# **Communication Network**

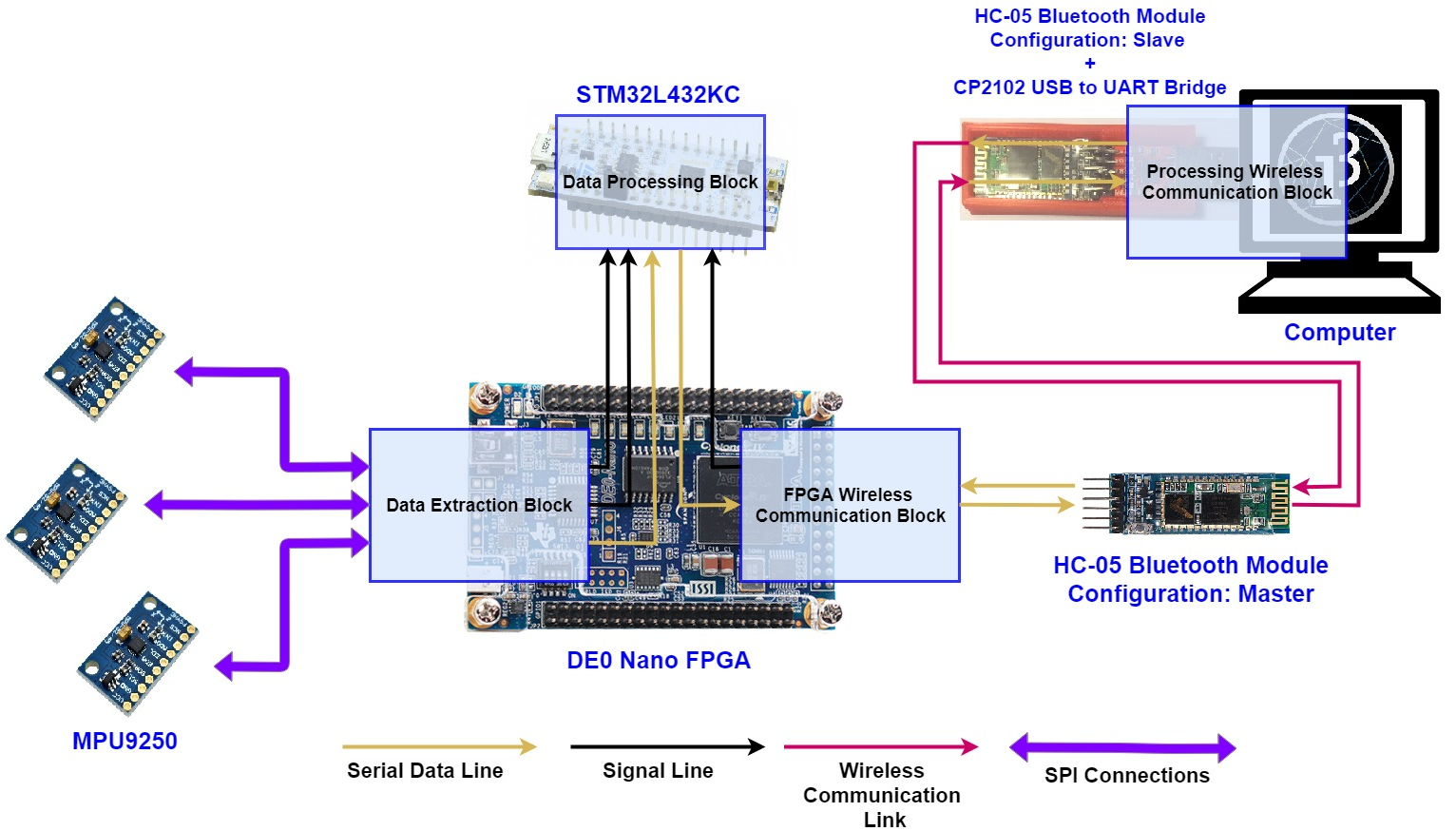
The purpose of this communication network is to provide a means of moving data from three internal measurement units (IMUs), through the system, to the destination which is the processing simulation. This communication network consists of 4 distinct sections; the data extraction block, data processing block, FPGA wireless communication block and the processing communication block. Figure 1 provides an overview of how the data moves across hardware before it arrives in the processing simulation. This part of the design section is concerned with the design of the network and how it enables the data to move before it reaches the processing simulation. The Blocks inside of the FPGA function at a clock speed of 1MHz and each component is synchronised through the same clock. Because each component in the system is connected to this clock, the connections to the clock have not been shown to maintain clarity in the upcoming diagrams.

Figure 1: Diagram depicting the flow of data across hardware.

## **Data Extraction Block**

The ‘Data Extraction Block’, as the name suggests, is responsible for the extraction of data from the internal measurement units. The data is extracted using the serial peripheral interface and then is stored in a buffer where it waits to be transmitted to the ‘Data Processing Block ‘using the serial peripheral interface once again. To achieve parallel data extraction, three IMU extraction components are present, each consisting of the above mentioned serial peripheral interface component, buffer, and also an address loader, data extraction controller and data transmission controller. The address loader is used to load addresses to the SPI such that it can request data from registers inside the MPU9250 which contain the gyroscope and accelerometer data. The two controllers are used to synchronise the extraction and transmission process.

Because the there are three IMU extraction blocks within the data extraction block but there is only one serial peripheral interface component that communicates with the data processing block inside of the STM32L432KC board, the ‘FPGA-STM32L432 SPI Access Controller’ is implemented to control the access such that data can be sent without any conflict. Figures 2 and 3 show all the components that can be found inside the data extraction block and how they are connected.

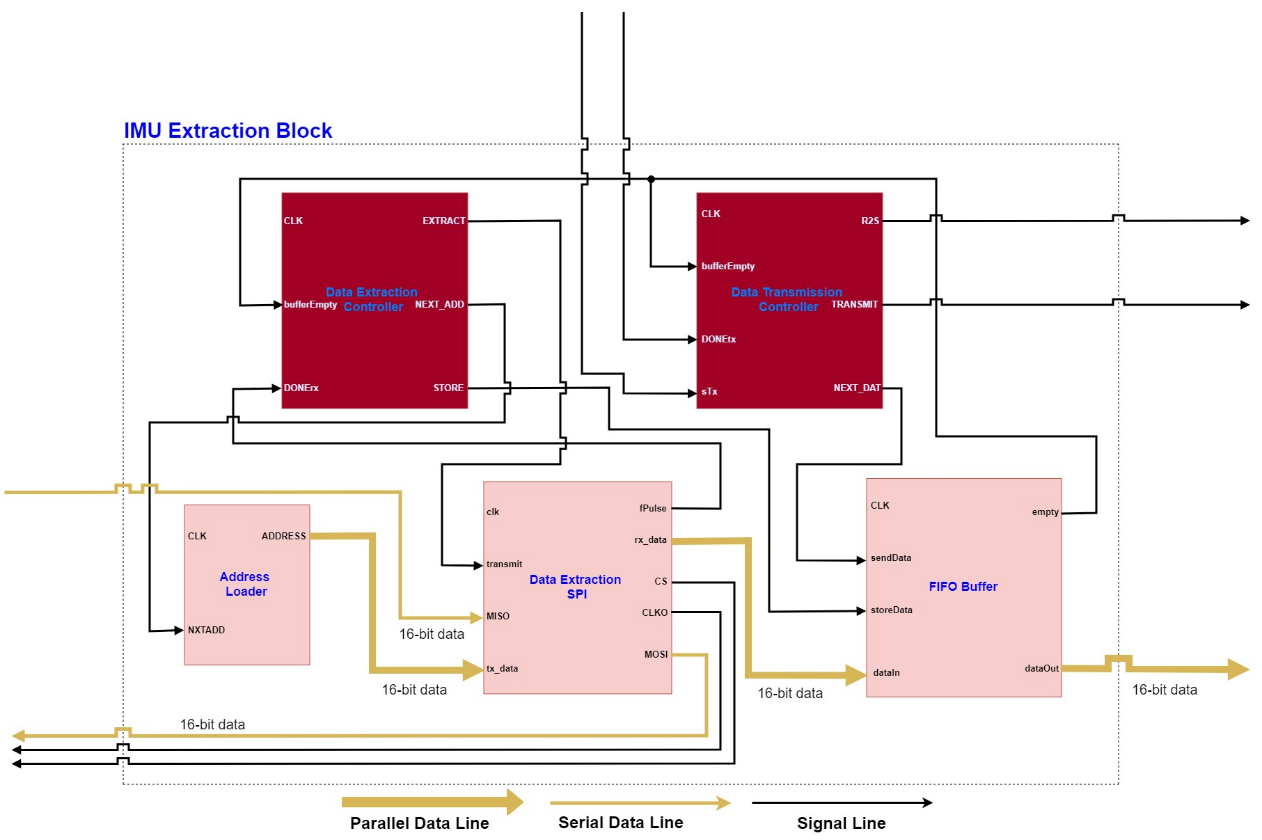
Machine generated alternative text:
Data Extraction 
Block 
CLK 
r2slMU1 
txlMU1 
datalMU1 
16-bit data 
dataOut 
TRANSMIT 
FPGA-STM32 
clk 
MISO 
cs 
CLKO 
FPGA-STM32rx data 
r2slMU2 
SPI 
Access 
txlMU2 
Controller 
datalMU2 
r2slMU3 
txlMU3 
datalMU3 
DONEtx 
IMU Extraction 
strtTx3 
strtTx2 
strtTx1 
R2S 
SPI 
tx data 
transmit 
DONEtx 
MOSI 
fPulse 
R2S 
DONEtx 
16-bit data 
R2S 
Text 
Blockl TRANSMI 
IMIJ Extraction 
Block2 TRANSMIT 
MISO 
IMLJ Extraction 
Block3 TRANSMIT 
dataOut 
16-bit data 
MISO 
16-bit data 
O 
x 
dataOut 
o 
16-bit data 
MISO 
16-bit data 
16-bit data 
Serial Data Line 
O 
x 
dataOut 
o 
16-bit data 
O 
x 
o 
16-bit data 
16-bit data 
Parallel Data Line 
16-bit data 
Signal Line 

Figure 2: Diagram depicting the component composition and connections between the inside the Data Extraction Block.

Figure 3: Diagram depicting the component composition and connections inside the IMU Extraction Block that is part of the Data Extraction Block.

### **Accessing MPU9250 registers**

MPU9250 internal measurement unit uses the serial peripheral interface to communicate with the master device. The first bit (bit 15) sent to the device indicates a read (1) or a write (0). The next 7 bits indicate the actual address of the register to be accessed. The last 8 bits represent a dummy byte which is used to enable the device to write the 8-bit data from the accessed register back to the master device. Table 1 depicts the data format that is used to write to the device. A complete data sample is made up of 16-bits contained in two registers. For example, accelerometer data for the x-axis is contained in two registers, one register contains the most significant byte and the second contains the least significant byte. By combining these two bytes a complete x-axis accelerometer data sample is formed.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Description** | Read (1)/Write (0) | Address bit 6 | Address bit 5 | Address bit 4 | Address bit 3 | Address bit 2 | Address bit 1 | Address bit 0 | Dummy Byte Bit | Dummy Byte Bit | Dummy Byte Bit | Dummy Byte Bit | Dummy Byte Bit | Dummy Byte Bit | Dummy Byte Bit | Dummy Byte Bit |

Table : Data format used to write to MPU9250 registers.

### **Data Extraction Controller**

The extraction controller is responsible for the extraction and storage of the IMU data that it extracts. The controller is a state machine that consist of several states which it transitions between based on the input in order to accomplish its task. Table 2 describes the functions that the inputs and outputs of the extraction controller serve to drive the state transitions. The following descriptions of the states explain how the controller operates.

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Name** | **Input/Output** | **Usage** | **Signal Type** |
| CLK | Input | The clock input allows the component to be synchronised to a common clock across the system. Every component in the system has an input clock. | Clock |
| bufferEmpty | Input | The FIFO buffer can signal to the controller through this pin that it is either full (0) or empty (1). | High/Low |
| DONErx | Input | The SPI can signal the controller that a new sample has been extracted and can be stored. | One clock cycle pulse |
| EXTRACT | Output | This pin is used to command the SPI component to start the extraction of another sample. | One clock cycle pulse |
| NEXT\_ADD | Output | This pin is used to command the address block to load the next address to be transmitted to the IMU through the | One clock cycle pulse |
| STORE | Output | This pin is used to command the FIFO buffer to store the newly extracted data piece. | One clock cycle pulse |

Table2: Description of inputs and outputs of the Data Extraction Controller Component.

* When in the ‘IDLE’ state, the component waits for the bufferEmpty signal to go high in order to start the data extraction process. The controller transitions to the ‘NXTADDR’ state when the bufferEmpty signal goes high.
* When in the ‘NXRADDR’ state, the controller sends the ‘NEXT\_ADD’ signal to command the Address Loader component to load the new address on its output for the SPI master component to use. The controller then automatically transitions to the ‘STRTrx’ state.
* When in the ‘STRTrx’ state the controller sends a signal using the ‘EXTRACT’ pin to command the SPI component to start the transaction. The controller then automatically transitions to the ‘WAITINGrx’ state.
* When in the ‘WAITINGrx’ state, the controller waits for the ‘DONErx’ input to go high. This signal through this input notifies the controller that an SPI transaction has been completed. When this signal is received, the controller transitions to the ‘STOREstate’.
* When in the ‘STOREstate’, the controller sends a one clock cycle pulse using the ‘STORE’ output pin to command the FIFO buffer to store the data that the SPI extracted in the latest transaction. The controller then increments counter which tracks the number of transactions; this way the controller knows how many more data samples need to be extracted to have a complete set of gyroscope and accelerometer data samples (see Address Loader component). If the counter value is less than 11 then the state transitions back ‘NXTADDR’ state, otherwise the controller resets the counter to zero and transitions to the ‘W4BFFRrx’ state.
* When in the ‘W4BFFRrx’ state, the controller waits for the bufferEmpty signal to go low. This state is used to let the FIFO buffer finish its internal processes before it sends the signal as the buffer is designed to only holds as many samples as there are in a complete set of gyroscope and accelerometer data before it becomes full. If the controller transitioned to the IDLE state before the bufferEmpty signal transitioned to logic 0, the controller would continue initiating new sets of transactions causing data corruption in the later stages of operation such as data transmission. Adding this state between the IDLE state and the ‘STOREstate’ prevents this from happening. When the bufferEmpty input transitions to logic 0, the controller transitions to the IDLE state.

### **Address Loader**

The Address Loader component stores the addresses of the MPU9250 registers which contain the accelerometer and gyroscope data (See Appendix A). When commanded by the Data Extraction Controller this component loads these addresses on its output for the SPI component to use to write to MPU9250 in order to extract accelerometer and gyroscope data. Table 3 describes how each input and output of this component is used.

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Name** | **Input/Output** | **Usage** | **Signal Type** |
| CLK | INPUT | The clock input allows the component to be synchronised to a common clock across the system. Every component in the system has an input clock. | Clock |
| NXTADD | INPUT | This input is used by the extraction controller to command the addressBlock to assert next address on its output. | One clock cycle pulse |
| ADDRESS | OUTPUT | This output is connected SPI's tx\_data input such that the SPI can use the addresses to extract data from the IMU. | Data |

Table 3: Description of inputs and outputs of the Address Loader component.

The Address Loader component has an internal counter which tracks how many addresses have been loaded during the extraction process. This counter is reset to at the beginning of each new set of transactions and stops at the last address at the end of each set of transactions. This way the data is always extracted in the correct order. When the ‘NXTADD’ signal goes high, the component increments the counter and thereby asserts a new address on the ‘ADDRESS’ output for the SPI component to use.

### **FIFO Buffer**

This First In First Out Buffer design serves two purposes; the first is to store the data that was extracted from the internal measurement unit before it is sent to the ‘Data Processing Block’ and the second is to append a data identification byte the each 8-bit data piece extracted from the IMU. The reason for appending this bit is the fact that when the data is sent to the STM32L432 development board, the device needs to know what kind of data it is receiving such that it knows how to process and sue this data. This byte is also used by the STML432 for error checking to determine whether the batch of data was sent correctly. The identification byte conveys the following information (See Appendix B for more detail):

**IMU ID** – The IMU identification lets the STM32L432 determine which IMU the data came from so that it can apply appropriate offsets to the data that are unique to each internal measurement unit. Additionally, when the STM32L432 device sends the data to the FPGA for wireless transmission it needs to know what identification byte to append to the data such that the processing simulation also know how the data should be used.

**X, Y or Z axis** – Specifies whether the data byte conveys x, y or z axis information.

**Accel, Gyro or Mag** – Specifies whether the data byte comes from the accelerometer, gyroscope or magnetometer. The STM32L432 needs to know this information in order to process the data correctly as gyroscope data undergoes different processes to that of the accelerometer to obtain useful orientation information.

**Upper or Lower Byte** – Specifies whether the data byte is the most significant byte or the lest significant byte. This information is needed such that the two halves of the sample can be concatenated in the correct order.

The FIFO Buffer is a state machine which uses three states to store and send the data for transmission based on the input signals. Table 4 describes how each input and output of this component is used and the following description of each state the buffer can transition to explain the operation of the buffer.

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Name** | **Input/Output** | **Usage** | **Signal Type** |
| CLK | INPUT | The clock input allows the component to be synchronised to a common clock across the system. Every component in the system has an input clock. | Clock |
| storeData | INPUT | Using this pin the extraction controller can command the FIFO buffer to store the newly extracted sample. | One clock cycle pulse |
| sendData | INPUT | Using this pin the Transmission Controller can command the FIFO buffer to assert the oldest sample on the output such that it can be loaded to the FPGA-STM32L432 SPI master for transmission. | One clock cycle pulse |
| dataIn | INPUT | This pin is used to load the newly extracted sample from the SPI to the FIFO buffer. Data is only loaded when the extraction controller sends a pulse signal to the storeData pin. | Data |
| dataOut | OUTPUT | This pin is used to output the data to the FPGA-STM32L432 SPI master. This pin is connected to the accessControl block such that only the IMU\_Extraction\_Block which has access permission granted by the accessControl block can pass data to the SPI. New data is asserted on the output only when the transmission controller sends a pulse to the sendData pin. | Data |
| empty | OUTPUT | This pin is used by the FIFO buffer to signal to the extraction and transmission controllers that it is either full or empty. Extraction controller starts the process when the buffer signals that it is empty, and the Transmission controller starts the transmission process when the buffer signals that it is full. The state of this pin is high when empty and low when full. | High/Low |

Table 4: Description of inputs and outputs of the FIFO Buffer component.

* When in the ‘IDLE’ state, the buffer waits for the ‘storeData’ or ‘sendData’ input to go high. The buffer will transition to the ‘STORE’ state if the ‘storeData’ input is high and will transition to the ‘SEND’ state if the ‘sendData’ input goes high. The system was designed such that the two signals are never sent simultaneously from the two controllers and therefore there is no risk of race conditions occurring.
* When in the ‘STORE’ state, the buffer reads in the data into the next free space in the buffer and then increments the ‘Newest Sample’ (NS) and the ‘Next Free Space’ (FS) pointers in order to prepare to store the next sample. If the ‘Newest Sample’ pointer reaches a value of 11, the buffer will be considered full and the ‘empty’ output will go low in order to notify the controllers about the state of the buffer. After the buffer increments these pointers, it returns to the IDLE state. Each data read into the buffer is appended a unique identification based on the type of data and its origin (See Appendices A and B).
* When in the ‘SEND’ state, the buffer increments the ‘Oldest Sample’ (OS) pointer to assert a new piece of data on the output for transmission to the ‘Data Processing Block’ before returning to the ‘IDLE’ state. It is worth noting that the SEND signal is sent by the transmission controller after each transaction so that new data is loaded on the output before the next transaction. This happens because the ‘Oldest Sample’ always points to the oldest sample which is automatically asserted on the output by default and so the ‘Data Transmission Controller’ can start the first transaction before it signals to the buffer to assert the next piece of data on its output. When the buffer is commanded to assert a new piece of data on its output after the last data piece has been transmitted, the buffer resets the three pointers (FS, NS and OS) and sets the ‘empty’ output high to notify the extraction and transmission controllers that the buffer is now empty before returning to the ‘IDLE’ state.

### **Data Transmission Controller**

# **Appendix A**

Table of MPU9250 registers which contain accelerometer and gyroscope data and the order in which they are stored inside the ‘Data Loader’ component.

|  |  |  |
| --- | --- | --- |
| **Address Name** | **Address Value (Binary)** | **Order** |
| Accelerometer x-axis High Byte | 1011101100000000 | 1 |
| Accelerometer x-axis Low Byte | 1011110000000000 | 2 |
| Accelerometer y-axis High Byte | 1011110100000000 | 3 |
| Accelerometer y-axis Low Byte | 1011111000000000 | 4 |
| Accelerometer z-axis High Byte | 1011111100000000 | 5 |
| Accelerometer z-axis Low Byte | 1100000000000000 | 6 |
| Gyroscope x-axis High Byte | 1100001100000000 | 7 |
| Gyroscope x-axis Low Byte | 1100010000000000 | 8 |
| Gyroscope y-axis High Byte | 1100010100000000 | 9 |
| Gyroscope y-axis Low Byte | 1100011000000000 | 10 |
| Gyroscope z-axis High Byte | 1100011100000000 | 11 |
| Gyroscope z-axis Low Byte | 1100100000000000 | 12 |

# **Appendix B**

Information specifying the configuration and the information that will be sent to the Data Processing Block.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bits** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Section** | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| **Description** | IMU ID 2 | IMU ID 1 | IMU ID 0 | X, Y or Z axis 1 | X, Y or Z axis 0 | Accel, Gyro or Mag 1 | Accel, Gyro or Mag 0 | Upper or Lower Byte | DATA BIT 7 | DATA BIT 6 | DATA BIT 5 | DATA BIT 4 | DATA BIT 3 | DATA BIT 2 | DATA BIT 1 | DATA BIT 0 |

Section 1

000 – Internal Measurement Unit 1 (Upper Arm)

001 – Internal Measurement Unit 2 (Forearm)

010 – Internal Measurement Unit 3 (Hand)

011 – N/A

100 – N/A

101 – N/A

110 – N/A

111 – N/A

Section 2

00 - x-axis

01 - y-axis

10 - z-axis

11 - N/A

Section 3

00 – Accelerometer

01 – Gyroscope

10 – Magnetometer

11 – N/A

Section 4

1 – Most Significant Byte

0 – Least Significant Byte

Section 5

This section carries the actual data extracted from the IMU.