**UART – Universal Asynchronous Receiver Transmitter**

* 5 to 9 data bits
* 1 start bit
* 0 to 1 parity bit
* 1 or 2 stop bits
* Asynchronous because communicating devices don’t share common clock
* Start bit used to let the device know that communication is about to start so that receiving device can synchronise its clock with the incoming data to receive it properly.
* Standard baud rates include: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
* 1 to 1 communication
* Least significant bit gets transferred first
* Requires 1 line for transmitting and one for receiving.
* Consequences of baud rate mismatch

<https://learn.sparkfun.com/tutorials/serial-communication/all>

<http://www.circuitbasics.com/basics-uart-communication/>

**SPI – Serial Peripheral Interface**

* Supports transmission of 8 or 16-bit data
* For operating modes: CPHA = CPOL = 0, CPHA = CPOL = 1, CPHA = 0 & CPOL = 1, CPHA = 1 & CPOL = 0.
* 4 Lines: CLK, MISO, MOSI and CS
* Data sent MSB first
* Clock signal is sent along with data
* CS lowered first before data is sent
* At high frequencies should not be used over long distances otherwise will give errors

<https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all>

**IMU**

* Accelerometer, gyroscope and magnetometer theory
* Getting angles from the three theory
* Gyroscope noise, gimbal lock
* sensor accel, gyro and magnetometer offsets and noise and cross-axis sensitivity

**Complementary Filter**

* Complementary filter equation
* Theory

<https://www.pieter-jan.com/node/11>

<https://www.researchgate.net/publication/308850497_MEMS_based_IMU_for_tilting_measurement_Comparison_of_complementary_and_kalman_filter_based_data_fusion/download>

**Direct Memory Access**

* Theory and benefits

**Bluetooth Wireless**

**FPGA**

* Synchronisation is required so that metastability does not occur
* Controller Datapath architecture

**Euler Angles**

* Gimbal lock
* Theory

<https://en.wikipedia.org/wiki/Euler_angles>

<https://en.wikipedia.org/wiki/Gimbal_lock>

**UART – Universal Asynchronous Receiver Transmitter**

The UART communication protocol will be used to write data to the to the Bluetooth modules for wireless transmission. The protocol operates on 1 to 1 basis, meaning that the transaction can only happen between two devices at a time. This protocol is called asynchronous because the two sides of the transmission do not share a common clock and rely on their own clocks to synchronise themselves with the transmission of data in order to receive the information. The benefit of this is that the protocol only requires two lines to transmit data; one for transmitting and one for receiving data. The protocol consists of the following:

**Start Bit** – This bit is transmitted first to communicate with the other device that a transaction is about to begin. This bit is always 0 as the IDLE state of the communication line is of logic 1.

**Data Bits** – The data bits are transmitted after the start bit and the least significant bit of the data is transmitted first. The number of the data bits can range from 5 to 9.

**Parity Bit** – This bit is optional and is used as a low level error checking method. The parity can be odd or even. In order to produce the parity bit the data bits are added and the evenness of the data whether the bit is set or not. If parity is set to even and the number of 1’s in the data was odd, the parity bit would be set to 1. If the number of 1’s was even the parity bit would be set to 0. The opposite would happen if the parity was set to odd; if number of 1’s is odd the parity bit would be 0 and when the number of 1’s was even, the parity would be set to 1.

**Stop bits** – The stop bit is transmitted last and is a transition to the IDLE state which is 1 and therefore the stop bit is always 1. The stop bit can be either one or two bits wide depending on configuration.

The speed of transmission is determined by the baud rate which measures the amount of bits transmitted per second. The standard baud rates in devices are: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200. The communicating devices need to be configured identically in order to avoid data mismatch which will cause the data to be received incorrectly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UART Data Frame** | Start | Data | Parity | Stop |
| **Size** | 1 | 5-9 | 0-1 | 1-2 |

**SPI – Serial Peripheral Interface**

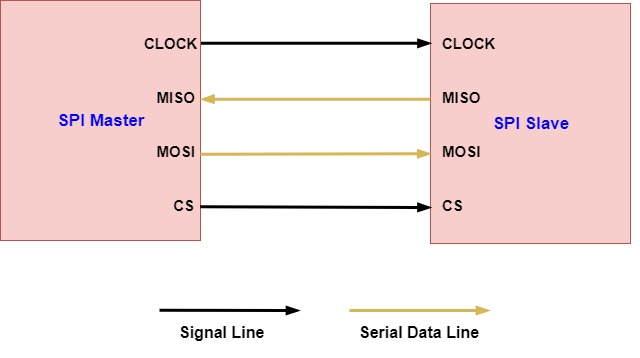
The Serial peripheral interface will be used for communication between internal measurement units and FPGA and also for communication between the FPGA and STM32L432KC. Unlike the UART protocol, SPI provides a synchronous solution to communication by having a separate line for the clock signal that is sent along with the data. The serial peripheral interface also possesses the ability to communicate with more than one slave device by using an additional chip select line for each device connected to the interface. The Serial peripheral interface therefore requires four lines to enable communication between devices. These lines are:

**CLOCK** – The clock signal is sent to slave device from master using this line.

**MOSI** – The Master Out Slave In line is used to send data from the master to the slave.

**MISO** – The Master In Slave Our line is used to send data from the slave device to the master

**CS** – The Chip Select line, also known as Slave Select (SS or SSEL) is used by the master to select the slave device with which the master wishes to communicate with.



Before the data is transmitted the SPI master lowers the chip select line to notify the slave device that a transmission is about to begin. The master device then provides the clock signal to which the data being sent is synchronised. once all the data bits have been sent, the master pulls the chip select high to end the transaction. In contrast to the UART protocol, the data is sent most significant bit first and The data transmitted can only be configured to be 8 or 16 bits wide. There are also four communication modes in SPI which are determined by the clock polarity (CPOL) and the clock phase (CPHA). The clock polarity determines the edge on which the serial peripheral interface transitions to the idle state. The clock phase determines which clock edge is used to place data on the line and which clock edge is used to capture the data on the receiving end.

**CPOL = 0 & CPHA = 0** – The clock starts on the rising edge and goes idle on the falling edge. Data is placed on the line on falling edges and is captured on rising edges of the clock. In this mode, the first bit has to be present on the line before the transmission starts.

**CPOL = 1 & CPHA = 0** – The clock starts on the falling edge and goes idle on the rising edge. Data is placed on the line on rising edges and captured on falling edges of the clock.

**CPOL = 0 & CPHA = 1** – The clock starts on the rising edge and goes idle on the falling edge. Data is placed on the line on rising edges and is captured on falling edges of the clock.

**CPOL = 1 & CPHA = 1** – The clock starts on the falling edge and goes idle on the rising edge. Data is placed on the line on falling edges and is captured on rising edges of the clock.

The communicating devices must be configured identically in order for the transaction to be successful.

The communication protocol is capable of working at much higher frequencies (millions of bits per second) compared to UART. Due to higher frequency of operation the serial peripheral interface is not suited for long range communications as capacitive effects of the wire become significant with increasing frequency and distance causing the signal to distort and being interpreted by the receiving device incorrectly.

