**Scope:**

This project requires the use of many serial communication channels to allow several devices to communicate with each other. The system uses several UART channels and one SPI channel. In order to reduce the complexity of the system, the same packet protocol was used for all of the UART channels. The QUARK thermal imager uses a predefined packet protocol, therefore the rest of the UART channels will adopt the same protocol. The SPI channels protocol is also predefined for the serial flash module. These protocols will be discussed in this section. The commands used in each of the communication channels will also be discussed in detail.

**UART Ports:**

**Port Settings:**

All of the serial port channels used the same port settings as shown in Figure 1.

|  |  |
| --- | --- |
| Parameter | Value |
| Signaling Polarity | Standard logic |
| Baud rate | /115200 |
| Data bits | /8 |
| Parity | None |
| Start bits | /1 |
| Stop bits | /1 |
| Flow control | None |
| Bit order | Least significant bit first |

Figure 1: Port Settings

Standard logic polarity is used, where the idle state is high, the start bit is low, the stop bit is high, logic 1 is high and logic 0 is low. This can be seen in Figure 2.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Idle | Start | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | Stop |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

Figure 2: Standard Logic Signaling Polarity

A baud rate of 115200 was chosen because it is a standard baud rate that is compatible with all of the components while being fast and reliable. The remainder of the port settings were chosen based on what is commonly used in UART communication.

**Cyclic Redundancy Check (CRC):**

In order to maintain and verify message integrity, the UART packet protocol makes use of an error-detection method, the cyclic redundancy check (CRC). Multiple CRC codes are used in the packet protocol to allow for some data to be verified even if part of the message was corrupted. In general, CRC uses an initialization value and a polynomial to generate a code that is specific to a data set. The methods used to calculate the CRC are discussed in the appendix. There are many implementations of CRC that use different polynomials and the initialization values, but this project uses CRC-CCITT initialized to 0x0000. CRC-CCITT uses the polynomial . This CRC generates a 2 byte CRC code that will be referred to from here on as the CRCn, where n is a number that refers to which CRC code in a packet protocol is being identified. The most significant and least significant byte of the CRCn will be indicated with the abbreviations MSB and LSB respectively (Texas Instruments, 2004), (FLIR, 2015).

**Packet Protocol:**

The packet protocol for the UART channels was adopted from the QUARK thermal imager (FLIR, 2015).The packet protocol consists of an 8 byte header describing the nature of the message, followed by payload of N+2 bytes containing the argument of the messages function. Both the header and the payload include a 2 byte CRC that verifies the integrity of each packet segment. The QUARKs packet protocol is shown in Figure 3 (FLIR, 2015).

|  |  |
| --- | --- |
| Byte # | Purpose |
| 1 | Process Code |
| 2 | Status |
| 3 | Reserved |
| 4 | Function |
| 5 | Byte Count (MSB) |
| 6 | Byte Count (LSB) |
| 7 | CRC1 (MSB) |
| 8 | CRC1 (LSB) |
| N | Argument |
| N+1 | CRC2 (MSB) |
| N+2 | CRC2 (LSB) |

Figure 3: UART Packet Protocol

The first byte of the UART packet protocol is the process code, which is used to detect the start of a new message. This byte will always have a value of 0x6E. The second byte of the protocol is the status byte. The status byte is used to transmit some information about the state of the transmitting device. The status byte will be discussed in detail in the Commands section below. The third byte is reserved and will only have a value of 0x00 except for the Main Controller to Computer channel, where it will indicate the Main Controller type. The fourth byte is the Function byte, which is used to identify the purpose of the message. The values and meanings of this byte is different for each channel, and is discussed in detail in the Commands section below. Bytes five and six are the byte count, and are used to identify how many bytes will be transmitted in the payload minus the two CRC bytes. The byte count used two bytes to allow for large payloads of up to 65535 bytes. If the byte count was only one byte, the payload would be limited to 255 bytes, slowing down the transmission and adding a lot of overhead. Bytes seven and eight are CRC1, which is the CRC code calculated from bytes 1 through 6. The next N bytes are the argument of the function identified in byte 4. The length of N is the value of the byte count from bytes 5 and 6. The meaning of the argument bytes is different for each command in each channel, and will be discussed in detail below. Bytes N+1 and N+2 are CRC2, which is the CRC code calculated from the N bytes of the argument.

**Commands:**

Even though all of the UART channels use the same command protocol, each channel uses different sets of commands. These commands will be discussed in depth in this section. This section is divided into subsections that describe the specific commands of each UART channel.

Computer to Main Controller:

The computer to main controller UART channel is the only communication between the user interface and the system. The status byte is only used to indicate if the user interface wants to receive status updates or not. The status byte will have a value of 0x00 for all commands except the “Detect System” commands. The computer only uses a small set of messages to send the Main controller. These commands are discussed below.

Abort/Shutdown:

The abort/shutdown message is sent to the Main Controller to quickly shutdown the system. The main controller is expected to respond to the “Abort/Shutdown” command with the “ACK” command. Figure 4 shows the Abort/Shutdown command.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | 0x00 | 0x00 | 0x00 | 0x0000 | | 0xDFBB | |

Figure 4: Computer to Main Controller, Abort/Shutdown Command

ACK:

The “ACK” message is only sent to the main controller from the computer to acknowledge the receipt of a “Data Transfer” message from the main controller. No response to the “ACK” message is expected. Figure 5 shows the ACK command.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | 0x00 | 0x00 | 0x01 | 0x0000 | | 0xE88B | |

Figure 5: Computer to Main Controller, ACK Command

Detect System:

The “Detect System” message is the first command the computer will send to the main controller, and it should only be sent once. In the “Detect System” command, if the user interface want so receive status updates the status byte will have a value of 0xFF otherwise it will have a value of 0x00. The main controller is expected to respond to the “Detect System” command with a “Status Update” message. If the “Detect System” commands status byte was a value of 0xFF, the main controller should transmit a “Status Update” message every time one of the main controllers system status variables changes in order to keep the graphical user interface updated. Figure 6 shows the two possible “Detect System” commands.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | 0x00 | 0x00 | 0x02 | 0x0000 | | 0xB1DB | |
| 0x6E | 0xFF | 0x00 | 0x02 | 0x0000 | | 0xEB74 | |

Figure 6: Computer to Main Controller, Detect System Command

Begin Process:

The “Begin Process” command is sent to the main controller to start the automated procedure of collecting thermal images and processing the data. The computer expects an “ACK” message from the main controller in response. The status byte of the packet is used to transmit the desired delay between image sets. This delay is given in seconds, allowing for a range between 0 and 255 seconds. The delay represents the time between the conclusion of one image set capture and the start of the next image set capture process. This feature is included to allow for perspective changes in order to improve the result of the structure from motion algorithm. Figure 7 shows the “Begin Process” Command.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | Image Spacing Delay | 0x00 | 0x03 | 0x0000 | | CRC1 | |

Figure 7: Computer to Main Controller, Begin Process Command

Begin Transmitting Data:

The “Begin Transmitting Data” command is sent to the main controller to start the process of transmitting the stored data to the computer. The computer expects an “ACK” message from the main controller in response. Figure 8 shows the “Begin Transmitting Data” command.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | 0x00 | 0x00 | 0x04 | 0x0000 | | 0x037B | |

Figure 8: Computer to Main Controller, Begin Transmitting Data Command

Orientation Data:

The “Orientation Data” command is sent to the main controller as a response to the command “Request Orientation Data” from the main controller. The computer expects an “ACK” message from the main controller in response. The “Orientation Data” is the only command sent from the computer that sends argument data and a second CRC code. The argument contains three values transmitted in eight byte ascii format. The first byte contains the ascii character for the sign of the value, the second to fourth bytes contain the ascii values for three places of the integer part of the value, followed by a byte containing the ascii value for a decimal, finished by the sixth through eighth bytes containing the ascii values for the first three digits of the fractional part of the value. For example, if the value 359.824 was being transmitted, the character array {+,3,5,9,.,8,2,4} would be transmitted. The first value contains the current latitude in decimal degree format. The second value contains the current longitude decimal degree format. The third value contains the current azimuth in decimal degree format. These values are used by the system to perform translation and rotation calculations. Figure 9 shows the message header of the “Orientation Data” command. Figure 10 shows the payload structure for the “Orientation Data”.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | 0x00 | 0x00 | 0x05 | 0x0018 | | 0xA772 | |

Figure 9: Computer to Main Controller, Orientation Data Command: Header

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bytes 9 - 16 | Bytes 17 - 24 | Bytes 25 - 32 | Byte 33 | Byte 34 |
| Current Latitude | Current Longitude | Current Azimuth | CRC2 | |

Figure 10: Computer to Main Controller, Orientation Data Command: Payload

NACK:

The “NACK” message is sent from the Computer to the Main Controller when it receives an incomplete or corrupted message packet from the Main Controller. When the Main Controller receives the “NACK” message, it will retransmit the message if it was a “Request Orientation Data” or “Data Transfer” message. Figure 11 shows the “NACK” message.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | 0x00 | 0x00 | 0xFF | 0x0000 | | 0x10D8 | |

Figure : Computer to Main Controller, NACK Command

**INSERT OTHER COMPUTER TO MAIN CONTROLLER HERE**

Main Controller to Computer:

The main controller to computer UART channel is used to update the system status display on the graphical user interface and to transmit data from the system to the computer. The reserved byte is used in this UART channel to transmit the System ID that indicates the Main Controller type. The System IDs for the various Main Controller types are shown in Figure 11.

|  |  |
| --- | --- |
| System ID | Main Controllers |
| 0 | MSP-430/FPGA |
| 1 | Raspberry-pi |
| 2 | MSP-430 |
| 3 | FPGA |
| 4 | .......... |

Figure 12: Main Controller to Computer, System ID

The status byte is used to indicate which stage the system is presently in. Each main controller has a different set of stages which are discussed in the main controller chapter. Therefore, each main controller has a different Status code discussed below. The commands that the main controller uses to communicate with the computer are also discussed below.

Status Byte:

As discussed above, each main controller uses a different status byte code. The value of the status byte reflects which stage the system is currently in. These stages are described in the earlier chapters of this paper. The following figures describe the status byte codes for the different main controllers. Figure 12 shows the status byte code for the MSP430/FPGA main controller. Figure 13 shows the status byte code for the Raspberry-pi main controller. Figure 14 shows the status byte code for the MSP-430 main controller. Figure 15 shows the status byte code for the FPGA main controller.

|  |  |
| --- | --- |
| Status Byte Value | System Stage |
| 0x00 | Stage 0 |
| 0x01 | Stage 1 |
| 0x02 | Stage 2 |
| 0x03 | Stage 3 |
| 0x04 | Stage 4 |
| 0x05 | Stage 5 |
| 0x06 | Stage 6 |
| 0x07 | Stage 7 |
| 0x08 | Stage 8 |
| 0x09 | Stage 9 |

Figure 13: MSP430/FPGA Main Controller Status Byte Code

|  |  |
| --- | --- |
| Status Byte Value | System Stage |
| 0x00 | Stage 0 |
| 0x01 | Stage 1 |
| 0x02 | Stage 2 |
| 0x03 | Stage 3 |
| 0x04 | Stage 4 |
| 0x05 | Stage 5 |
| 0x06 | Stage 6 |
| 0x07 | Stage 7 |
| 0x08 | Stage 8 |
| 0x09 | Stage 9 |

Figure 14: Raspberry-pi Main Controller Status Byte Code

|  |  |
| --- | --- |
| Status Byte Value | System Stage |
| 0x00 | Stage 0 |
| 0x01 | Stage 1 |
| 0x02 | Stage 2 |
| 0x03 | Stage 3 |
| 0x04 | Stage 4 |
| 0x05 | Stage 5 |
| 0x06 | Stage 6 |
| 0x07 | Stage 7 |
| 0x08 | Stage 8 |
| 0x09 | Stage 9 |

Figure 15: MSP430 Main Controller Status Byte Code

|  |  |
| --- | --- |
| Status Byte Value | System Stage |
| 0x00 | Stage 0 |
| 0x01 | Stage 1 |
| 0x02 | Stage 2 |
| 0x03 | Stage 3 |
| 0x04 | Stage 4 |
| 0x05 | Stage 5 |
| 0x06 | Stage 6 |
| 0x07 | Stage 7 |
| 0x08 | Stage 8 |
| 0x09 | Stage 9 |

Figure 16: FPGA Main Controller Status Byte Code

Error Encountered/Shutdown:

The “Error Encountered/Shutdown” command is sent when the main controller encountered an error or entered into an unknown state that requires the system to shutdown. No response is expected from the computer. The “Error Encountered/Shutdown” command is shown in Figure 16.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | Status Byte | System ID | 0x00 | 0x0000 | | CRC1 | |

Figure 17: Main Controller to Computer, Error Encountered/Shutdown Command

ACK:

The “ACK” command is sent to acknowledge the receipt of one of several commands: “Abort/Shutdown”, “Begin Process”, “Begin Transmitting Data”, and “Orientation Data”. The “ACK” command expects no response from the computer. Figure 17 shows the ACK command.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | Status Byte | System ID | 0x01 | 0x0000 | | CRC1 | |

Figure 18: Main Controller to Computer, ACK Command

Status Update:

The “Status Update” command is used to update the graphical user interface on the computer with the current state of the system. The “Status Update” command is sent whenever the stage of the system changes if status updates are enabled. No response is expected for this command. The “Status Update” command is shown in Figure 18.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | Status Byte | System ID | 0x02 | 0x0000 | | CRC1 | |

Figure 19: Main Controller to Computer, Status Update Command

Request Orientation Data:

The “Request Orientation Data” command is used to request the current orientation information from the computer. This command is used when the orientation data is needed for rotating and transforming captured images to allow for further data processing. The “Orientation Data” message is expected from the computer in response to this function. The “Request Orientation Data” command is shown in Figure 19.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | Status Byte | System ID | 0x03 | 0x00 | | CRC1 | |

Figure 20: Main Controller to Computer, Request Orientation Data

Data Transfer:

The “Data Transfer” command is used to transfer stored data to the computer. An “ACK” message is expected from the computer in response to this command. The size of the payload, represented by N, can vary up to 65535 bytes, but is usually the length of one line of the image plus four. The “Data Transfer” header is shown in Figure 20, and the payload is shown in Figure 21.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
| Process Code | Status | Reserved | Function | Byte Count | | CRC1 | |
| 0x6E | Status Byte | System ID | 0x04 | N | | CRC1 | |

Figure 21: Main Controller to Computer, Data Transfer Header

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Byte 9 | Byte 10 | Byte 11 | Byte 12 | Bytes 13 - N | Byte N+1 | Byte N+2 |
| Packet Number | | Total Packets | | Data | CRC2 | |

Figure 22: Main Controller to Computer, Data Transfer Payload

The “Data Transfer” payload starts with a two byte value “Packet Number” which identifies which packet is contained in the payload. The following two bytes contain the “Total Packets” value which identifies the number of packets in the file being transmitted. The “Packet Number” starts at one for the first data packet of a file and increments by one for each following packet until the value equals the “Total Packets” value.

Main Controller to QUARK:

J

MSP430 to FPGA:

**TBD**

FPGA to MSP430:

**TBD**

**SPI Ports:**

**Serial Flash:**

**TBD**