}}$$

31

Relationship Between G and H

$$G = \begin{pmatrix} 1\,0\,1\,1\,0 \\ 1\,1\,0\,1\,1 \\ 0\,1\,0\,1\,0 \end{pmatrix}$$ Through canonical reduction $$G = \begin{pmatrix} 1\,0\,0\,0\,1 \\ 0\,1\,0\,1\,0 \\ 0\,0\,1\,1\,1 \end{pmatrix}$$

After canonical reduction, the resulting G matrix:
First row = (11011)+(01010);
Second Row =the original third row;
Third row=(10110)+11011)+(01010).

$$H = \begin{pmatrix} P^T\;I_{n-k} \end{pmatrix}$$

$$= \begin{pmatrix} 0\,1\,1\,1\,0 \\ 1\,0\,1\,0\,1 \end{pmatrix}$$

$$U * H^T = \underline{0}$$

32

Distance and Error Control

How do we relate distance of a code to G?

Theorem 2: For any codeword $\underline{u}$ of weight d in a linear code $\underline{\boldsymbol{C}}$, d columns of its H matrix are linear dependent

Theorem 3: A linear code $\underline{\boldsymbol{C}}$ has distance at least d iff every $d-1$ or fewer columns of its H matrix are linearly independent

recall that distance of a code is also the minimum weight.

33

An Example

Design a 5-bit linear code that can detect 2 bits errors

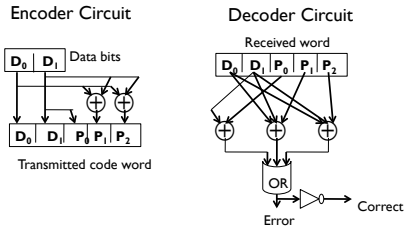
- come up with a H matrix with distance 3
- derive G from H
- generate all codewords
- encoder and decoder circuit.

$$H = \begin{pmatrix} 0\,0\,0\,1\,1 \\ 0\,1\,1\,0\,0 \\ 1\,0\,1\,0\,1 \end{pmatrix}$$ Through canonical reduction $$H = \begin{pmatrix} 0\,1\,1\,0\,0 \\ 1\,1\,0\,1\,0 \\ 1\,1\,0\,0\,1 \end{pmatrix}$$ $$G = \begin{pmatrix} 1\,0\,0\,1\,1 \\ 0\,1\,1\,1\,1 \end{pmatrix}$$

34

Encoder and Decoder Circuits Design

With this generating matrix, we can design simple encoder and decoder circuits as shown here.



35