



Norwegian University of
Science and Technology

System Integration of Unmanned Aerial Vehicle with Thermal Camera

Elizabeth Roy

Embedded Computing Systems

Submission date: January 2016

Supervisor: Thor Inge Fossen, ITK

Co-supervisor: Tor Arne Johansen, ITK
Frederik Stendahl Leira, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

This thesis describes the implementation of two systems, both relevant to the FLIR Tau 2 thermal camera, for use in a UAV payload with the aim of increasing the efficiency and cost-effectiveness of the existing system. The first is a camera control interface, enabling settings of the camera, such as brightness, contrast or region of interest to be changed remotely from the ground station during UAV flight. This involved embedding a camera control GUI into the existing ground station software, adding to the existing message passing system and implementation of a payload computer driver to interface directly with the camera. All modules were successfully implemented and the interface can be used with the existing UAV system, increasing efficiency of UAV missions by removing the requirement to land for changing of camera settings. The second is a custom PCB containing a microcontroller, designed to retrieve digital data directly from the camera. The design aims to render the existing system's analog-to-digital converter and its unique, inconvenient power source obsolete, effectively decreasing the payload weight by approximately one third and increasing cost-effectiveness of a UAV mission by increasing the flight time to fuel ratio. Time constraints meant only the design of the PCB hardware was completed, and the prototypes of the microcontroller program and the payload computer driver were unable to be verified. The hardware design does however provide a solid foundation for future work, after which a suitable solution should be found.

Acknowledgment

To my co-supervisor Frederik Stendahl Leira, for always answering my questions, helping me with whatever I needed, and proof reading. To my other co-supervisor Tor Arne Johansen, for reassuring me that everyone panics. To João Fortuna for helping me with fundamental understanding. To Artur Piotr Zolich for helping me with the final PCB layout. To Daniel Ryan for introducing me to the wonderful world of embedded systems. To Brendan Williams for being the best mentor anyone could ask for. To Bernt Breivik for being a patient fountain of knowledge when I was learning about complex microcontroller programs. To my family and friends for the constant support whether it be face to face or from across the globe. And finally, last but not least, to my supervisor Thor Inge Fossen for agreeing to be my supervisor earlier than normal, enabling me to return to the beautiful Norway to write my thesis.

Thank you all so much.

Tusen takk.

E.R.

Preface

This thesis was completed as part of the Erasmus Mundus Embedded Computing Systems (EMECS) master program, with the Unmanned Aerial Vehicle Laboratory at the Norwegian University of Science and Technology (NTNU) in the autumn semester of 2015. Its primary aim is to make improvement contribution to the existing design of a UAV payload including a thermal imaging camera. This system is designed to be an easy-to-use and robust UAV, for use in applied research such as search and rescue missions, emergency situations, sea traffic surveillance and livestock management by a non-expert operator. This thesis primarily aims to decrease the necessary payload weight of the pre-existing design, by utilizing the capability of the FLIR Tau 2 thermal camera to output digital imaging data. It also aims to implement a real-time control interface for the camera, enabling camera settings to be changed in-flight. It is assumed a reader of this document has a technical background, with knowledge in basic UAV definition and functionality, the basics of signal processing, particularly the difference between analog and digital data, and the steps involved in implementing an embedded system.

Contents

Abstract	i
Acknowledgment	ii
Preface	iv
1 Introduction	1
1.1 Background	1
1.2 Objectives	6
1.3 Limitations	6
1.4 Approach	7
1.5 Structure of the Report	9
2 Control Interface	11
2.1 Components	11
2.2 Implementation	12
2.3 Modular Testing	15
3 Digital Data Retrieval	23
3.1 Digital Data Formats	23
3.2 Hardware Options for Retrieval	25
3.3 Custom PCB Functionality	28
3.4 Hardware Design	28
3.5 Firmware and Software Design	36
4 Summary and Recommendations for Further Work	43
4.1 Summary and Conclusions	43
4.2 Discussion	46
4.3 Recommendations for Further Work	50

4.3.1	Short Term	50
4.3.2	Mid Term	50
4.3.3	Long Term	51
A	Acronyms	53
B	List of Camera Control Commands	55
C	Custom PCB Design	61
D	Demos and Accompanying Files	67
	Bibliography	69

List of Figures

1.1	FLIR Tau 2 thermal camera with 19mm lens. From [14].	2
1.2	A depiction of the functional components contained in the current payload system.	3
1.3	The existing system's software toolchain. Adapted from [41].	4
1.4	An abstract depiction of the approach to implementing a method of retrieval of digital data from the thermal camera.	9
2.1	Abstract depiction of the camera control interface components.	12
2.2	Camera control command packet protocol. Adapted from [13].	13
2.3	Camera control GUI location within Neptus, the ground station software.	14
2.4	Camera Control GUI Status Tab.	15
2.5	Camera Control GUI Setup Tab.	16
2.6	Camera Control GUI Analog Video Tab.	17
2.7	Camera Control GUI Digital Video Tab.	18
2.8	Camera Control GUI AGC/DDE Tab.	19
2.9	Camera Control GUI ROI Tab.	20
2.10	Camera control IMC command format.	21
3.1	LVDS line timing. Adapted from [9].	24
3.2	LVDS valid bit sets [9].	25
3.3	CMOS timing diagrams. Adapted from [9].	26
3.4	FLIR Tau PCB Wearsaver with Solder Pads.	27
3.5	FLIR Camera Link Expansion Board.	27
3.6	Abstract depiction of custom PCB functionality.	29
3.7	The final layout design of the custom PCB.	30
3.8	EFM32 Microprocessor manually defined library package.	33

3.9 Length match check figures for the CMOS signals in the EAGLE PCB layout software.	34
3.10 CMOS signal lines highlighted in the PCB layout.	35
3.11 Manufactured custom PCB without components placed.	36
3.12 First prototype functionality of the MCU program on the custom PCB. . .	37
3.13 USB descriptor heirarchy.	40
C.1 Custom PCB Schematic	62
C.2 Custom PCB Bill of Materials	63
C.3 Custom PCB Layout without ground plane polygons filled.	64
C.4 Custom PCB front layout with ground plane.	65
C.5 Custom PCB back layout ground plane.	66

Chapter 1

Introduction

Unmanned Aerial Vehicles are advantageous in several ways, primarily in that they do not require an onboard pilot, and they are therefore able to venture into environments which pose a risk to human life. They are also able to perform repetitive monitoring or searching of a region for long periods of time, remotely and autonomously, in all kinds of weather, meaning they can be considerably more efficient and cost effective in completing a task than a piloted aircraft.

1.1 Background

At the NTNU UAV-Lab, there are several such systems in development, being actively used in applied research projects. One particular system is of focus in this thesis, which originally was designed to be fitted in an X8 airframe, with the focus of the payload being a thermal imaging camera, the FLIR Tau 2 [17], seen in Figure 1.1 with a gimbal allow remote or onboard control of pan and tilt. This further extends the capability of the UAV to be useful in harsh weather and darkness, in that it does not rely on visible light to provide imaging data. It also proves very useful in maritime environments, which generally have points of interest with an apparent temperature differential to the ocean, such as ships or marine life [53]. Payload weight is a very important consideration in choosing the appropriate system for a particular application. A lighter payload means more flight time, in that the same amount of fuel can sustain flight for longer with less of a load, which is especially advantageous in an application involving search or monitoring, as this one is.



Figure 1.1: FLIR Tau 2 thermal camera with 19mm lens. From [14]

Remark: The ‘payload’ of a UAV is the equipment carried by the aircraft that is not crucial for operational reasons such as flight, communication and navigation.

The main focus of this thesis is to make improvements to the existing design, which is comprised of the following setup, depicted in Figure 1.2. This was primarily interpreted as a focus in decreasing the payload weight, while also maintaining the robustness of the system and its easy-to-use nature, even by a non-trained operator. The operational part of the UAV system will not be changed.

The central component of the existing payload design is a custom 3-port ethernet switch, with a serial-to-ethernet interface. It is connected to all other components of the payload, including the radio modem, which receives messages from the ground station. Every component of the payload, as well as the ground station can send and receive messages through this switch. The FLIR Tau 2 thermal camera analog output is connected to an analog-to-digital converter, which feeds digital image data into this custom switch. It is then transmitted via the radio modem to the ground station, while the onboard computer is also allowed to record video if commanded to. The autopilot is also connected to the central switch, through which the camera gimbal is also controlled, acting as an additional servo to those for the flight control surfaces. Also not included in the diagram is the power source, providing 12V power, and two transformers, providing a step up to 48V for the analog-to-digital converter and a step down to 5V for the payload computer.

The software toolchain relevant to the existing system is depicted in Figure 1.3.

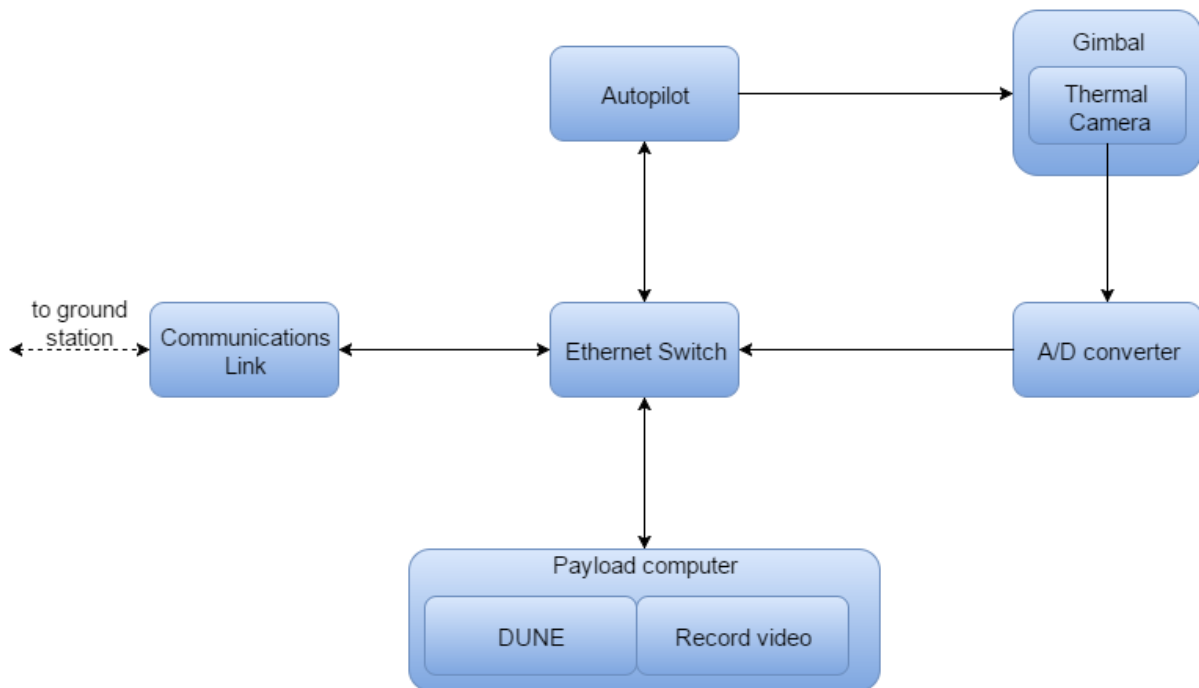


Figure 1.2: A depiction of the functional components contained in the current payload system.

This includes ground station software, a message passing protocol and the onboard computer software. Firstly, the ground station software is called Neptus. It is a "distributed command and control infrastructure for the operation of all types of unmanned vehicles. Neptus supports the different phases of a typical mission life cycle: planning, simulation, execution and post-mission analysis" [40]. In the case of this document, it will be referred to as the ground station software. Next, IMC or Inter-Module Communication protocol "defines a common control message set understood by all [...] nodes in networked environments" [39]. It will be referred to as the existing message passing protocol. Lastly, the payload computer software, called DUNE, is the "embedded software at the heart of the vehicle, [containing tasks] for control, navigation, simulation, networking, sensing and actuation" [38]. It is essentially a collection of tasks which are overseen by the same scheduler, in order to efficiently perform tasks; it will be referred to as the payload computer software.

The main goal of this thesis is to eliminate the need for the analog-to-digital converter present in the existing system and retrieve the digital data from the camera directly. Given that the camera already in use is capable of outputting several different formats of digital data, and it was immediately available, no other cameras were considered. The camera settings are changed through a serial interface, with a simple

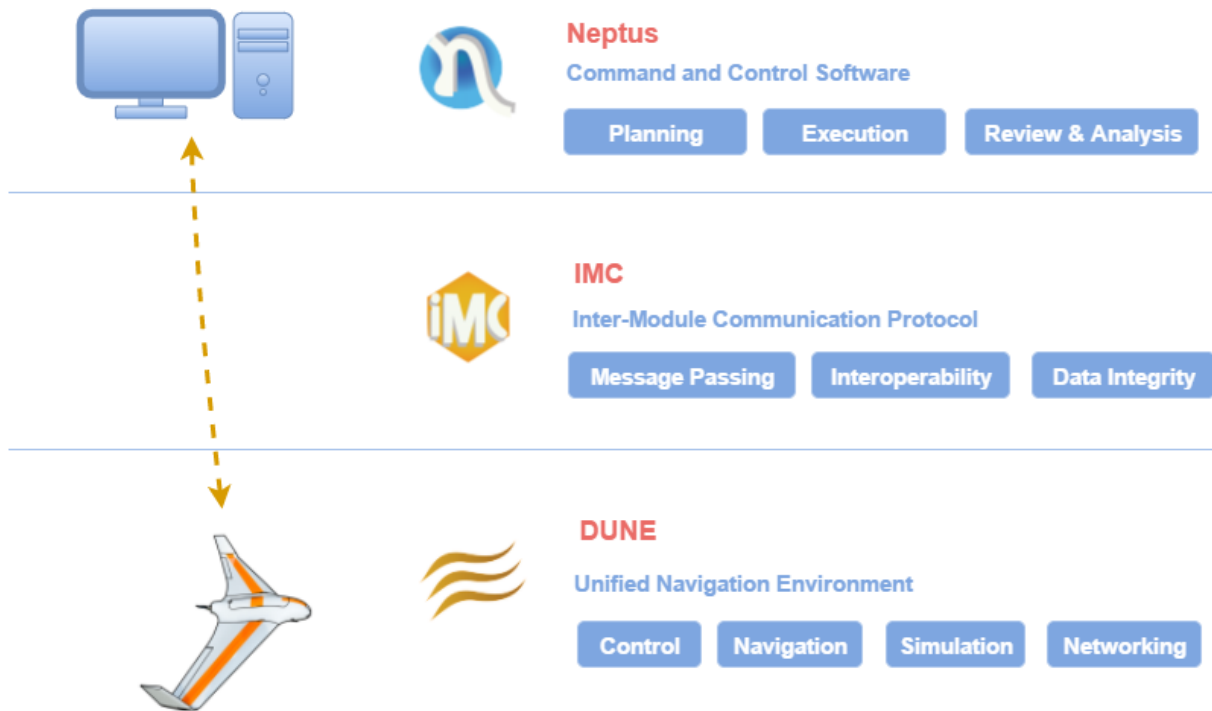


Figure 1.3: The existing system's software toolchain. Adapted from [41].

protocol and a relatively small amount of data [13]. It also has a feature called external sync, which enables a master/slave relationship between the FLIR Tau 2 and other payload components, for the purpose of synchronisation [9]. This feature could potentially be used for synchronisation with another camera, operating in the visual spectrum as opposed to the infrared spectrum, to provide additional video data, or with a positional component [17], to provide positional data corresponding to the current image frame. Considering the significant potential applications that access to these features would provide, it was decided to add access to camera control communications and its external sync function to the objective requirements. There are three different camera accessories available from FLIR, all of which were considered during the approach decision process. The first is the VPC Module, which provides access to the analog data via a MCX-to-BNC cable, and access to control communications via mini-USB. This is the module currently in use, but is unsuitable for use during this thesis as it does not provide access to any digital data [12].

Both other options for retrieval of camera image data do allow for access to digital data. The first is the Camera Link Expansion Board, and FLIR recommends using either this or the aforementioned VPC Module with the camera [10]. The Camera Link accessory allows access to everything being transmitted by the camera, but requires

the use of a Camera Link connector, which cannot easily be interfaced with using the existing payload computer board. Camera power and communications are provided with an additional USB connection, and the external sync function cannot be accessed without modification to the module. It is also designed to be used with the FLIR Camera GUI (for Windows), and therefore it would be difficult to interface with using the existing payload computer (running Linux). This accessory is however a viable option in attempting to achieve retrieval of digital data, albeit at a cost of USD1500. In an effort to decrease the potential cost of the system to the end user, and keeping ease-of-use in mind, it was not considered a primary option, but instead will be used as a comparison solution during future work.

The last option is the PCB Wearsaver with Solder Pads, which is not strictly defined as a data retrieval accessory by FLIR, but implied as a means to evaluate the camera performance (however there is no documentation or comment from FLIR on their website explicitly stating its purpose [11]). This accessory provides direct access to the analog and digital LVDS signals from the camera with no processing via solder pads, and also allows for control communications. It does not provide access to any other digital data, or the external sync function. LVDS, or Low Voltage Differential Signalling, is difficult to process due to its low voltage (max 350mV) and the necessity to differentiate between a pair of signals for a result [21]. Using this accessory would require extensive processing on the payload computer, and a custom receiver to decode and amplify the LVDS data. This solution was also deemed insufficient for attempting a solution, however was the inspiration for the eventual decided approach.

1.2 Objectives

With an already functioning system able to be used within the university department for research, optimisation becomes a goal when the system is going to be used externally, such as commercially or in emergency situations, where efficiency and cost effectiveness become crucial. All objectives therefore relate to optimisation of and improvement to the existing payload design, with the goal of increasing efficiency and cost-effectiveness. They are as follows:

1. Implement a control interface for the FLIR Tau 2 thermal camera, enabling settings to be changed in-flight, eliminating the need for landing purely for this purpose.
2. Implement a method to retrieve digital imaging data from the FLIR Tau 2 camera, thereby making the analog-to-digital converter and its power source component redundant; this comprises approximately 300g of the existing 1kg payload. Access to the camera serial interface and external sync functions should be included.

1.3 Limitations

In attempting to achieve these objectives, there are some limitations which need to be considered before deciding an approach, whether they are physical, operational or environmental.

As was aforementioned, this thesis is focused on improving an existing system, therefore limiting the choices of components that are to be used in the payload. The camera in the existing system is the FLIR Tau 2 thermal camera, which is capable of outputting digital imaging data, and therefore it is possible to eliminate the need for an analog to digital converter while keeping it. It also saved time in that the necessary equipment was quickly available.

While keeping the existing components, there is a requirement to keep the solution easy to use by a non-expert operator. Therefore there should not be any complicated connectors or operational processes before it can be used, and the solution should be as close to “plug-and-play” as possible.

Again, due to the fact that there is an existing system to be improved, the existing software (toolchain depicted in Figure 1.3) is to be extended to encompass the improvements. This means the onboard computer software is to be written in object-oriented C++ within the current payload software, DUNE, the ground station GUI software is to be written in Java, within the current ground station software, Neptus, and the existing message passing system is to be used, IMC. Particularly the Java aspect of this limitation is further considerable in terms of time, due to the lack of experience of the programmer in Java. Any other skills which are lacking will obviously also lead to a similar limitation, as with any project. The nature of the existing project is that it is continually evolving, with many people constantly making changes. This creates a limitation in that it is considerably difficult to gauge the goal of other people's work, especially in a programming environment. There is also a limited amount of documentation to assist in this aspect, and therefore creates the need for more time to be spent on system familiarisation and understanding.

The full UAV system is constantly in use for different research purposes by various people in the department, and therefore it is very rarely available. While the entire system is not absolutely necessary in order to develop the designs in this thesis, it does pose a limitation in the sense that comprehensive validation of the complete system may not be possible, when considered in conjunction with the correct stage of execution coinciding with the availability of the system. It is also unfortunate timing that the final stages of this thesis are in midwinter in Norway, meaning the weather conditions are not optimal for preliminary testing of a first prototype system.

1.4 Approach

Regardless of what limitations may exist, it is possible to make significant improvement to the existing system while maintaining its robustness and ease of use. While referring to the objectives stated above, the following section outlines the approach to be made in achieving them. Further detail can be found later in the report.

Firstly, the camera control interface will be implemented in the initial system familiarisation phase. It is an opportunity to engage the benefits of kinaesthetic learning while developing an understanding of the existing system, particularly the software el-

ement. This will involve addition of:

- a message packet structure containing camera setting information to be defined in the message passing software, IMC;
- a driver task for the onboard computer software program, DUNE, to communicate with the thermal camera via a USB connection;
- a section to be implemented in the ground station software, Neptus, to enable the user to easily change camera settings via a GUI.

This task is not of a particularly high priority, but aims to enable the time used to become familiar with the existing system to be useful beyond developing understanding, since it has elements in all main sections of the system software.

Secondly, and more significantly, the retrieval of the digital data will be approached after a sufficient understanding of the existing system has been achieved. Since the relevant equipment available from the manufacturer for the FLIR Tau 2 thermal camera are not exact solutions, an alternative approach will be used, involving the design of a custom PCB, involving:

- operational, schematic and layout design of the hardware, enabling access to digital data, serial communication with the camera, the external sync function, necessary MCU peripherals and USB functionality;
- component selection and design completion, ready for manufacturing;
- implementation of a program for the microcontroller, enabling digital data retrieval, handling of the camera control interface messages and USB functionality;
- implementation of a driver task for the payload computer, responsible for compressing the image data and forwarding it to the ground station.

The overall payload layout will therefore be changed according to Figure 1.4, in contrast to Figure 1.2 on page 3, depicting the existing system. The imaging data from the camera will be captured and processed by the custom PCB MCU, and sent to the onboard computer via an easy-to-use USB connection. Here the image frame will be compressed and sent as an IMC message to the ground station. Handling the imaging data in this way also enables potential association of relevant positional data, not to mention the decrease in payload weight.

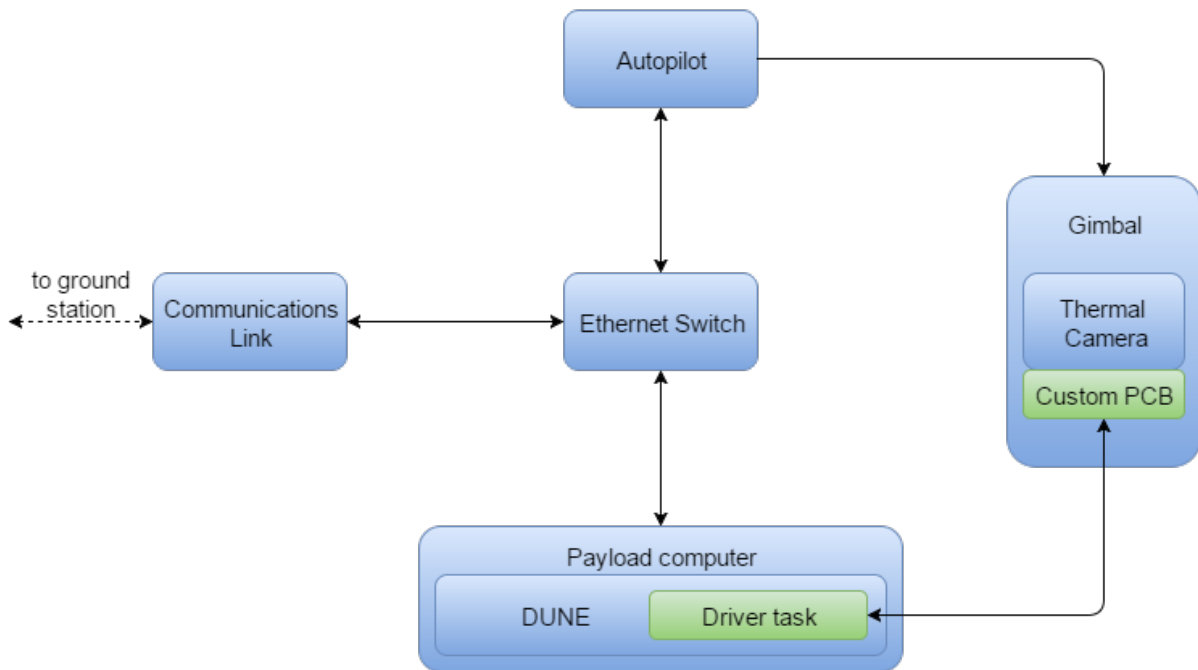


Figure 1.4: An abstract depiction of the approach to implementing a method of retrieval of digital data from the thermal camera.

1.5 Structure of the Report

The remainder of the report is structured as follows. Chapter 2 contains the specifics relevant to implementing the Camera Control Interface, performed during the initial system familiarisation phase. Chapter 3 outlines the solution to Retrieval of Digital Data objective, including the design of the custom PCB and relevant software. Chapter 4 is the Conclusion, with a summary, conclusions, discussion and an outline of possible future work. This is followed by Appendices containing Acronyms, an extended list of implemented and encoded camera control commands, the custom PCB design and the attached zip file description respectively. Lastly, a Bibliography.

Chapter 2

Control Interface

At the beginning of any project, it is necessary for those involved to gain an understanding of the problem, and to become familiar with the existing system if one exists. In this instance, the system familiarisation stage was completed simultaneously with the creation of a camera control interface, in order to gain the advantages associated with kinaesthetic learning. This was ideal in that the camera control interface task has implementable components in all parts of the system which need to be understood. It is also not extremely high priority, in that this feature is not critical to UAV mission completion and therefore there would be low impact were the attempt to be unsuccessful. However, upon successful completion, it would prove to be considerably advantageous in improving the efficiency and cost-effectiveness of the existing system [53], in that it would allow for camera settings to be changed in-flight, instead of requiring landing of the UAV. Camera command examples include changing brightness and contrast, selecting a region of interest or storing the current image frame in the camera's memory. The diagram in Figure 2.1 outlines the components of the camera control interface.

2.1 Components

The commands are sent from the ground station GUI by the user, via a specific pre-defined message format to the payload computer. The message is then decoded by a driver task and a command is sent to the camera via USB. Control commands sent directly to the camera need to be structured as in Figure 2.2. Contained within the message is a process code used for verifying communication type, status of the cam-

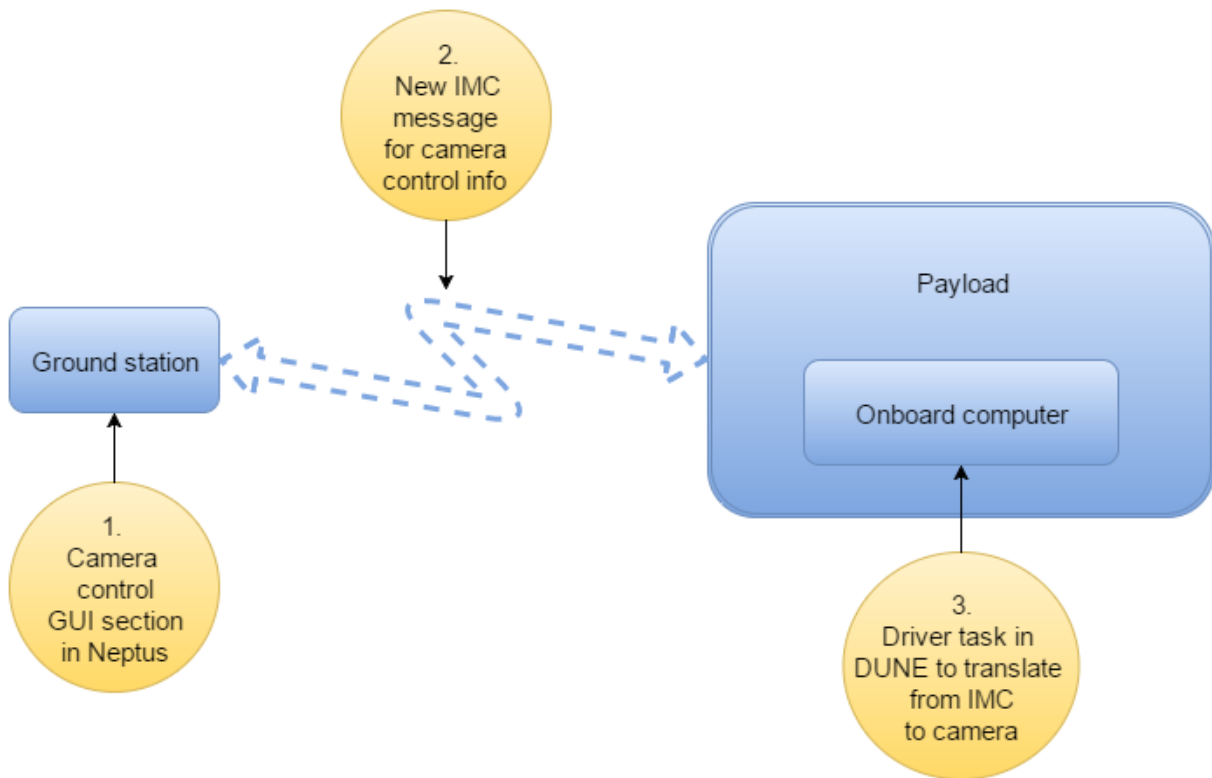


Figure 2.1: Abstract depiction of the camera control interface components.

era used in reply messages, a function code to indicate the camera setting the message pertains to, a byte count relevant to the argument field, a header CRC code, the arguments relevant to the camera setting, and a full message CRC code. Specific commands are defined uniquely by a combination of function code and a set of arguments.

2.2 Implementation

This information is hardcoded into the camera control GUI, along with a description for human inspection, and an indication of where the relevant replies should be handled in the GUI. This keeps modularisation and abstraction at a satisfactory level for understanding by others who may work on the same project at a future stage. The camera control GUI can be found in the Tools menu of the ground station UAV panel, as seen in Figure 2.3.

The GUI is modelled after the provided camera control GUI from FLIR [8] because it provides significant ease of use, however was recreated using Java in order to embed it seamlessly into the existing ground station software, Neptus [7]. This was done in

Byte #	Upper Byte
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 2.2: Camera control command packet protocol. Adapted from [13].

Eclipse IDE for Java [16], using the swing classes specifically defined for GUI development [45, 46]. Depictions of the embedded GUI can be seen in Figures 2.4 through 2.9. A combination of GUI features provide an easy-to-use interface capable of transmitting FLIR Tau 2 control messages, without requiring the user to have prior knowledge of the camera capabilities or setting limits. These include radio buttons to indicate mutually exclusive choices, sliders to indicate minimum and maximum values, and buttons to indicate a one-time action.

When a button is clicked in the camera control GUI, the module in charge of that particular GUI panel encodes the aforementioned function code and arguments into an IMC message. The IMC message protocol is used by the existing system [42, 47], and therefore all system modules are already equipped to encode and decode these messages. In order to send the camera control information, a new IMC message type needed to be added to the record, and the library re-built. This allows for the existing message passing protocol to be used while also extending its functionality by enabling the camera control information to be sent. The new IMC message type is indicated in Figure 2.10a as XML code and in Figure 2.10b abstractly. It can be seen that the IMC message is designed to closely reflect the camera's serial communication protocol. This is done in order to simplify the task of the payload computer driver.

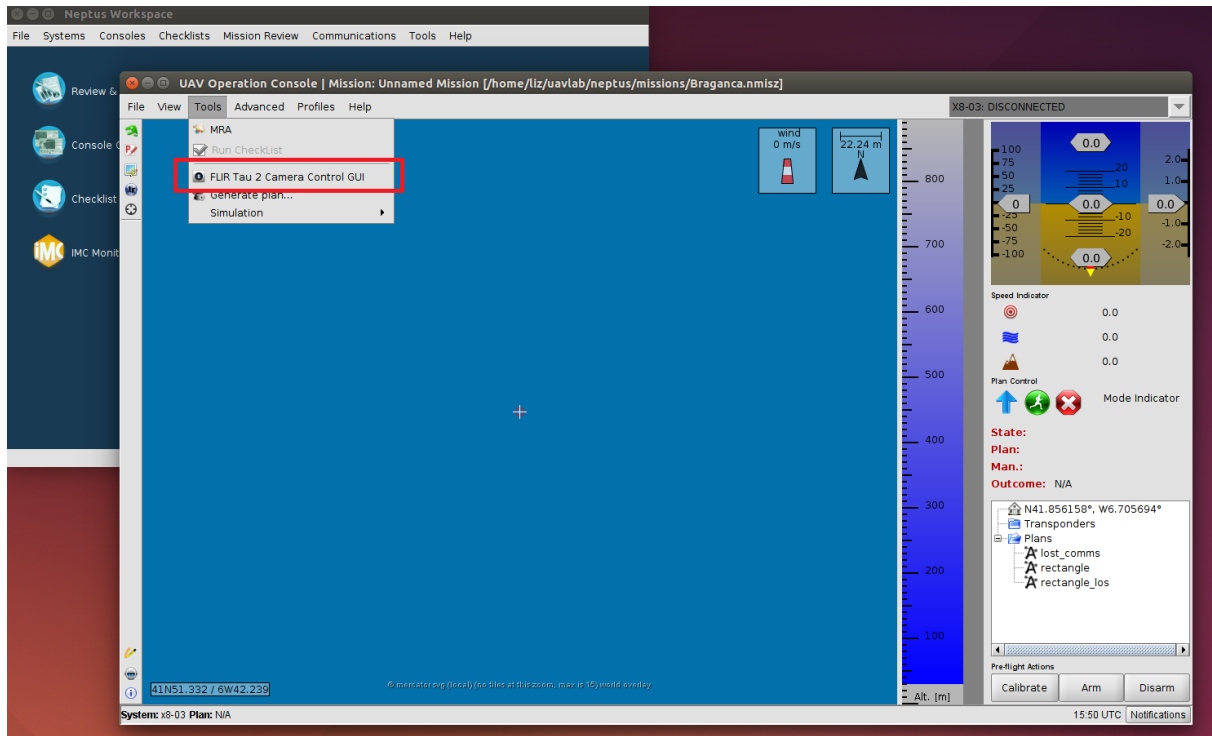


Figure 2.3: Camera control GUI location within Neptus, the ground station software.

Once it is encoded into the IMC message format, it is then sent to the payload onboard the UAV via the radio modem, and reaches the camera via the onboard computer, where a driver task [3] is setup to decode the camera control IMC message and send a command to the camera in its specific message format, including calculation of the CRC values. This is the format indicated in Figure 2.2. This driver task was added to the existing payload computer software, DUNE [5], using Eclipse IDE for C/C++ [15]. An algorithm was also added to the utilities folder available to all tasks, to enable calculating of the CRC-CCITT values necessary for valid camera communication messages (these files can be seen in Appendix D). There are an extensive number of possible camera control commands able to sent to the camera [13], not all of which were implemented in the GUI. All possible command function codes and arguments were however hardcoded into the "ThermalCamFunctionCodes" and "ThermalCamArguments" class files respectively for the purpose of potential future work. Enabling their use would only mean coding the GUI skeleton and connecting its function to the instruction reference in this class. A full list of encoded commands and an indication of those which were implemented can be found in Appendix B.

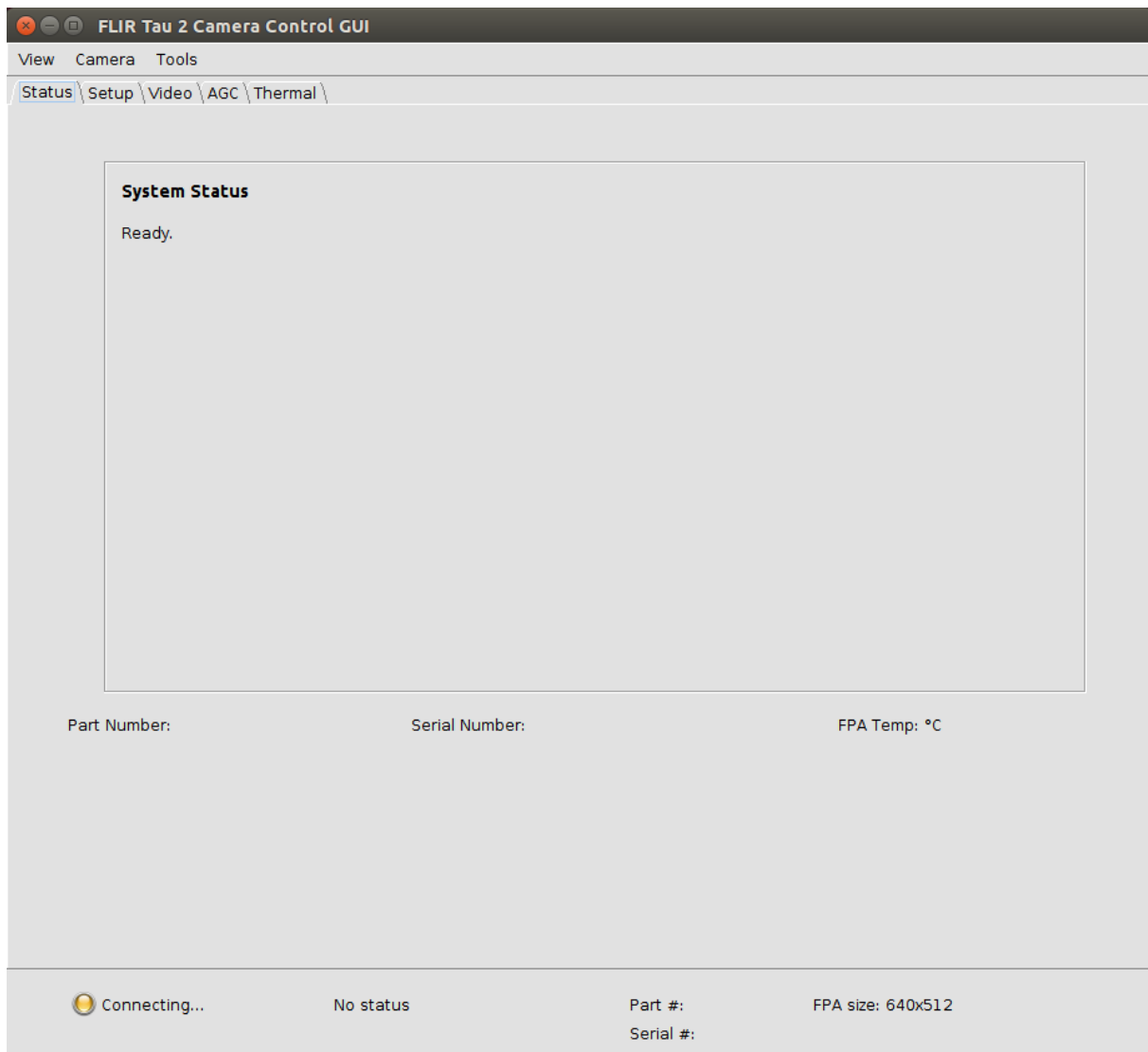


Figure 2.4: Camera Control GUI Status Tab.

2.3 Modular Testing

All of the required components of the camera control interface were successfully implemented. The onboard computer driver was successful in communicating with the camera via a USB connection, using a range of 10 different commands. This was done using the existing VPC Module camera accessory provided from FLIR, therefore verifying the production of the camera control message protocol, and reception of camera replies. The verification of camera replies was done via visual inspection when the driver outputted the received code to a terminal. The replies indicated if an error had occurred in the message format, including the calculation of the CRC values.

The driver was then tested in its ability to receive IMC messages, theoretically from

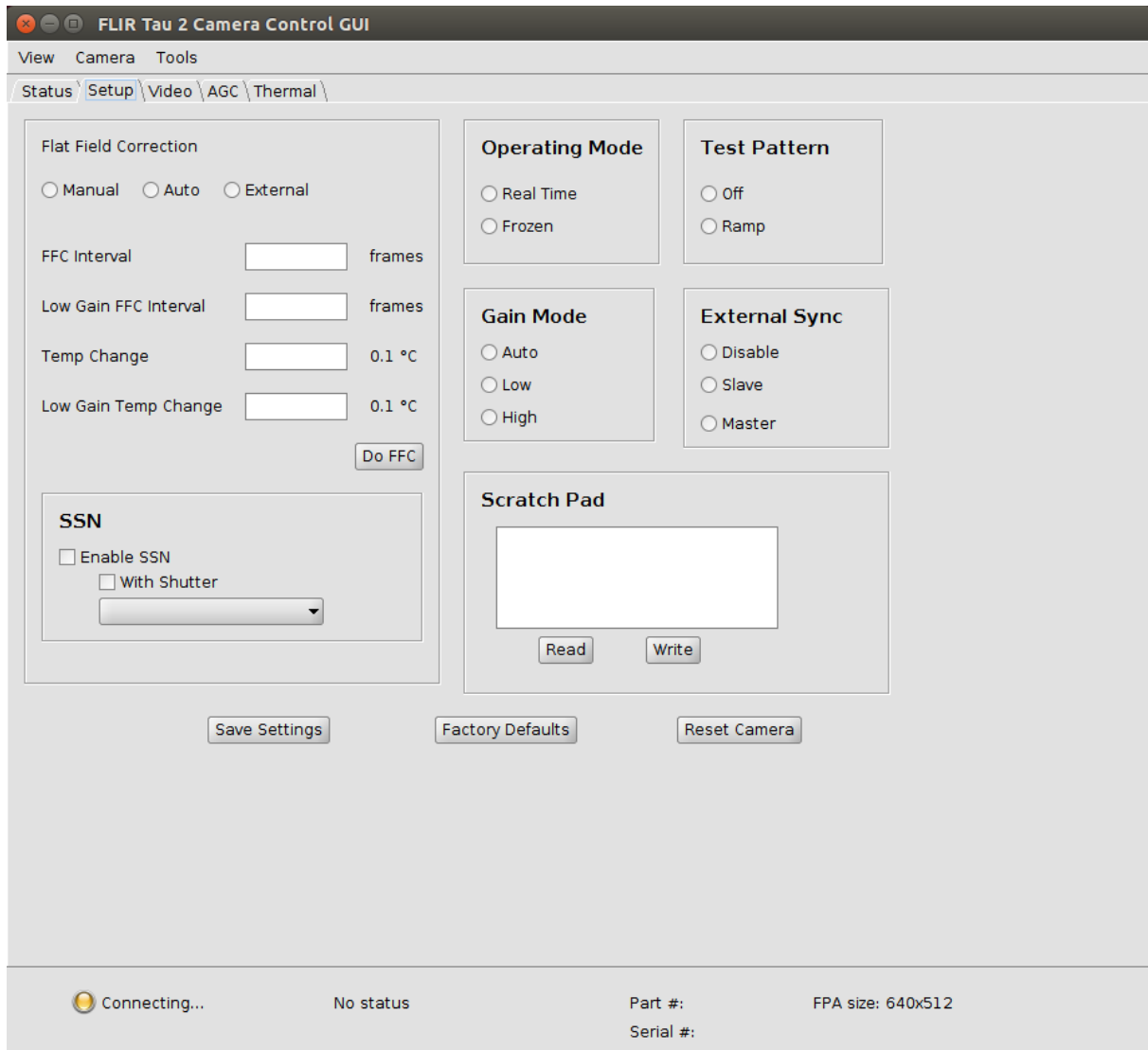


Figure 2.5: Camera Control GUI Setup Tab.

the ground station. This was simulated with a separate task producing IMC messages on the same computer, and having the driver task receive them as if they had come through wirelessly. This eliminated possible errors due to the wireless link, and enabled verification of the driver task's ability to receive and decode camera control IMC messages. This was successful, gauged in the same way as the previous test, using the same range of 10 commands, verified by visual inspection of the camera reply message.

The GUI was also tested in its ability to produce IMC messages when a button is clicked. This was done by visual inspection using a range of 10 different commands, having the messages outputted to a terminal and checked manually. The entire system composed of the GUI, the new IMC message type and the payload computer driver,

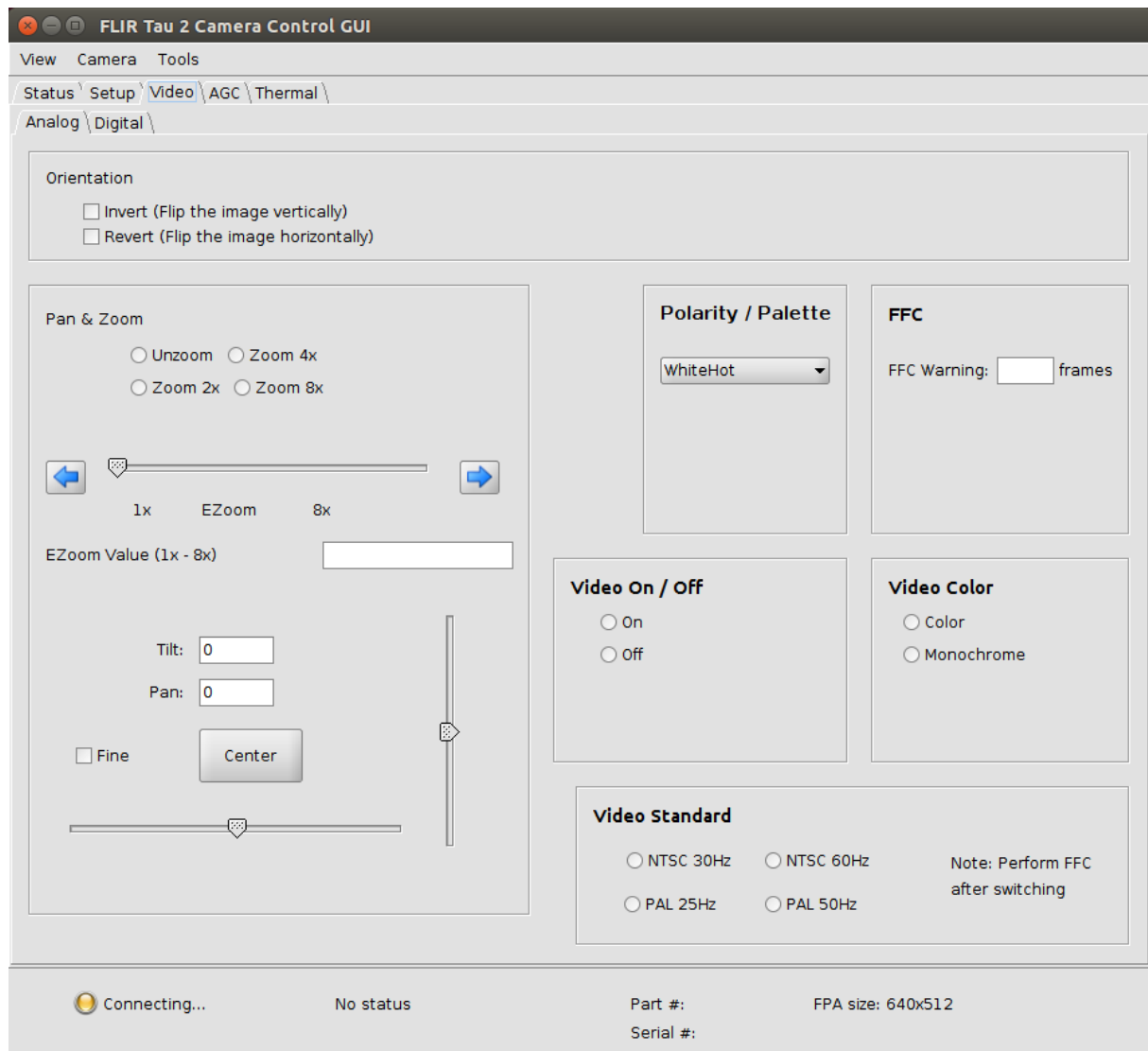


Figure 2.6: Camera Control GUI Analog Video Tab.

was however not tested together, due to there not being a full uav payload system available for testing. The relevant objective however did state that implementation was the goal, and thorough testing was deemed a lower priority than implementation of the digital data retrieval system. Therefore the objective regarding implementation of a camera control interface was successfully completed. Simultaneously, existing system familiarisation was also achieved.

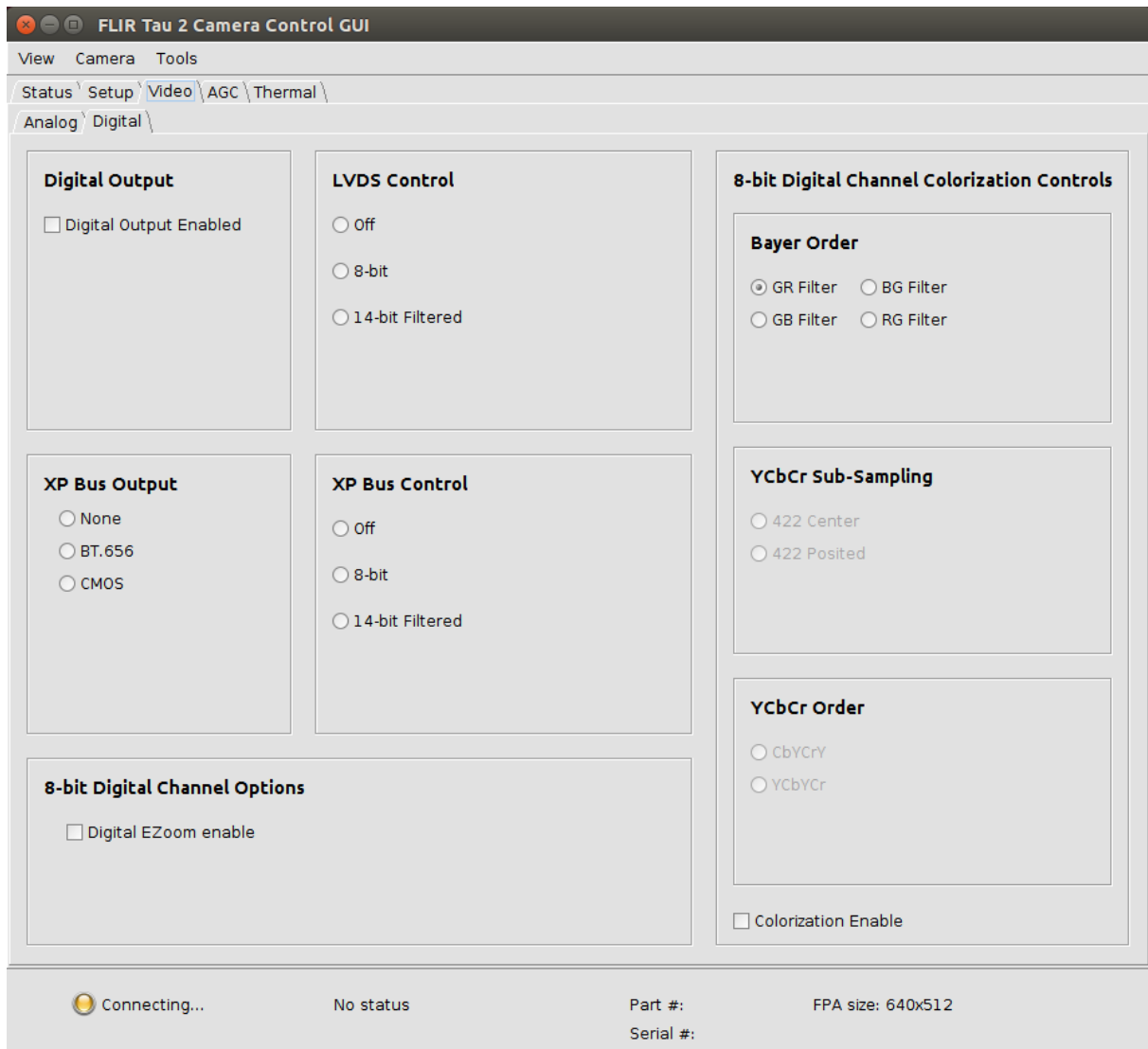


Figure 2.7: Camera Control GUI Digital Video Tab.

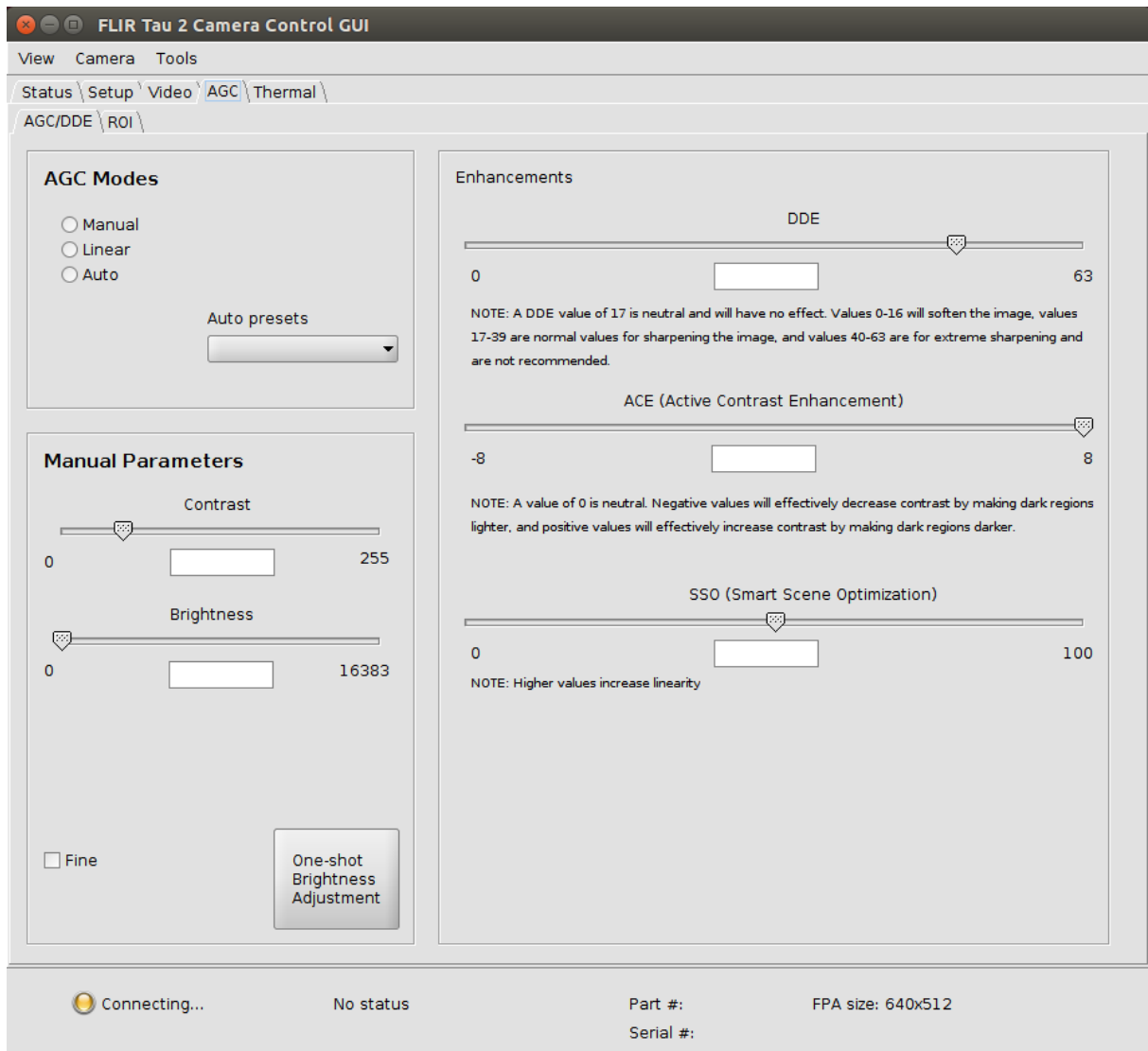


Figure 2.8: Camera Control GUI AGC/DDE Tab.

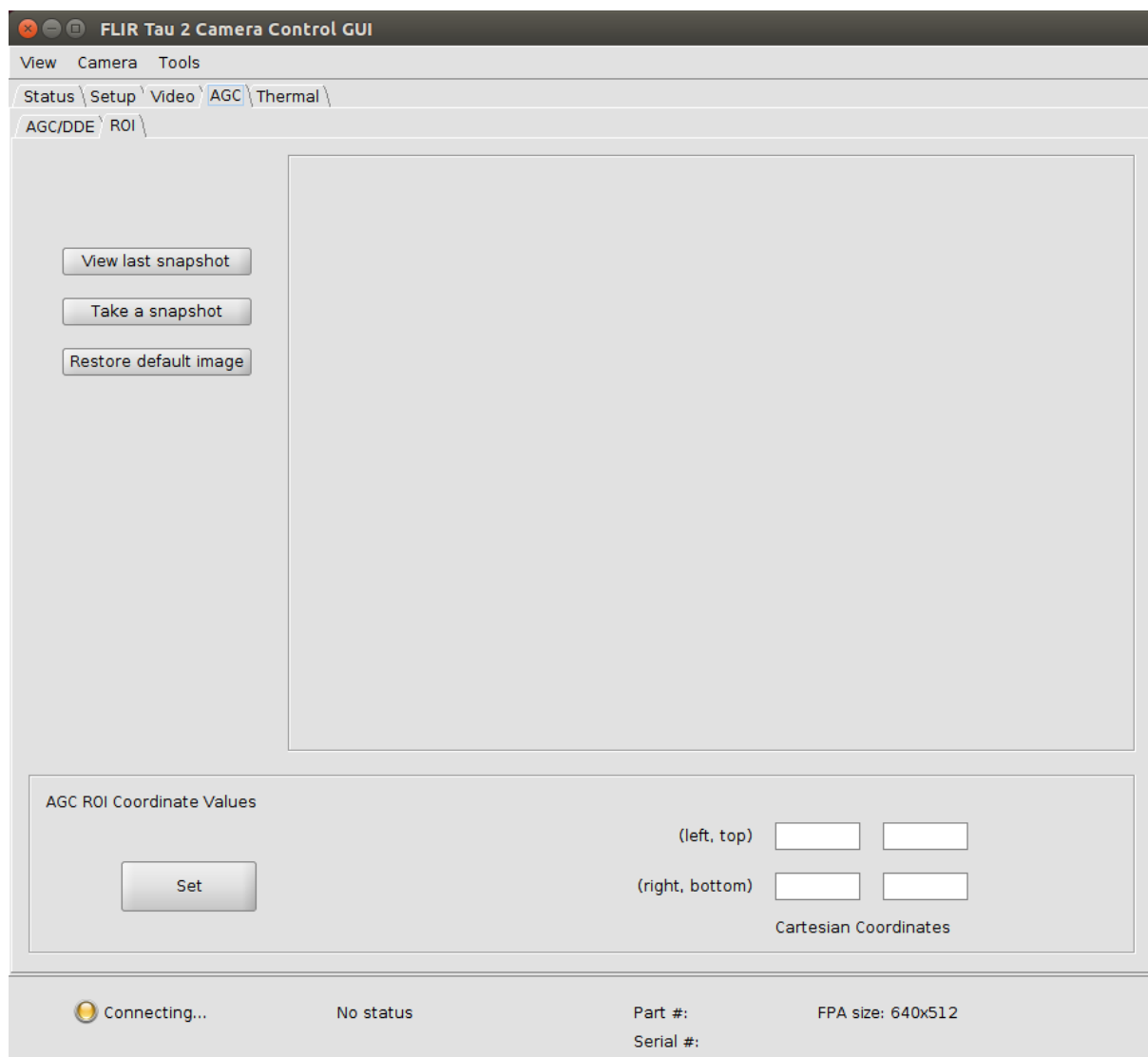


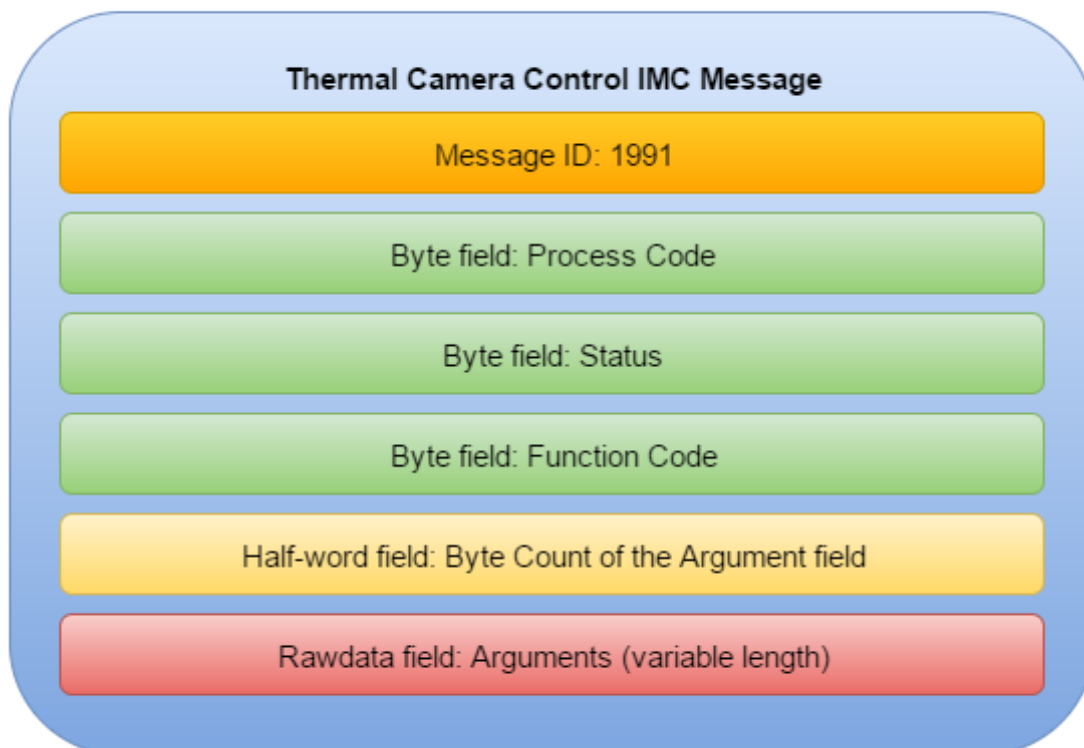
Figure 2.9: Camera Control GUI ROI Tab.

```

- <message id="1991" name="Thermal Camera Control" abbrev="ThermalCamControl">
  - <description>
    Message containing the protocol for thermal camera control.
  </description>
  - <field name="Process code" abbrev="processCode" type="uint8_t">
    - <description>
      Process code. 0x6E for all valid messages to/from thermal cam.
    </description>
  </field>
  - <field name="Status" abbrev="status" type="uint8_t">
    <description> Thermal camera status. </description>
  </field>
  - <field name="Function" abbrev="function" type="uint8_t">
    <description> Function of command. </description>
  </field>
  - <field name="Byte count" abbrev="byteCount" type="uint16_t">
    <description> Byte count for command argument. </description>
  </field>
  - <field name="Arguments" abbrev="args" type="rawdata">
    <description> Byte vector of command arguments. </description>
  </field>
</message>

```

(a) XML code.



(b) Abstract representation.

Figure 2.10: Camera control IMC command format.

Chapter 3

Digital Data Retrieval

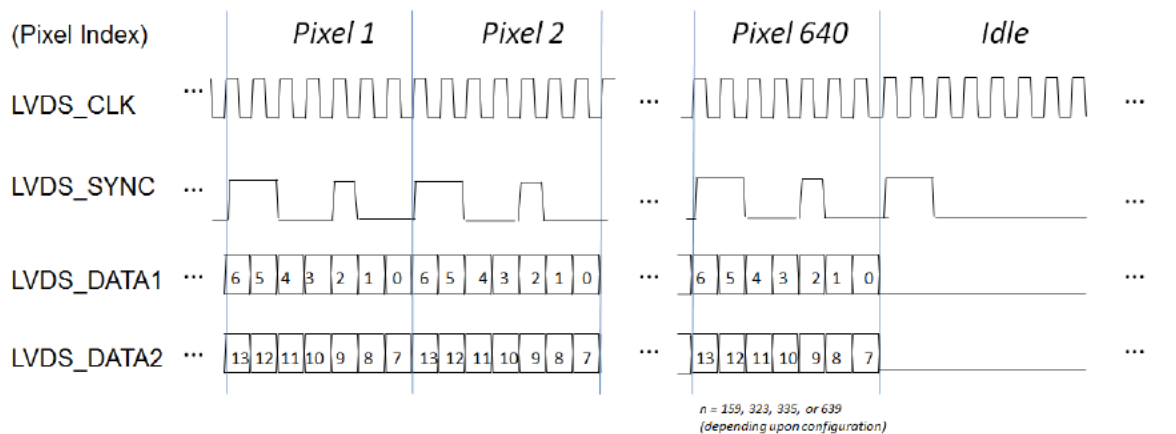
The FLIR Tau 2 thermal camera is capable of outputting several different image data formats [9]. The largest difference factor is between analog and digital data output. The existing UAV payload retrieves image data from the thermal camera in an analog format and uses an analog-to-digital converter in order to transmit it in a digital format to the ground station (and also for video capture on the payload computer). This A/D converter requires a unique 48V power source, and therefore an additional transformer to be part of the payload. Aiming to retrieve digital imaging data directly from the camera instead of converting it externally seemed the most prudent and efficient way of decreasing the payload weight, and hence improving the system's efficiency and cost effectiveness. Implementing this would deem the A/D converter and the transformer providing its power source obsolete, effectively decreasing the payload weight by approximately 300g. This is quite significant, considering the 1000g maximum payload weight.

3.1 Digital Data Formats

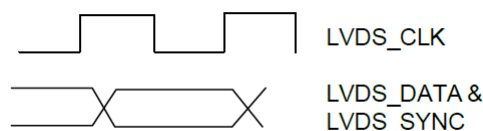
There are many different formats that the thermal camera is capable of outputting, all of which offer different levels of detail, and different modes of colouring. According to the FLIR Tau 2 Hardware IDD, the maximum frame rate is 9 frames per second, achieved when using both 8 bit CMOS and 8 bit LVDS. These were therefore chosen as the two formats to focus on, in an effort to achieve the fastest possible frame rate.

The first of the digital formats under focus is LVDS, or Low Voltage Differential Sig-

nalling format. It is a serial format, comprised of several sets of differential pairs, one pair each for the clock, line valid and two data lines, for a total of 8 lines. The line timing can be seen in Figure 3.1. The valid signal has various sets of bits to indicate validity status; these can be seen in Figure 3.2. LVDS is designed to decrease noise [21], since the resulting signal is retrieved by the difference of the two relevant lines in the differential pair, but also therefore requires more processing than CMOS. The end of the frame is not explicitly signalled, but is implied after the last line has arrived, and therefore the lines must be counted by the receiver. It requires the same number of hardware connections despite the level of detail in the image data, because it is a serial protocol.



(a) Line Timing.



(b) Data timing relative to clock signal.

Figure 3.1: LVDS line timing. Adapted from [9].

The second of the digital formats being considered is CMOS, or Complementary Metal Oxide Semiconductor format. It is a parallel format, requiring one clock signal, two valid signals (line and frame), and one data line for each bit (e.g. 8 bit CMOS format would need 8 data lines), totalling 11 lines for 8 bit CMOS. The line and frame timing for the CMOS format is depicted in Figure 3.3. Both the end of the line and the end of the frame is explicitly indicated by their respective valid signals, and there is less processing by the receiver, given that there is no differential to calculate. It

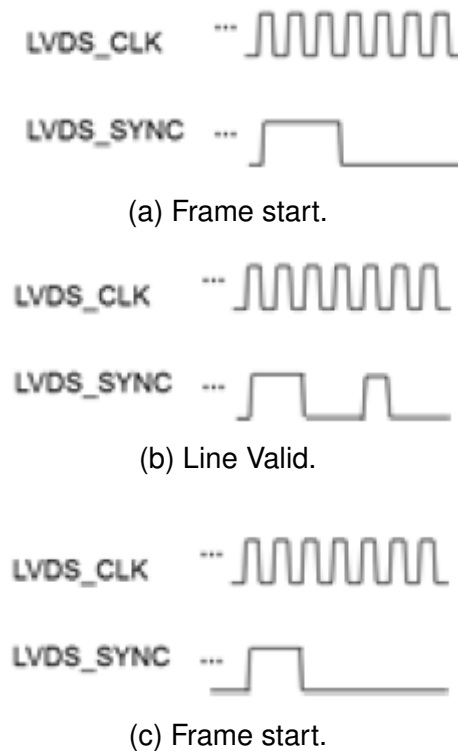


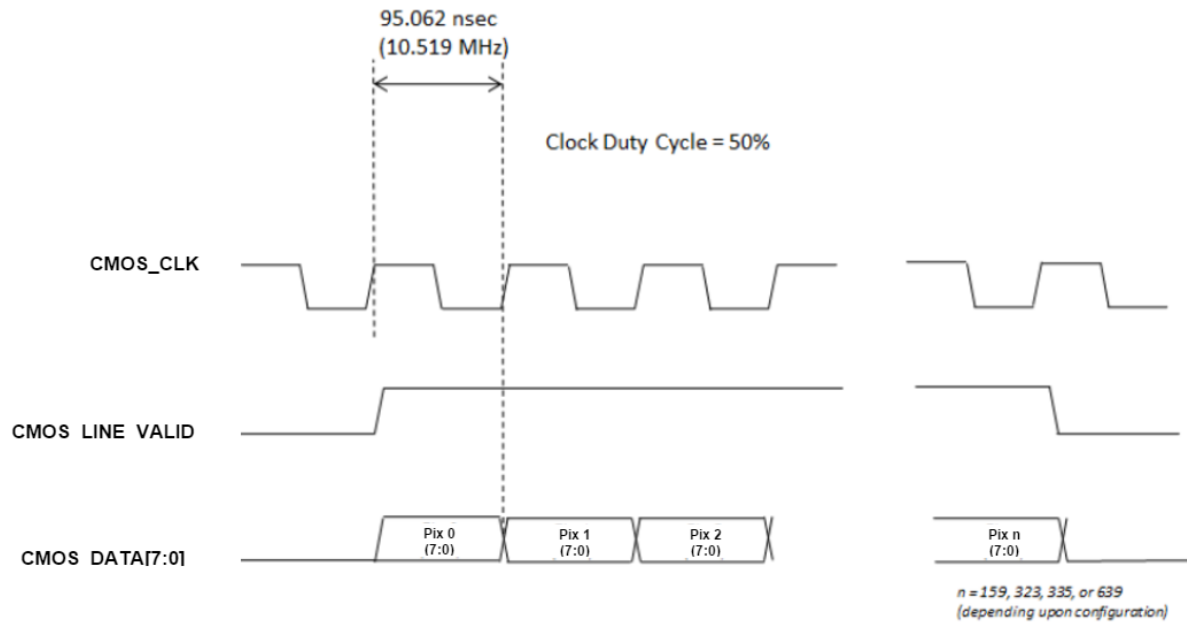
Figure 3.2: LVDS valid bit sets [9].

does however require more hardware lines to be connected, increasing with the level of detail in the image data.

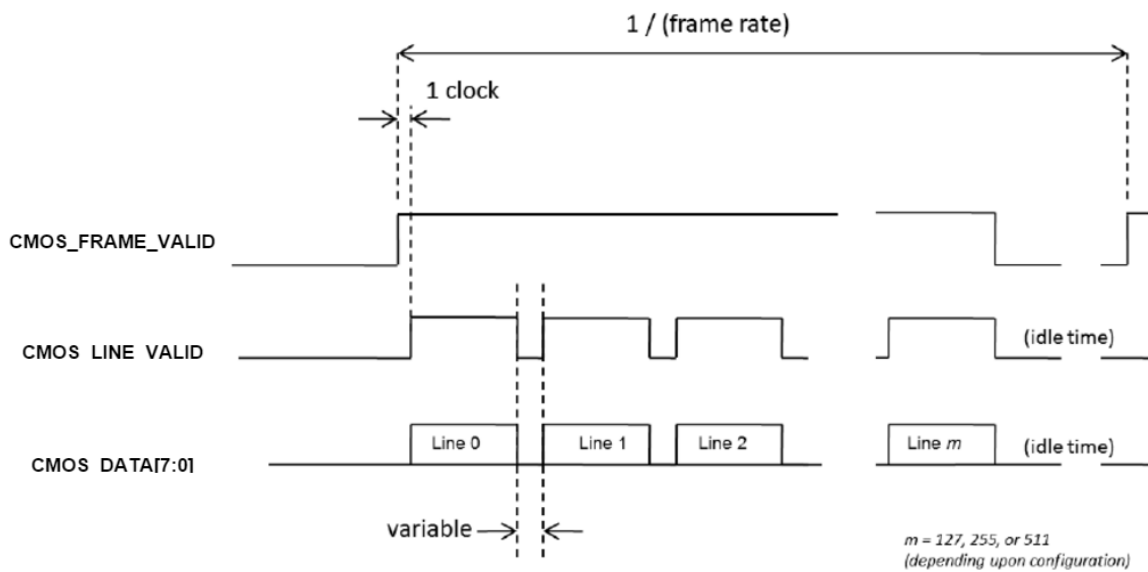
3.2 Hardware Options for Retrieval

One camera accessory provided by FLIR is the PCB Wearsaver, depicted in Figure 3.4, which allows access to the LVDS data via solder pads [11]. The logic level of LVDS signals (max 350mV [9]) is much lower than the level required to be received on the payload computer GPIO (max 1.8V) [44], and would therefore require an additional PCB containing an LVDS receiver chip to be effective. It also does not allow access to the external sync function of the camera, or the CMOS digital data [52], and was therefore deemed insufficient for the application, in that creation of a custom PCB could in theory allow access to all desired signals.

Another camera accessory available from FLIR which allows access to both of the digital data formats to be accessed is the Camera Link Expansion Board [10], depicted in Figure 3.5. It however is not deemed easy-to-use as it uses a camera link connector, which is not able to be connected to the existing onboard camera directly, and it



(a) CMOS line timing.



Note: Figure is not to scale.

(b) CMOS frame timing.

Figure 3.3: CMOS timing diagrams. Adapted from [9].

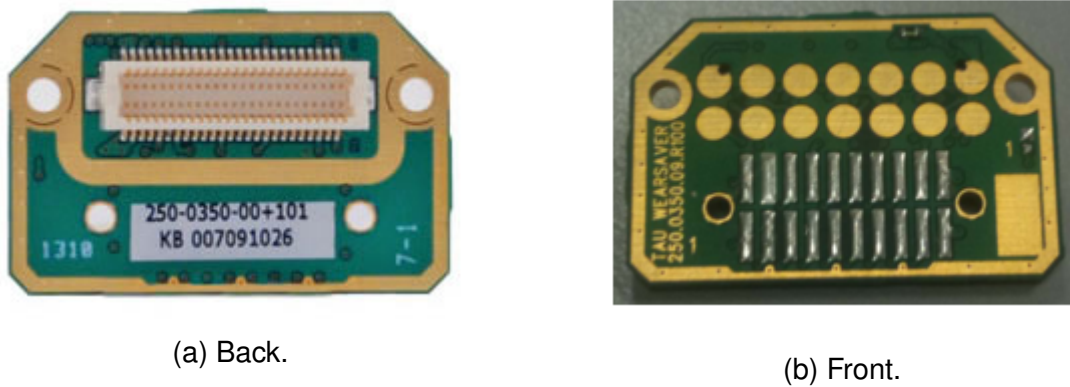


Figure 3.4: FLIR Tau PCB Wearsaver with Solder Pads.



Figure 3.5: FLIR Camera Link Expansion Board.

requires an additional USB connection to provide power and transmission of camera control signals. It does not allow access to the external sync function of the camera without adaptation, and according to what little documentation can be found, it can only be used with the provided ground station software, which is only functional in Windows OS (which becomes problematic in that the payload computer runs Linux OS). Considering also the cost of this accessory is 1500USD (which will affect end user cost), and it is not exactly what is wanted for the application, it was decided to attempt the production of a custom PCB, enabling access to what features are desired, compatibility with the payload computer software, and ease-of-use with a single USB connection.

Both of the considered digital formats are included in the first prototype of the custom PCB, in order to gauge the benefits of each one during initial testing. It is hy-

pothesized that the benefit of the decreased processing of CMOS data will outweigh the benefits of decreased noise in LVDS considering the importance of achieving the fastest possible frame rate, however it cannot be confirmed without proper testing. The presence of both will not affect the performance of either, as only one will be active at any time.

3.3 Custom PCB Functionality

There were effectively three options with regards to where to perform the digital data processing, including the ground station, the payload computer, and potentially the custom PCB. With the former two options, there would be issues with the GPIO logic level similar to with the Wearsaver option, and also with line timing, as the correct reception of the data requires a very specific order of arrival. Since there are so many signals which need to be the same length for both digital data formats, it was decided to include an MCU on the custom PCB. This allows for the processing of the image data to be done as close to the camera as possible, keeping the lines which need to be matched in length as short as possible. The programming of the MCU can be changed via the debug header, which was also included in the PCB design due to it being the first prototype. Communication with the onboard computer is via a micro-USB connection, which also provides power. An abstract representation of the custom PCB functionality can be seen in Figure [3.6](#).

3.4 Hardware Design

The custom PCB acts as a USB device, with a micro-USB connector feeding directly into a 32 bit microcontroller. A 60 pin Hirose connector retrieves the necessary signals from the camera. Two sets of digital data lines are connected to the MCU, one set of 11 lines for 8 bit CMOS, directly from the camera, and another set of four lines for LVDS, relayed from an LVDS receiver, which differentiates and amplifies the differential pairs from the camera. The layout is also designed to enable the MCU to act as a relay for communication messages, with a UART connection to the camera. There are several MCU peripherals present, in conjunction with support documentation [\[23\]](#),

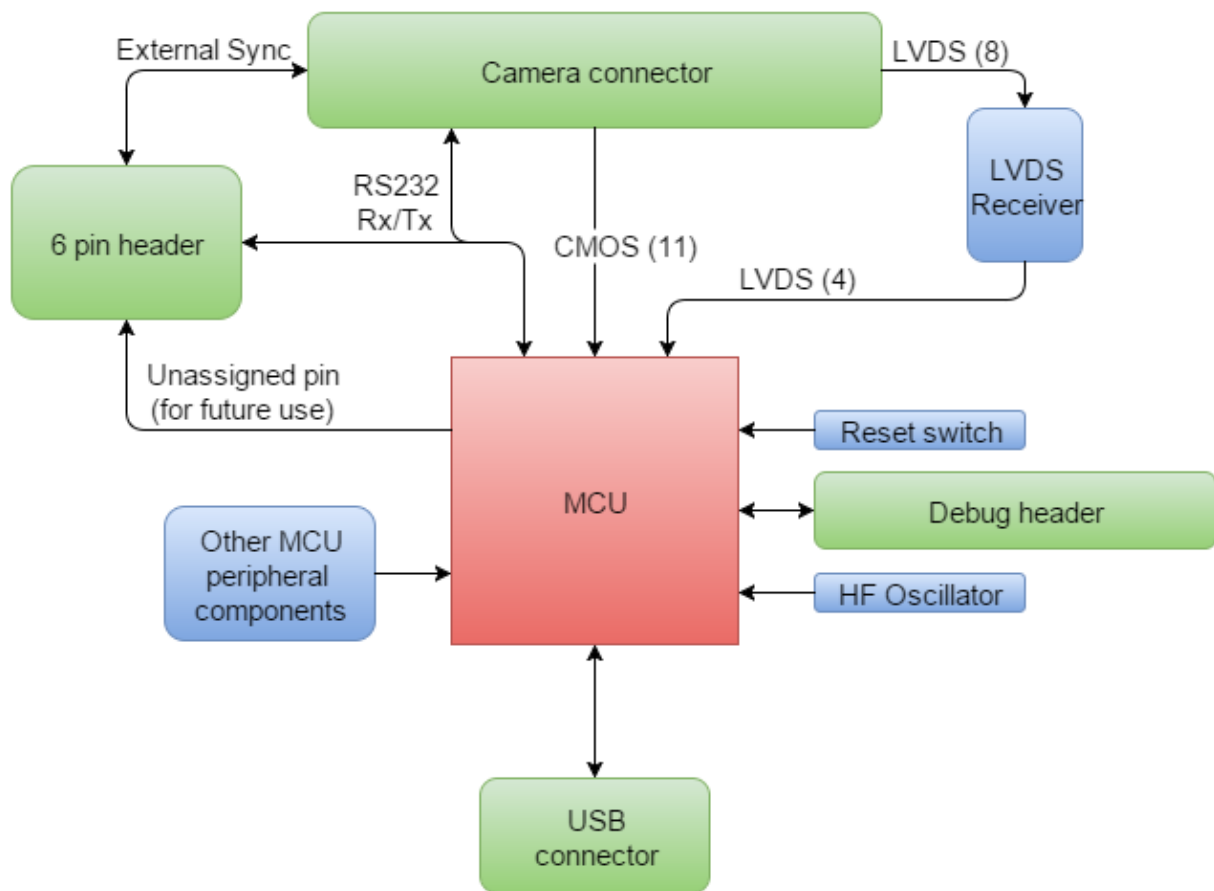


Figure 3.6: Abstract depiction of custom PCB functionality.

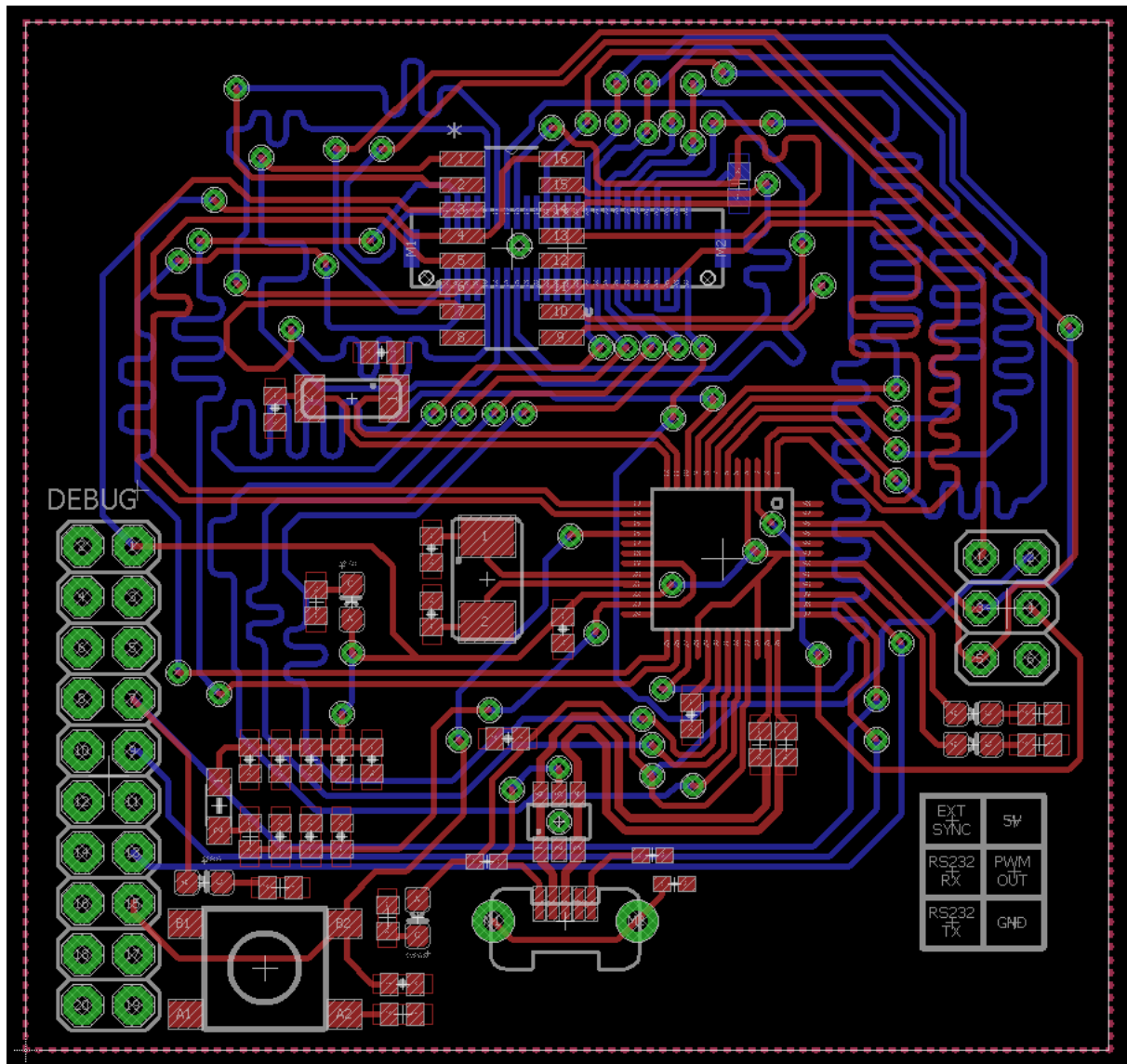


Figure 3.7: The final layout design of the custom PCB.

[29, 36] including components necessary for power regulation, two crystal oscillators, a debug header, a reset switch and two user LEDs. There are also LEDs to indicate 5V power, 3.3V power, and a debug connection, inclusion of which were inspired by the MCU starter kit [33]. There is also a 6-pin header included which provides direct access to some functions of the custom PCB. The final layout design can be seen in Figure 3.7.

Many elements are included on the first prototype hardware design due to the early developmental nature of the project. Especially due to the inexperience of the designer, many components included in this design may become obsolete in later prototypes. Some examples are the debug header, the user LEDs and the direct access to the

camera's serial interface via the 6 pin header. The debug header enables flashing of the MCU on the custom PCB with firmware, and allows for this code to be debugged by stepping through individual instructions and providing visual indication of the current instruction being executed. Connection to this header will be possible using a J-Link debug probe [48]. In the event that the debug header is no longer needed in a future prototype, it can be removed and the MCU can be flashed using the UART bootloader [31]. There were also two LEDs included in the hardware design, intended for use in debugging during program development. They are connected to two GPIO pins of the MCU, PF4 and PF5, as can be seen in the PCB schematic in Appendix C. Inclusion of the 6-pin header is to enable direct access to some functions of the custom PCB. The first being the camera serial interface, in the event that the MCU CPU is incapable of handling both reception of the image data and the camera control commands. In the event that this is the case, the direct access via the 6-pin header can be connected to a UART interface on the payload computer GPIO instead, allowing the instructions to bypass the PCB MCU, leaving it available for image processing. Also included in the 6 pin header is a 5V power pin (and a ground pin), intended for use in the event that defining the custom PCB as a bus-powered USB device becomes problematic. The external sync pin of the camera connector is also connected to this 6 pin header, allowing direct access in accordance with the objective requirement. As previously mentioned, there is also an undefined GPIO pin connected to the 6-pin header, intended for future use, for example as a PWM output for camera gimbal control. These elements were included in the spirit of "just in case". While the documentation for all components was extensively referred to during the design process, inexperience leading to human error is still definitely a possibility. These extra elements are useful in determining what is wrong in the event that the PCB is not functional, and also during firmware and software development if it is.

Component selection was performed while keeping in mind several factors. The MCU was chosen as a Silicon Labs Happy Gecko 32 bit MCU [35], due to the immediate availability of a corresponding development kit, and the experience of the programmer in using the brand. This meant that upon completion of the hardware design, there was no need to wait for the completion of manufacturing in order to start the software and firmware development. The connector for retrieval of signals from the camera was

selected to be the same as the FLIR PCB Wearsaver [18], and therefore implicitly recommended by FLIR. The chosen USB connector was used due to its vertical direction [51] (the plug direction is perpendicular to the PCB surface), and therefore enabled easier access when mounted inside the gimbal. The reset switch mechanism [43] was chosen to be as similar as possible to the switches used on the starter kit [33]. Both the high and low frequency oscillators [1, 2] were chosen after consulting the MCU Oscillator Design Considerations document [23], in which they are recommended. The ESD Solution component [19] was chosen for its similarity to the one in the schematic example given in the MCU USB Hardware Design Guide [24]. The LVDS receiver [20] was chosen for its four channels, the number needed for the LVDS configuration of the FLIR Tau 2, its small package and its requirement for a 3.3V power source, able to be outputted by the camera [9]. All other components (ferrite beads, headers, LEDs, resistors, capacitors etc) were chosen for small package, voltage rating, UK stock availability and price using the component selection tool on the Farnell website. A bill of materials can be found in Appendix B, along with the PCB schematic and layout.

The schematic and layout were designed in CadSoft EAGLE (Easy Applicable Graphical Layout Editor) [4], using a freeware license. The PCB design software uses device packages stored in libraries which enables correlation between the schematic symbol and the PCB footprint, viewed in the Schematic and Board views respectively. For some of the selected components, a device package needed to be manually designed. This involves drawing the schematic symbol, laying out the footprint, and storing both in a package with a definition of which pin (on the schematic symbol) corresponds to which pad (on the board footprint) [50]. This was necessary for the MCU, the camera connector, the USB connector (non-standard due to its vertical direction), the ESD protection device, and LVDS receiver. For the other components, existing or Farnell-provided library packages were able to be used. The manually implemented library package for the EFM32 microprocessor can be seen in Figure 3.8 as an example. All manually defined library packages are included in Appendix D.

Difficulty was involved in the PCB layout design, specifically with regards to matching the length of several sets of signals. It was necessary to have a 90 ohm differential pair for the USB connector [24], as well as differential pairs for the two crystal oscillators [23]. Additionally, it was necessary to match the lengths of the digital data lines

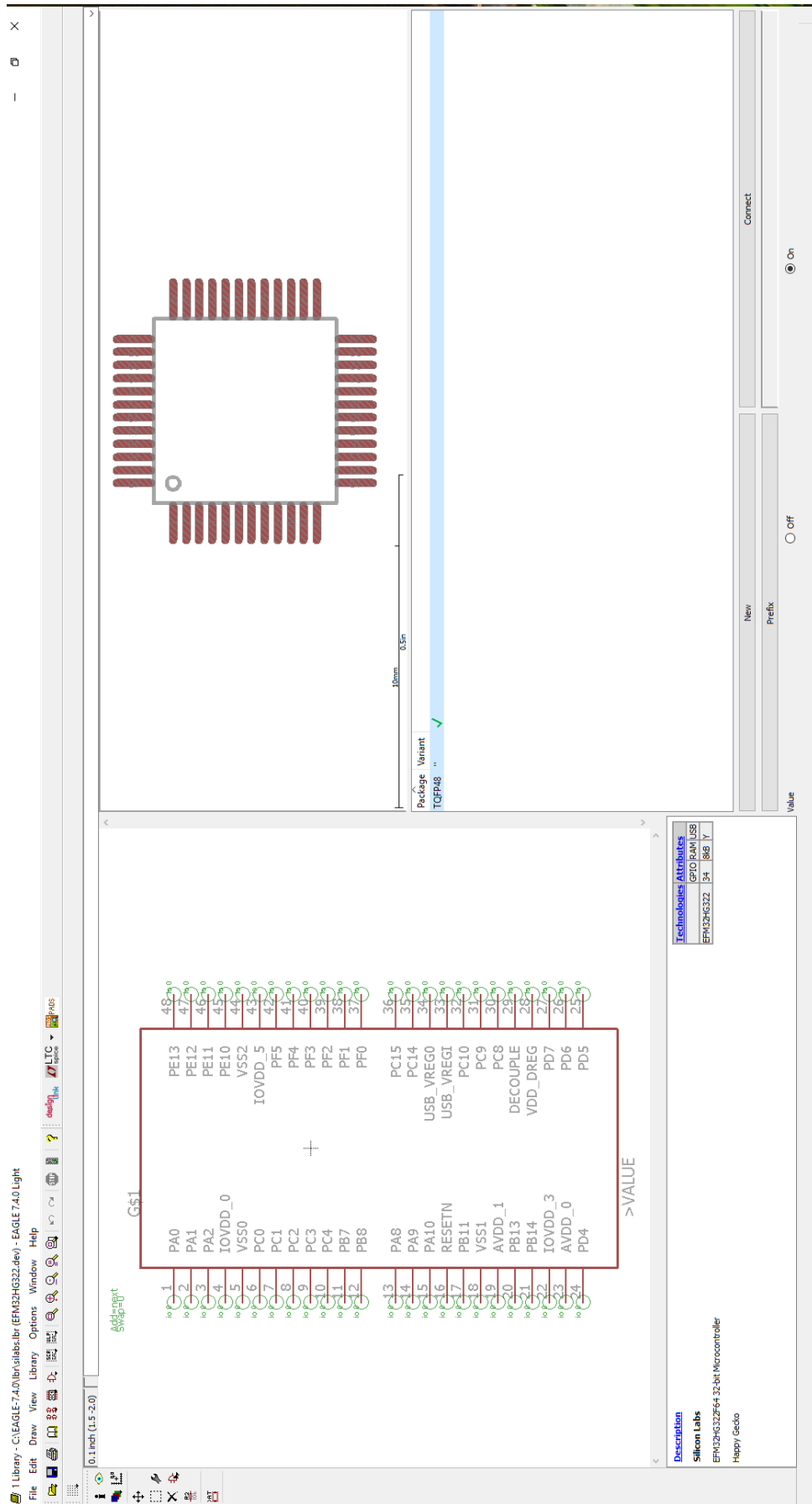
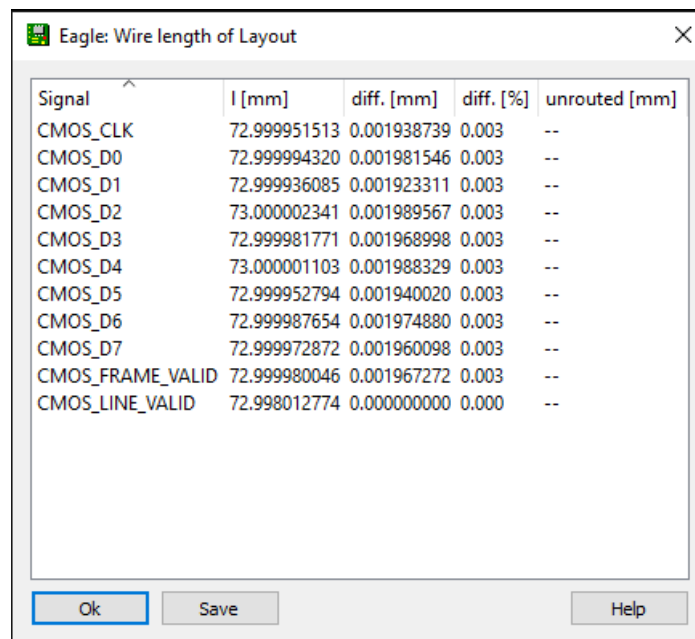


Figure 3.8: EFM32 Microprocessor manually defined library package.

in order to enable correct timing functionality, as detailed in Section 3.1. Meandering methods usually used for impedance matching [49] were instead used for matching length for the purpose of timing, necessary for the 11 CMOS signals from the camera to the MCU, the 8 LVDS signals from the camera to the LVDS receiver, and the 4 signals from the LVDS receiver to the MCU. Due to the high rate at which the digital image data is transmitted from the camera, the length of the traces were matched as accurately as possible, in order to maximise the time available for the valid data port value to be captured. The maximum difference in CMOS signals is less than 0.002mm, LVDS pre-receiver 1.016mm, and LVDS post-receiver 1.016mm. The length matching was executed using the meander function of the PCB layout software [4], and checked using the 'run length' command as demonstrated in Figure 3.9 with the CMOS signals. The CMOS signal lines on the PCB layout are highlighted in Figure 3.10. It was attempted to keep most of the meandering of signals on the bottom layer (blue), for the purpose of aesthetics.



Signal	l [mm]	diff. [mm]	diff. [%]	unrouted [mm]
CMOS_CLK	72.999951513	0.001938739	0.003	--
CMOS_D0	72.999994320	0.001981546	0.003	--
CMOS_D1	72.999936085	0.001923311	0.003	--
CMOS_D2	73.000002341	0.001989567	0.003	--
CMOS_D3	72.999981771	0.001968998	0.003	--
CMOS_D4	73.000001103	0.001988329	0.003	--
CMOS_D5	72.999952794	0.001940020	0.003	--
CMOS_D6	72.999987654	0.001974880	0.003	--
CMOS_D7	72.999972872	0.001960098	0.003	--
CMOS_FRAME_VALID	72.999980046	0.001967272	0.003	--
CMOS_LINE_VALID	72.998012774	0.000000000	0.000	--

Figure 3.9: Length match check figures for the CMOS signals in the EAGLE PCB layout software.

The final custom PCB design is a two-sided single-layer board of the dimensions 45mm width by 54mm high, which is approximately the same size as the back of the camera. The individual sides of the layout design can be seen in Appendix C, and pictures of the manufactured board can be seen in Figure 3.11, albeit without com-

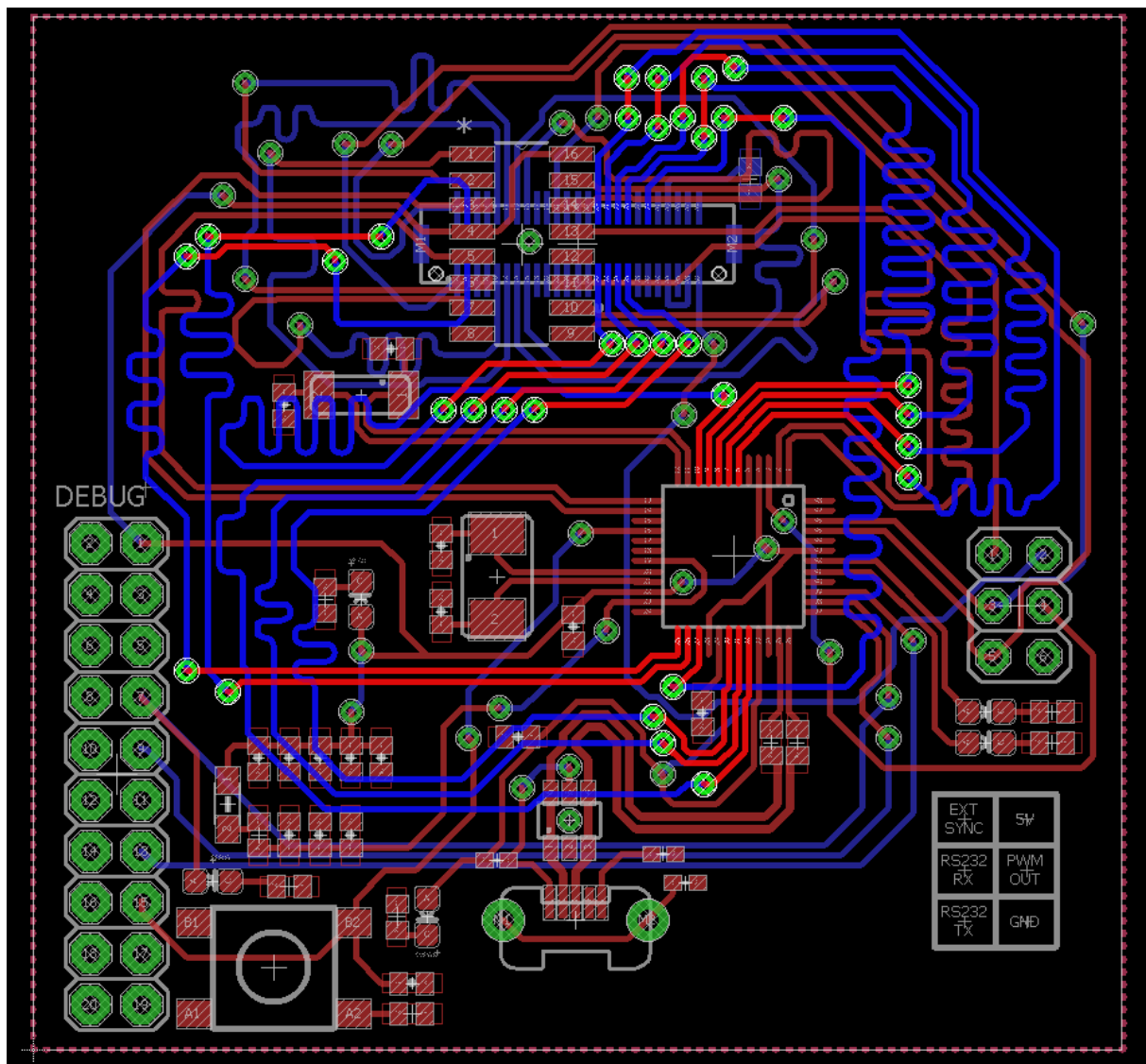


Figure 3.10: CMOS signal lines highlighted in the PCB layout.

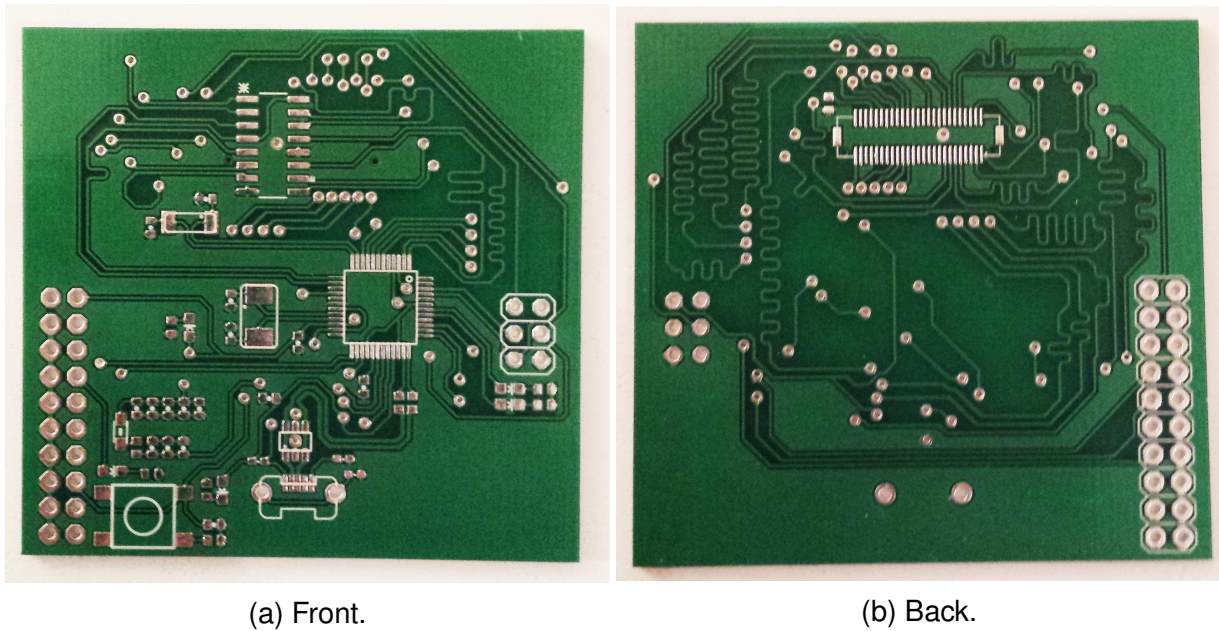


Figure 3.11: Manufactured custom PCB without components placed.

ponent placement completed. The PCB layout software's internal error checking tools were used to verify the viability of the design. This included an Electrical Rules Check (ERC), which pertains to general electrical rules relevant to the parts that are present in the design, such as that all components are powered and the necessary pins are connected, and the Design Rule Check (DRC), which verifies that a set of rules pertaining to the layout are kept, such as clearance and access to the ground plane.

3.5 Firmware and Software Design

Despite the large amount of effort involved in designing the custom PCB hardware, it is essentially useless without programming of the central MCU component, as well as a driver program to handle interfacing between the PCB and the payload computer. While it is crucial, it is also not possible to start development until the hardware design has been finalised. Critically, a development kit for the chosen MCU was already attained, and therefore development for the MCU firmware could begin before manufacture of the hardware was completed. The MCU program is designed to function as depicted in Figure 3.12, in correspondence with the finalised hardware design in order to retrieve the digital data from the camera.

In the first prototype of the MCU program, the digital CMOS data is retrieved via

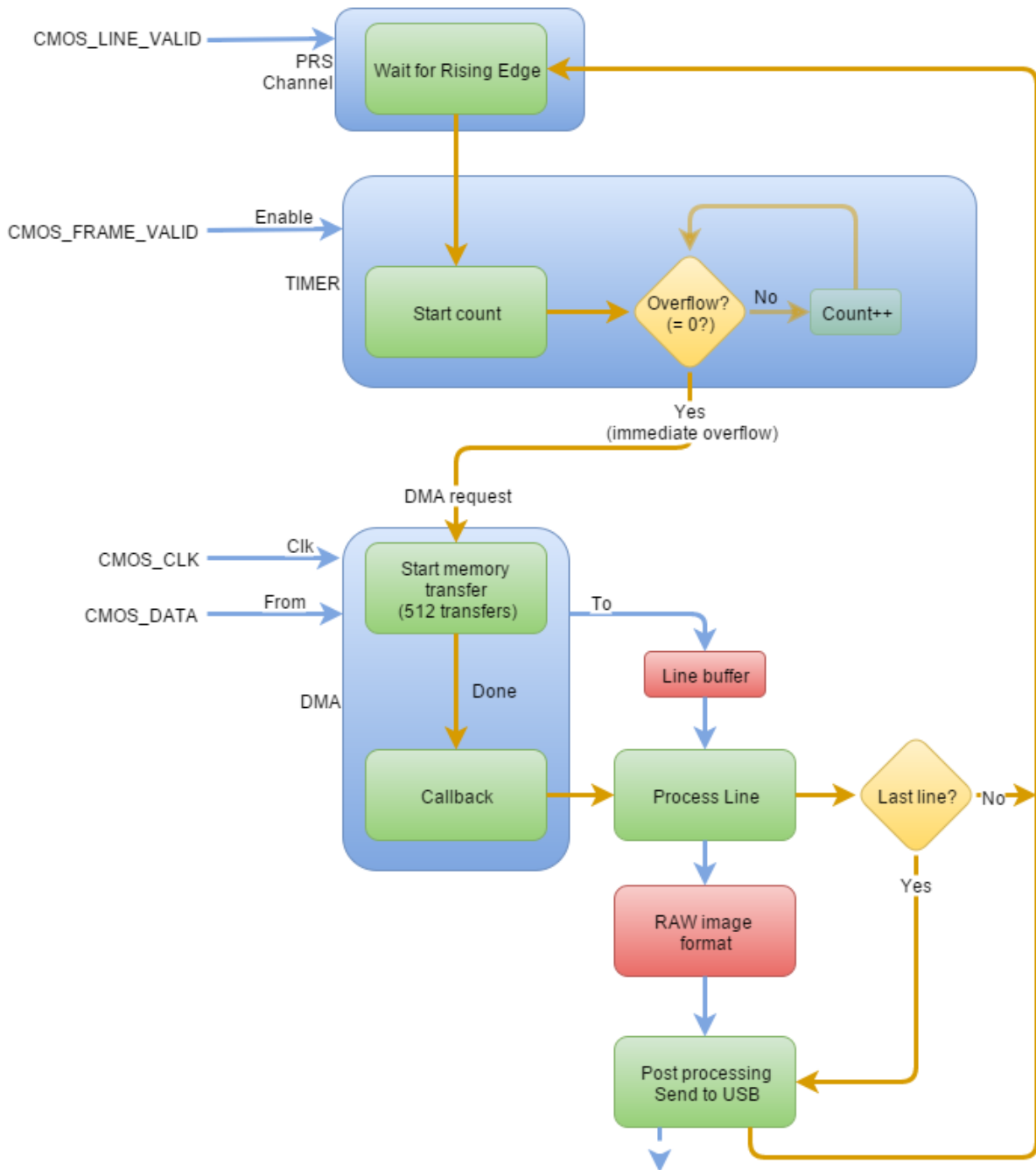


Figure 3.12: First prototype functionality of the MCU program on the custom PCB.

a set of GPIO inputs, using the input clock and the line and frame valid signals for timing. This data is then rearranged into a RAW image format (an uncompressed but organized format of the image, using values indicating pixel value directly from the camera), in a line-by-line fashion. This image is then transmitted to the USB to the payload computer. There is significant difficulty involved in design of the firmware responsible for retrieving the CMOS data. The incoming data is clocked at 10.519MHz [9], and with a 48MHz oscillator the CPU is not capable of handling receipt of the incoming data using software alone, since there is less than 5 clock cycles available to capture the CMOS data port's value and a software interrupt takes 12 clock cycles to begin action [28]. Instead, the approach is hardware assisted, using the MCU's built-in peripheral reflex system, direct memory access module and a timer, the libraries for which are all easily accessible when using the MCU manufacturer's IDE, Simplicity Studio [37]. In using this hardware to assist in the data retrieval process, the only CPU time needed for retrieval of the data is the initial setup time on startup, and a short callback after a full line of data has been retrieved by the DMA module, in order to keep track of the number of lines read. This is in accordance with the CMOS line timing in Figure 3.3. There is also a function called on the falling edge of the frame valid signal, which handles transmission of the RAW format image via USB.

The MCU program uses several of the available hardware peripherals, which are all designed to decrease required CPU time, allowing it to either do other work or go to sleep and save power. In this case, they are useful in that they reduce the time needed to retrieve the digital image data from the camera. Firstly, the Peripheral Reflex System (PRS) allows different MCU peripheral modules communicate directly with one another without CPU involvement [27]. In the MCU program design, a PRS channel is used to route the CMOS_LINE_VALID signal from GPIO to a TIMER as input. The TIMER hardware peripheral has many applications [26], however in this instance it is set to request a DMA transfer on overflow. The overflow count value is set to zero, thereby immediately requesting a transfer upon a rising edge on the PRS channel. The TIMER is enabled with the CMOS_FRAME_VALID signal in order to only enable data retrieval when the frame is valid. The Direct Memory Access (DMA) module, is used for direct data transfer without need for the CPU [25]. It is set up to transfer directly from the GPIO port [30] data in register, where the CMOS_DATA values can be captured.

A DMA transfer cannot be started directly from GPIO, hence the need for the PRS channel and TIMER. The DMA is set up to receive one line of the frame in one request, while being clocked [22] with CMOS_CLK for synchronisation purposes. After this time the callback is executed and the relevant data processing is done by the CPU. In the instance that the CPU is still processing the data when the next line is being received, there is not a problem because receiving the incoming data is completely handled in hardware. At the end of the frame, the uncompressed image is sent via USB.

The USB protocol is very specifically defined, with descriptors necessary for the device, configuration, interface and endpoints [54]. The descriptor heirarchy as defined in the MCU program is depicted in Figure 3.13. As it is a first prototype, the device was defined as vendor-unique, while using the vendor ID of Silicon Labs (the manufacturer of the MCU). This is sufficient until such a time that the device is operational and more widely available, when a vendor ID and license should be acquired from USB-IF, after which a product ID can be defined. This information is contained in the device descriptor. The configuration descriptor contains information pertaining to the device's power source and the number of interfaces that are present. The chosen MCU only supports a single-interface, and has been set up to be bus-powered (all necessary power for the custom PCB and the camera comes from the USB). The interface descriptor defines the number of endpoints, which "are the entities on USB devices that facilitate communication" [32]. It is possible to specify a standard interface class (assigned by USB-IF) which defines the required endpoints, decreasing the need for a custom driver [55]. In this case, the standard interface class for video streaming devices (0x10) [6] was used in order to decrease the necessary complexity of the driver on the payload computer. Finally, the endpoints have their own descriptors, containing information about direction of the transfer and its type. In this case, there are two endpoints. The first, EP0 is present in all USB devices and is used for configuration and enumeration of the USB device ('control' type), as outlined in the USB specification [54]. It is bidirectional (both IN and OUT). Secondly, EP1 has been defined in this device as a 'bulk' type, unidirectional OUT pipe, to handle transmission of the camera image data. This type of pipe is intended for large amounts of data without regard for latency. This type was chosen due to the large amount of data to transfer but should be changed to an 'interrupt' type (which considers limited latency a priority) if latency from transmission becomes

a problem during thorough testing.

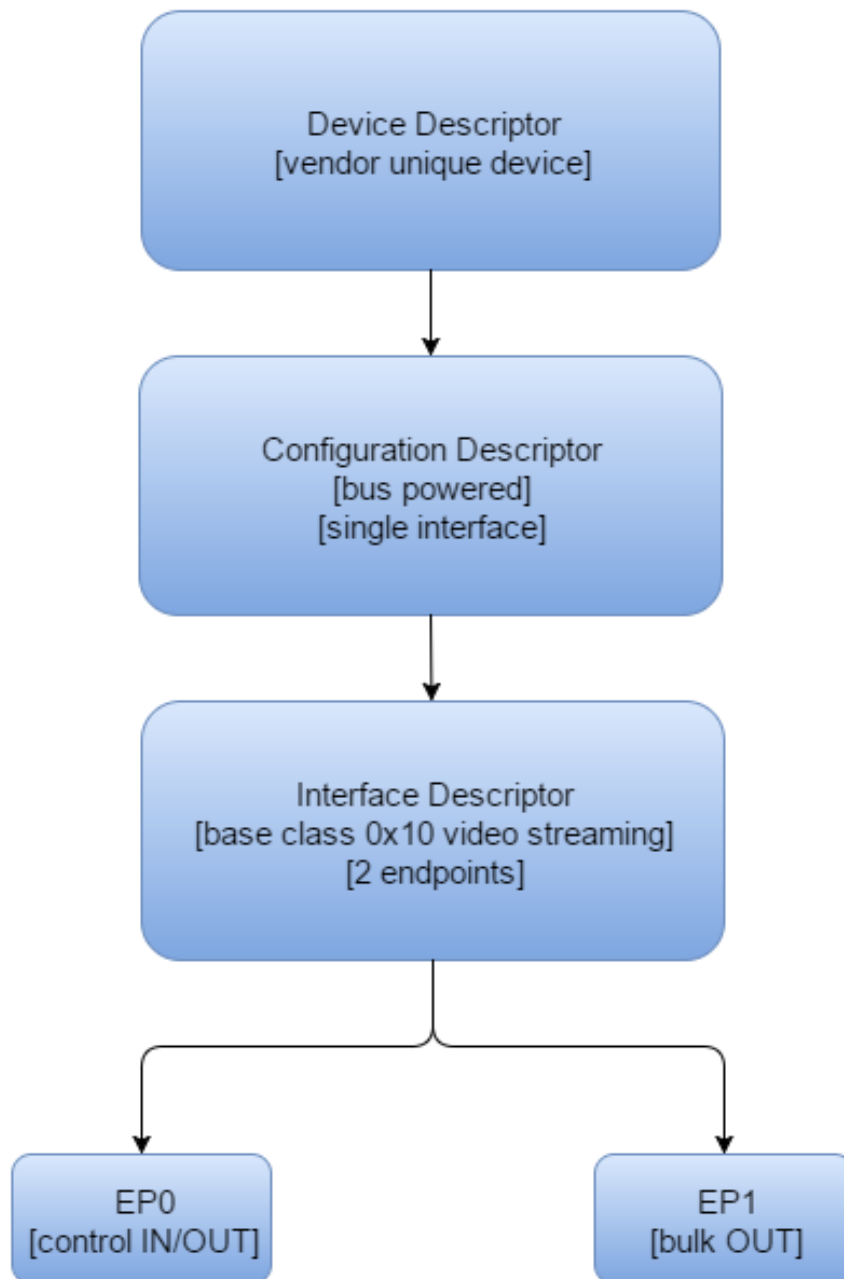


Figure 3.13: USB descriptor heirarchy.

During development of the MCU program, it was decided to focus firstly on the retrieval of the digital CMOS data, rather than the LVDS data. Due to time constraints, it was desirable to first try to retrieve data using the method with less difficult processing involved. It was also hypothesized that in needing less complicated processing, a faster frame rate could be achieved if there were to be delays from processing of the data. Nevertheless, it is also possible to change part of the designed program in order to use LVDS data instead, as the hardware is designed to be able to retrieve both sets of

digital image data. However it should be noted that the settings of the camera should be changed to correspond to the transmission of the digital data that the MCU is trying to retrieve.

Theoretically, this firmware design should enable effective digital data retrieval, as well as allow the CPU to be available for processing of the incoming data, however there are many unknowns, specifically with regards to the time between both line and frame valid signals, as they are defined as variable in the documentation [9]. Verification of this design is therefore only possible with further development, with the aid of a functioning version of the designed hardware.

Necessarily, there was also a payload computer driver designed. It was implemented to convert the incoming RAW frame to JPEG format, and transmission of the JPEG frames to the ground station as an IMC message. The necessary complexity of the driver for receiving the incoming image data is decreased since the custom PCB is defined as a standard USB video streaming device. It is further simplified since the driver is implemented as a DUNE task. This enables the driver to use tools already available through DUNE to perform the required function, specifically with regards to establishing and using a serial USB port, compressing to JPEG format, and transmitting an image frame as an IMC message to the ground station. It should be noted that while a prototype of this driver has been implemented, its functionality has not been verified. Additional functionality also does need to be developed in order to enable the onboard computer to record video (in storing several still images).

Even though the MCU starter kit was available, it was impossible to complete development of the MCU program without the designed PCB. Unfortunately, due to the time necessary to manufacture the hardware, it did not arrive in time to verify its design and functionality, or that of the MCU firmware and payload computer driver task.

Chapter 4

Summary and Recommendations for Further Work

4.1 Summary and Conclusions

Firstly, the camera control interface was implemented as part of the initial phase of system familiarisation. As detailed in Chapter 2, there were three parts of the system to create: the camera control GUI, to be added as part of the ground station; the new IMC message type, to send camera control arguments using the existing message format; and the onboard computer driver, used to receive the relevant IMC messages, decode them for camera settings information and relay them in the correct format to the camera, as well as relay the camera reply to the ground station in the same way. Due to there being necessary implementable components in all parts of the UAV system, the advantages of kinaesthetic learning could be utilised while gaining and understanding of the existing system. Simultaneously, a useful feature was implemented without success being highly crucial.

The three necessary parts of the camera control interface were implemented, beginning with definition of the new IMC message type. The payload software system was then rebuilt, and two tasks were added to send and receive respectively, an IMC message of this type in order to verify the process had been done correctly. These tasks were then extended to implement the functionality of the payload computer driver. It was tested in phases, to verify several things, including its ability to communicate through a serial port to the camera, format messages in the correct way for giving

the camera commands, calculate the correct CRC values, receive and decode camera control IMC messages, and relay the information in the camera replies. The existing ground station software was extended to incorporate a camera control GUI, able to relay camera control IMC messages upon the selection of settings graphically. It was tested in its ability to encode camera control IMC messages correctly by visual inspection of messages outputted to a terminal. The first objective was therefore successfully met, as the three parts of the interface were implemented and tested modularly.

With regards to the second objective, there was a large amount of progress made towards retrieving the digital data from the FLIR Tau 2 camera, as was detailed in Chapter 3. After consideration of hardware accessories available from FLIR, it was decided that the best approach to a hardware solution would be an attempted implementation of a custom PCB. This would allow access to all the desired camera features, while maintaining the ease-of-use of the UAV system, as it would be able to be used as a simple USB device. The functionality of the designed PCB is depicted in Figure 3.6. The hardware design consisted of operational, schematic and layout design steps.

The final design contains a central microcontroller, which has a USB peripheral capability, and can therefore act as a USB device. This USB connection also provides power. The MCU has direct access to 8 bit CMOS digital data directly from the camera, and LVDS data from a receiver designed to differentiate and amplify the signals from the camera. The timing constraints relevant to reception of CMOS and LVDS data are outlined in Figures 3.1 and 3.3 respectively. This means the signal lines on the PCB need to be as close to the same length as possible to enable the maximum amount of time for the MCU to retrieve the data. This length matching for the purpose of timing was achieved in the PCB layout design phase, using meandering methods usually used for impedance matching.

There is also a 6 pin header included to allow direct access to camera serial communications, power, ground, external sync, and an undefined MCU GPIO pin, intended for future use. Several MCU peripherals are also contained in the hardware design, as well as other components, some of which are purely for the purpose of firmware and software development aid, and can be eliminated in future prototypes. Specific hardware components were chosen for small package size, voltage rating, stock availability and price. The design was executed in CadSoft EAGLE (Easy Applicable Graphical

Layout Editor), and checked with the internally included error checking tools, however due to the time necessary for manufacturing, was not able to be physically verified.

Due to the availability of an MCU development kit, the same MCU was chosen to be used in the design, in order to accelerate the process of developing the firmware for the custom PCB and the software designed to interface with it on the payload computer. The MCU program is designed to retrieve digital data from the camera, and transmit it to the payload computer via USB in a predefined uncompressed format.

The firmware program utilizes the hardware peripherals available in the chosen MCU, which enable it to receive image data without use of the CPU in a faster manner than software. These include the Peripheral Reflex System to transfer the line valid signal to the TIMER, which overflows immediately on a rising edge to request a Direct Memory Access transfer of the image data from the GPIO port to memory. When a full line has been transferred it is processed in software. In the event that processing of the line is still active upon the next line valid rising edge, there is no problem as reception of the incoming data is handled completely in hardware.

The first prototype of the MCU program was designed to retrieve the 8 bit CMOS digital data, rather than the LVDS data, due to the processing being less complicated, and therefore the frame rate is most likely to be maintained. While there was a development kit available, full testing of the MCU program was unable to be performed due to time constraints and the necessary time needed for PCB manufacturing. In the event that it could be tested, there should be specific focus on correct reception of the data considering clock rate.

A payload computer driver was designed to receive the images from the custom PCB via a USB connection, compress them to JPEG format and relay them to the ground station in an IMC message marked as an image frame. This driver task utilizes tools accessible to all DUNE tasks.

Even though all components of the digital data retrieval system were implemented, their functionality has not been fully confirmed. The time needed for manufacturing of the custom PCB meant there was insufficient time to verify its functionality or that of the firmware and software. Future work should lead to a suitable solution, especially since the design of the hardware included several components meant for aid in development.

4.2 Discussion

The camera control interface was implemented in order to meet the objective of enabling camera settings to be changed during UAV flight. Its largest advantage is that it explicitly increases efficiency and cost-effectiveness, because landing the UAV is not necessary. The implemented control interface is immediately able to be used with the existing UAV system and the FLIR VPC Module to change camera settings. While it has been successfully implemented, it is not complete. There are an extensive number of possible camera commands and settings to change, and not all of them were implemented in the graphical user interface. Nevertheless, all possible camera commands were encoded into a Java class contained within the GUI files, in order to decrease the necessary work in order to fully implement the ones not already included in the GUI. The commands that have had an interface implemented are included in a GUI that fits seamlessly into the existing ground station software, Neptus, and can be easily accessed through the Tools menu in the selected console. It is simple to use where it utilizes radio buttons which indicate mutually exclusive options, sliders which clearly indicate maximum and minimum values, and buttons to execute single-time commands, therefore not requiring the user to have any specific prior knowledge about the limits of the camera.

In contrast, there are some commands which were included in the GUI but may need to have their interfaces improved in terms of ease-of-use. One specific example is the Region of Interest tab, where a selected portion of the current view can be selected as the focus of a snapshot, in order to affect contrast and quality in that area. The ideal interface for this command would be a rectangle indicating the current frame, with the user able to select an area of the rectangle using their mouse in a click-and-drag fashion. The selected area would then be transmitted as the required ROI. However, this is significantly difficult to implement, especially for a beginner Java programmer. The current implementation is four text boxes where the user should enter the coordinates of the corners indicating their desired region of interest. This is clearly not as easy to use as a selectable area on an image, and therefore the GUI has some areas where ease-of-use is lacking.

The GUI has been specifically tailored to the FLIR Tau 2 camera, in accordance to the existing system. In the event that a different camera were to be used, even from the

same manufacturer, the control interface would most likely malfunction, if not become completely useless. The GUI was however implemented in such a way that commands are easy to change, in that they are all included in the same class file. In order to send a different code (but indicating the same command), all that would need to be done is change the command function class file to the one corresponding to the camera in use. If there became such an instance where the maximum and minimum values were to differ, the command argument class file would need to be changed.

Each file is responsible for a group of corresponding commands, and is set up to structure the GUI interface, as well as handle all messages corresponding to the commands (both sent and reply). This modular structure is useful in that in the event that more commands are added to the GUI, a separate file just needs to be added to the existing structure and included in the main camera control command file. While it is not of a significantly high priority, there should also be a small note to indicate the aesthetics of the GUI. It obviously does not affect functionality, but in the event that it were to be used more extensively, the look and feel of the GUI should be improved in order to reflect a professional vibe.

The implemented IMC message for carrying camera control command information works seamlessly with the existing IMC system, enabling the existing encoding and decoding software to be used in both the ground station software and the payload computer software. The defined structure of the new message reflects the structure of the camera serial interface protocol, containing all values relevant to the command, minus the CRC values, which are calculated on the payload computer. This simplifies the process used by the payload computer driver to decode the message sent from the GUI, and encode it in the format necessary for camera communication. The structure of the camera control IMC message also makes it easier for the GUI to differentiate between camera replies, since there are different reply-handling functions for every set of commands.

The payload computer software, DUNE is essentially a collection of tasks performing different functions, all overseen by the same scheduler. The payload driver is contained within a DUNE task, enabling convenient use of the common tools available to all tasks. Implementing the driver in this way decreases its necessary complexity, as all functions necessary for receiving an IMC message, opening a serial connection to

the camera, and sending a receiving messages from the connection already exist in DUNE. This means that any changes that may need to be made can be executed easily by someone who is already familiar with the DUNE software. The modular nature of the task code enables simplicity and flexibility even where a different communication protocol needs to be used: the only functions that need to change are those responsible for encoding and decoding the camera protocol. The function for calculation of the CRC-CCITT values was also implemented modularly, and placed in the directory containing utilities accessible to all DUNE tasks. This will allow for possible other feature application tasks to use it.

While the system has been deemed operational and all modules have been tested, the system has not been thoroughly validated in an operational environment. Therefore there may be disfunction present in the system, and further testing is definitely necessary before it is relied upon in a UAV mission.

The approach for retrieving the digital data from the FLIR Tau 2 camera involved design of a custom PCB. The design included many components purely for the purpose of aiding in development of the MCU program, to enable different methods of completing necessary tasks, or to enable additional functions to be added in future work, without needing to re-design the hardware layout and repeat manufacture. These features will also enable efficient evaluation of the PCB hardware. Flashing of the MCU and debugging of its program can be done using the debug header with a J-Link accessory. The user LEDs can also be used for debugging or to indicate when actions being performed by the MCU. The 6-pin header enables access to several functions. The external sync function of the camera is accessible, which will enable synchronized interfacing between the thermal camera and other payload components. The serial interface of the camera is also made available on this header, which will enable handling of camera control messages to and from the camera. Power can also be provided via this header, in the event that power from the USB is not sufficient. There is also an undefined MCU pin connected to the header, allowing for future use, possibly as a PWM output to be used to control the gimbal. Many of these features may be deemed redundant in future prototypes, but for now they are included to ease the development process and allow for extensive system flexibility.

Ease-of-use was kept in mind during the entire process of custom PCB design, not

just for the end user, but also for those who will develop with it in the future. The PCB is of a similar size to the back of the camera, and fits comfortably into the gimbal. The PCB design specifically uses a vertical USB connector, in order to enable easy access to the plug when inside the gimbal. There are however no mounting screws, which may lead to noise interfering with the digital data retrieval, and this should be noted in further testing. The chosen MCU is easy to interface with, given the included debug header in the design, and extensive documentation for it being easily accessible either online or through the manufacturer's IDE, Simplicity Studio. The existing libraries increase efficiency of firmware and software development.

There is a significant difficulty in retrieving the digital data from the camera in that it is transmitted at a very high rate. It was attempted during the hardware design phase to maximise the time during which the data is valid, by matching the lengths of the relevant signals using layout methods usually used for impedance matching. This problem has been further addressed by using a hardware-assisted approach in designing the MCU program, however could still be an issue if the current program design is not sufficient in retrieval rate.

Unarguably, the largest limitation with the digital data retrieval system is its lack of testing. While there were many decisions made to increase the likelihood of attaining the required functionality in all implemented components, it has not been fully confirmed. It therefore follows that there is a limited amount of information available with regards to discussion of the system's strengths and weaknesses. There was significant effort put into the early stages of development, in order to account for as many possible scenarios as possible and maximise flexibility in terms of future work. However, there may still be problems that arise that have not been accounted for. This may result in the need for redesign of the hardware and software. Nevertheless, the progress made as described in this document should prove to be a solid foundation for future work, and a suitable solution should eventuate, enabling the analog-to-digital converter to be removed from the payload.

4.3 Recommendations for Further Work

4.3.1 Short Term

There was significant progress made with regards to retrieving the digital data from the FLIR Tau 2 camera, however it was not possible to test the system due to time constraints. Therefore the primary focus in future work should be attaining system functionality. Firstly, component placement on the manufactured PCB should be completed with accordance to the design layout, after which it should be evaluated. Assuming it is functional, it should be verified that the hardware-assisted design of the MCU program is fast enough to capture the incoming data. If this is not the case, options could include further optimization of the existing MCU program, using a faster oscillator to clock the MCU allowing for more clock cycles to capture the port value, or interchanging of the MCU itself in favour of one designed specifically for image data processing. Failing all of these, a solution could be attempted using the FLIR Camera Link Accessory.

In the event that a functioning system is verified using the custom PCB, the resulting frame rate should be measured. If the frame rate is maintained, but the image is distorted due to noise, the program should be changed to use LVDS data instead of CMOS. Noise from vibrations should also be noted, in order to gauge whether mounting screws should be included in future designs of the hardware.

4.3.2 Mid Term

Once the functionality of the digital data retrieval system has been confirmed, complete system verification and validation should be performed. This will involve ground testing where the payload is communicated with wirelessly but not in flight (verification of functionality), and flight testing in both normal conditions (validation of functionality) and harsh conditions (validation of robustness). There should also be tests to gauge ease-of-use by a non-expert operator.

When the digital data retrieval system has been validated, incorporation of the camera control interface should be implemented. The PCB hardware already includes access to the camera serial communication interface, so therefore there only needs to be changes made in software. Two approaches to this could be considered: connecting a

serial port on the payload computer GPIO to the camera serial communication access on the custom PCB's 6-pin header, or handling the camera control commands via the USB connection responsible for image data transmission. Both approaches are possible using the existing hardware. For the former, the DUNE task for handling camera control commands could be used, with an alternate target for the serial port connection (GPIO instead of USB). For the latter, the DUNE task for handling incoming digital data from the custom PCB should be altered to also handle camera control commands and replies. This approach would also require alteration of the MCU program to forward commands from the payload computer to the camera and vice versa.

Eventually, the remaining camera control commands in the ground station GUI should be implemented. As they are already encoded, and due to the modular nature of the existing GUI, this will involve adding new JPanel class files to the existing program heirarchy. Improvement should be made to already implemented functions such as ROI in order to make them easier to use and understand. The GUI should also be improved with regards to aesthetics.

4.3.3 Long Term

If a functioning system is attained using the custom PCB, use of the undefined MCU pin accessible on the 6 pin header could be specified. This could possibly include a PWM output on this pin to control the gimbal, enabling use of this payload system on a non-autonomous UAV (that is, a remote controlled UAV with no autopilot on board). PWM output using the selected MCU is simply executed using a hardware TIMER, as this is a listed function in its documentation.

Following this, the external sync function accessible from the same header could be used for interfacing with other components. These components could include a camera operating in a different spectrum (i.e. visible spectrum as opposed to thermal, in order to provide two different versions of the same frame), and/or a positional module to provide GPS coordinates for the transmitted frame. In the event that the custom PCB solution is suitable for retrieving the digital data, 300g of the 1kg payload should be available to include more payload components.

Once all possible functionality has been implemented on the custom PCB, the redundant components can be excluded from the hardware layout. Once the MCU pro-

gram has been finalised, the MCU can be flashed using the USB, and the debug header and user LEDs can be removed, along with several other components (depending on the established design, these will vary). This will mean the PCB can be downsized, and possibly a 3D printed case made for it in order to add a layer of abstraction and exude a professional vibe.

Appendix A

Acronyms

A/D Analog-to-Digital

ACE Active Contrast Enhancement

AGC Automatic Gain Control

CMOS Complementary Metal Oxide Semiconductor (type of signal logic)

CRC Cyclic Redundancy Check

DDE Digital Detail Enhancement

EAGLE Easy Applicable Graphical Layout Editor

EMECS Erasmus Mundus Embedded Computing Systems

ESD Electrostatic Discharge

GUI Graphical User Interface

IDD Interface Description Document

IMC Intermodule Communication

LVDS Low Voltage Differential Signalling

LWIR Long Wavelength Infrared

MCU Microcontroller Unit

NTNU Norges teknisk-naturvitenskapelige universitet (Norwegian University of Science and Technology)

OS Operating System

PCB Printed Circuit Board

PWM Pulse Width Modulation

ROI Region of Interest

SSO Smart Scene Optimization

UAV Unmanned Aerial Vehicle

USB-IF USB Implementers Forum, Inc.

USB Universal Serial Bus

Appendix B

List of Camera Control Commands

The following chapter shows the encoded camera commands in the implemented camera control interface. The format of is as follows:

FUNCTION_NAME(1, 2, 3, "FOUR", "FIVE")

where:

- 1 = Function Code
- 2 = Number of bytes of argument in command message
- 3 = Number of bytes of argument in reply message
- FOUR = Human readable description of the command
- FIVE = Home panel for handling of GUI representation, and message handling for the command

NOTE: The instructions WITH a home panel have been implemented in the camera control interface. Those with a "?" in this position have not yet been implemented, only encoded.

Further information on what each instruction does can be found in the camera's Software IDD [[13](#)].

NO_OP(0x00, 0, 0, "No operation", "ThermalCamControlGui"),
 SET_DEFAULTS(0x01, 0, 0, "Set defaults non-blocking", "SettingsButtonsPanel"),
 CAMERA_RESET(0x02, 0, 0, "Reset / reboot", "SettingsButtonsPanel"),
 RESTORE_FACTORY_DEFAULTS(0x03, 0, 0, "Revert to factory defaults", "SettingsButtonsPanel"),
 SERIAL_NUMBER(0x04, 0, 8, "Serial number request. "
 + "Reply bytes 0-3 camera, 4-7 sensor serial no.", "StatusPanel"),
 GET_REVISION(0x05, 0, 8, "Get software / firmware revision. "
 + "Reply 2 bytes ea: sw maj/min, fw maj/min", "StatusPanel"),
 BAUD_RATE_GET(0x07, 0, 2, "Get current baud rate", "?"),
 BAUD_RATE_SET(0x07, 2, 2, "Set baud rate", "?"),
 GAIN_MODE_GET(0x0A, 0, 2, "Get gain mode", "GainModePanel"),
 GAIN_MODE_SET(0x0A, 2, 2, "Set gain mode", "GainModePanel"),
 FFC_MODE_SELECT_GET(0x0B, 0, 2, "Get FFC Mode", "FFCPanel"),
 FFC_MODE_SELECT_SET(0x0B, 2, 2, "Set FFC Mode", "FFCPanel"),
 FFC_INTEG_FRAMES_GET(0x0B, 4, 2, "Get number of integrated frames during FFC", "FFCPanel"),
 FFC_INTEG_FRAMES_SET(0x0B, 4, 0, "Set number of integrated frames during FFC", "FFCPanel"),
 DO_FFC_AUTO(0x0C, 0, 0, "Auto FFC", "FFCPanel"),
 DO_FFC_SPECIFIED(0x0C, 2, 2, "FFC Long or short", "FFCPanel"),
 FFC_PERIOD_GET(0x0D, 0, 4, "Get interval between FFC frames for both gain states", "FFCPanel"),
 FFC_PERIOD_SET(0x0D, 2, 2, "Set interval between FFC frames for current gain state",
 "FFCPanel"),
 FFC_PERIOD_SPECIFY(0x0D, 4, 4, "Set interval between FFC frames for specific gain states. "
 + "Bytes 0-1 for high gain, 2-3 for low gain", "FFCPanel"),
 FFC_TEMP_DELTA_GET(0x0E, 0, 4, "Get temp diff used to trigger auto FFC. Reply bytes "
 + "0-1 for high gain, 2-3 for low gain", "FFCPanel"),
 FFC_TEMP_DELTA_SET(0x0E, 2, 2, "Set temp delta value for current gain state", "FFCPanel"),
 FFC_TEMP_DELTA_SPECIFY(0x0E, 4, 4, "Set temp delta value for specific gain states. "
 + "Bytes 0-1 for high gain, 2-3 for low gain", "FFCPanel"),
 VIDEO_MODE_GET(0x0F, 0, 2, "Get current video signal mode", "AnalogVideoOnOffPanel"),
 VIDEO_MODE_SET(0x0F, 2, 2, "Set analog video mode", "AnalogVideoOnOffPanel"),
 VIDEO_SYBOLOGY_OVERLAY_GET(0x0F, 4, 2, "Get symbology overlay state, different "
 + "arguments for analog/digital", "?"),
 VIDEO_SYBOLOGY_OVERLAY_SET(0x0F, 4, 4, "Set symbology overlay state, different "
 + "arguments for analog/digital", "?"),
 VIDEO_PALETTE_GET(0x10, 0, 2, "Get video palette", "PolarityPanel"),
 VIDEO_PALETTE_SET(0x10, 2, 2, "Set video palette", "PolarityPanel"),
 VIDEO_ORIENTATION_GET(0x11, 0, 2, "Get video orientation", "OrientationPanel"),
 VIDEO_ORIENTATION_SET(0x11, 2, 2, "Set video orientation", "OrientationPanel"),
 DIGITAL_OUTPUT_MODE_GET(0x12, 0, 2, "Get digital output mode", "DigitalPanel"),
 DIGITAL_OUTPUT_MODE_GET_SUB(0x12, 2, 2, "Get digital output mode (various sub functions)",
 "DigitalPanel"),
 DIGITAL_OUTPUT_MODE_SET(0x12, 2, 2, "Enable/disable digital output (both LVDS and XP
 channels)", "DigitalPanel"),
 DIGITAL_OUTPUT_MODE_SET_SUB(0x12, 2, 2, "Set digital output mode (various sub functions)",
 "DigitalPanel"),

AGC_TYPE_GET(0x13, 0, 2, "Get AGC algorithm", "AgcModesPanel"),
 AGC_TYPE_SET(0x13, 2, 2, "Set AGC algorithm", "AgcModesPanel"),
 AGC_SUB_GET(0x13, 2, 2, "AGC sub functions", "AgcModesPanel"),
 AGC_SUB_SET(0x13, 4, 0, "AGC sub functions", "AgcModesPanel"),
 CONTRAST_GET(0x14, 0, 2, "Get contrast value used by once-bright, auto-bright and manual"
 + "AGC algorithms", "ManualParamPanel"),
 CONTRAST_SET(0x14, 2, 2, "Set contrast value used by once-bright, auto-bright and manual"
 + "AGC algorithms", "ManualParamPanel"),
 BRIGHTNESS_GET(0x15, 0, 2, "Get brightness value used by manual and auto-bright AGC",
 "ManualParamPanel"),
 BRIGHTNESS_SET(0x15, 2, 2, "Set brightness value used by manual and auto-bright AGC",
 "ManualParamPanel"),
 BRIGHTNESS_BIAS_GET(0x18, 0, 2, "Get brightness bias value used by once-bright AGC", "?"),
 BRIGHTNESS_BIAS_SET(0x18, 2, 2, "Set brightness bias value used by once-bright AGC", "?"),
 TAIL_SIZE_GET(0x1B, 0, 2, "Get the tail rejection percentage for AGC", "?"),
 TAIL_SIZE_SET(0x1B, 2, 2, "Set the tail rejection percentage for AGC", "?"),
 ACE_CORRECT_GET(0x1C, 0, 2, "Get the Active Contrast Enhancement Correction for AGC",
 "EnhancePanel"),
 ACE_CORRECT_SET(0x1C, 2, 2, "Set the Active Contrast Enhancement Correction for AGC",
 "EnhancePanel"),
 LENS_NUMBER(0x1E, 0, 0, "", "?"),
 SPOT_METER_MODE_GET(0x1F, 0, 2, "Get the spot-meter mode", "?"),
 SPOT_METER_MODE_SET(0x1F, 2, 2, "Set the spot-meter mode", "?"),
 READ_SENSOR(0x20, 2, 2, "Get various data from the core, depending on "
 + "argument of incoming message", "?"),
 READ_SENSOR_ACCEL(0x20, 2, 8, "Get accelerometer data", "?"),
 EXTERNAL_SYNC_GET(0x21, 0, 2, "Get external sync mode", "ExternalSyncPanel"),
 EXTERNAL_SYNC_SET(0x21, 2, 2, "Set external sync mode", "ExternalSyncPanel"),
 ISOTHERM_GET(0x22, 0, 2, "Get isotherm mode on/off", "?"),
 ISOTHERM_SET(0x22, 2, 2, "Set isotherm mode on/off", "?"),
 ISOTHERM_THRESHOLD_GET(0x23, 0, 6, "Get isotherm thresholds in percent. Bit 15 units", "?"),
 ISOTHERM_THRESHOLD_SET(0x23, 6, 6, "Set isotherm thresholds in percent. Bit 15 units", "?"),
 TEST_PATTERN_GET(0x25, 0, 2, "Get the current test pattern", "TestPatternPanel"),
 TEST_PATTERN_SET(0x25, 2, 2, "Set the test pattern", "TestPatternPanel"),
 VIDEO_COLOR_MODE_GET(0x26, 0, 2, "Get the color mode (color enabled or monochrome)",
 "AnalogVideoColorPanel"),
 VIDEO_COLOR_MODE_SET(0x26, 2, 2, "Set the color mode (color enabled or monochrome)",
 "AnalogVideoColorPanel"),
 SPOT_METER_GET(0x2A, 0, 2, "Get the value of the spot meter in deg C", "?"),
 SPOT_DISPLAY_GET(0x2B, 0, 2, "Get the spot meter display mode", "?"),
 SPOT_DISPLAY_SET(0x2B, 2, 2, "Set the spot meter display mode", "?"),
 DDE_GAIN_GET(0x2C, 0, 2, "Get gain value for DDE in manual mode", "EnhancePanel"),
 DDE_GAIN_SET(0x2C, 2, 2, "Set gain value for DDE in manual mode", "EnhancePanel"),
 SYMBOL_CONTROL(0x2F, 2, 2, "Sets symbol command (set cmd 14-46 arg bytes)", "?"),
 SPLASH_CONTROL_GET(0x31, 0, 4, "Get the splash screen delay parameters", "?"),

SPLASH_CONTROL_SET(0x31, 4, 4, "Set the splash screen delay parameters", "?"),
 EZOOM_CONTROL_GET_CURRENT(0x32, 0, 2, "Get current zoom width", "PanZoomPanel"),
 EZOOM_CONTROL_GET_SPECIFIC(0x32, 4, 2, "Get specific zoom width", "PanZoomPanel"),
 EZOOM_CONTROL_SET(0x32, 4, 0, "Set zoom width", "PanZoomPanel"),
 FFC_WARN_TIME_GET(0x3C, 0, 2, "Get FFC warn time", "AnalogFFCPanel"),
 FFC_WARN_TIME_SET(0x3C, 2, 2, "Set FFC warn time in frames", "AnalogFFCPanel"),
 AGC_FILTER_GET(0x3E, 0, 2, "Get the AGC Filter value", "?"),
 AGC_FILTER_SET(0x3E, 2, 2, "Set the AGC Filter value", "?"),
 PLATEAU_LEVEL_GET(0x3F, 0, 2, "Get the plateau level for the plateau AGC algorithm", "?"),
 PLATEAU_LEVEL_SET(0x3F, 2, 2, "Set the plateau level for the plateau AGC algorithm", "?"),
 SPOT_METER_DATA_GET(0x43, 0, 2, "Get value of the spot meter in deg C. If not enabled, "
 + "returns average value of center four pixels", "?"),
 SPOT_METER_DATA_GET_DETAIL(0x43, 2, 20, "Gets the average, min and max pixel values"
 + "for the spot meter", "?"),
 SPOT_METER_COORDINATES_GET(0x43, 2, 12, "Get spot meter coordinates", "?"),
 SPOT_METER_COORDINATES_SET(0x43, 8, 4, "Set spot-meter coordinates", "?"),
 AGC_ROI_GET(0x4C, 0, 8, "Gets the region of interest", "RoiPanel"),
 AGC_ROI_SET(0x4C, 8, 8, "Sets the region of interest", "RoiPanel"),
 SHUTTER_TEMP_GET(0x4D, 0, 2, "Get the temperature of the shutter. reply degC x 100", "?"),
 SHUTTER_TEMP_SET(0x4D, 2, 0, "Set the temperature of the shutter in degC x 100", "?"),
 SHUTTER_TEMP_RADIOMETRY_CALC_GET(0x4D, 4, 2, "Get the shutter temp calculation mode "
 + "for radiometry", "?"),
 SHUTTER_TEMP_RADIOMETRY_CALC_SET(0x4D, 4, 0, "Set the shutter temp calculation mode "
 + "for radiometry", "?"),
 AGC_MIDPOINT_GET(0x55, 0, 2, "Get the AGC midpoint offset for plateau-equalization "
 + "and linear AGC algs", "?"),
 AGC_MIDPOINT_SET(0x55, 2, 2, "Set the AGC midpoint offset for plateau-equalization "
 + "and linear AGC algs", "?"),
 SERIAL_NUMBER_GET_REDUNDANT(0x65, 0, 8, "Get serial number of cam and sensor",
 "StatusPanel"),
 CAMERA_PART_GET(0x66, 0, 32, "Get camera part. Reply in ASCII", "StatusPanel"),
 READ_ARRAY_AVERAGE(0x68, 0, 4, "Read the mean of the current frame. Not ROI dependent",
 "?"),
 MAX_AGC_GAIN_GET(0x6A, 0, 2, "Get the max-gain parameter for plateau AGC", "?"),
 MAX_AGC_GAIN_SET(0x6A, 2, 2, "Set the max-gain parameter for plateau AGC", "?"),
 PAN_AND_TILT_GET(0x70, 0, 4, "Get the pan (x axis) and tilt (y axis) for the zoom window "
 + "relative to the center of the unzoomed window. Bytes 0-1: Tilt pos, 2-3: Pan pos in rows",
 "PanZoomPanel"),
 PAN_AND_TILT_SET(0x70, 4, 4, "Set the pan (x axis) and tilt (y axis) for the zoom window "
 + "relative to the center of the unzoomed window. Bytes 0-1: Tilt pos, 2-3: Pan pos in rows",
 "PanZoomPanel"),
 VIDEO_STANDARD_GET(0x72, 0, 2, "Get the video standard", "AnalogVideoStandardPanel"),
 VIDEO_STANDARD_SET(0x72, 2, 2, "Set the video standard", "AnalogVideoStandardPanel"),
 SHUTTER_POSITION_GET(0x79, 0, 2, "Get the shutter position. Non-blocking", "?"),
 SHUTTER_POSITION_SET(0x79, 2, 2, "Set the shutter position. Non-blocking", "?"),

SHUTTER_PROFILE_GET(0x80, 0, 34, "Get the shutter profile (safety timeout + close/open table",
 "?"),
 SHUTTER_PROFILE_SET(0x80, 34, 34, "Set the shutter profile (safety timeout + close/open table",
 "?"),
 TRANSFER_FRAME(0x82, 4, 4, "Capture a snapshot to a specified buffer location. Non-blocking",
 "?"),
 TLIN_COMMANDS_RESOLUTION_GET(0x8E, 2, 2, "Get the resolution of the TLinear digital video",
 "?"),
 TLIN_COMMANDS_RESOLUTION_SET(0x8E, 4, 0, "Set the resolution of the TLinear digital video",
 "?"),
 TLIN_COMMANDS_OUTPUT_GET(0x8E, 2, 2, "Get TLinear output enabled status", "?"),
 TLIN_COMMANDS_OUTPUT_SET(0x8E, 4, 0, "Enable/disable TLinear output", "?"),
 CORRECTION_MASK_GET(0xB1, 0, 2, "Get the corrections applied", "?"),
 CORRECTION_MASK_SET(0xB1, 2, 2, "Set the corrections applied", "?"),
 MEMORY_STATUS(0xC4, 0, 2, "Get the status for several non-blocking write/erase commands", "?"),
 WRITE_NVFFC_TABLE(0xC6, 0, 0, "Write the FFC map to nonvolatile memory", "?"),
 READ_MEMORY(0xD2, 6, 256, "Reads specified number of bytes beginning at specified address. "
 + "Bytes 0-3: address, 4-6: number of bytes to read", "?"),
 ERASE_MEMORY_BLOCK(0xD4, 2, 2, "Erases a block or a range of non-volatile memory", "?"),
 GET_NV_MEMORY_SIZE(0xD5, 2, 8, "Get the base address and block size of the non-volatile
 memory device. "
 + "Reply bytes 0-3: base address, 4-7: block size in bytes", "?"),
 GET_MEMORY_ADDRESS(0xD6, 4, 8, "Get the memory address and size of the specified buffer",
 "?"),
 GAIN_SWITCH_PARAMS_GET(0xDB, 0, 8, "Get the population (%) and temp (degC) thresholds for
 automatic high/low gain-state switching", "?"),
 GAIN_SWITCH_PARAMS_SET(0xDB, 8, 8, "Set the population (%) and temp (degC) thresholds for
 automatic high/low gain-state switching", "?"),
 DDE_THRESHOLD_GET(0xE2, 0, 2, "Get the threshold of the DDE filter", "?"),
 DDE_THRESHOLD_SET(0xE2, 2, 2, "Set the threshold of the DDE filter", "?"),
 SPATIAL_THRESHOLD_GET(0xE3, 0, 2, "Get the spatial threshold of the DDE filter and the DDE
 mode", "?"),
 SPATIAL_THRESHOLD_SET(0xE3, 2, 2, "Set the spatial threshold of the DDE filter and the DDE
 mode", "?"),
 LENS_RESPONSE_PARAMS_GET(0xE5, 2, 4, "Get the lens parameters for the calculated
 responsivity", "?"),
 LENS_RESPONSE_PARAMS_SET(0xE5, 6, 0, "Get the lens parameters for the calculated
 responsivity", "?"),
 LENS_RESPONSE_PARAMS_RADIOMETRY_GET(0xE5, 2, 2, "Get the scene parameters for
 radiometric calculations", "?"),
 LENS_RESPONSE_PARAMS_RADIOMETRY_SET(0xE5, 4, 0, "Set the scene parameters for
 radiometric calculations", "?"),

Appendix C

Custom PCB Design

In the following section, the specifics of the custom PCB design are depicted. This includes the schematic in Figure [C.1](#) on page [62](#), the bill of materials in Figure [C.2](#) on page [63](#), and the PCB layout in Figures [C.3](#), [C.4](#), and [C.5](#) beginning on page [64](#).

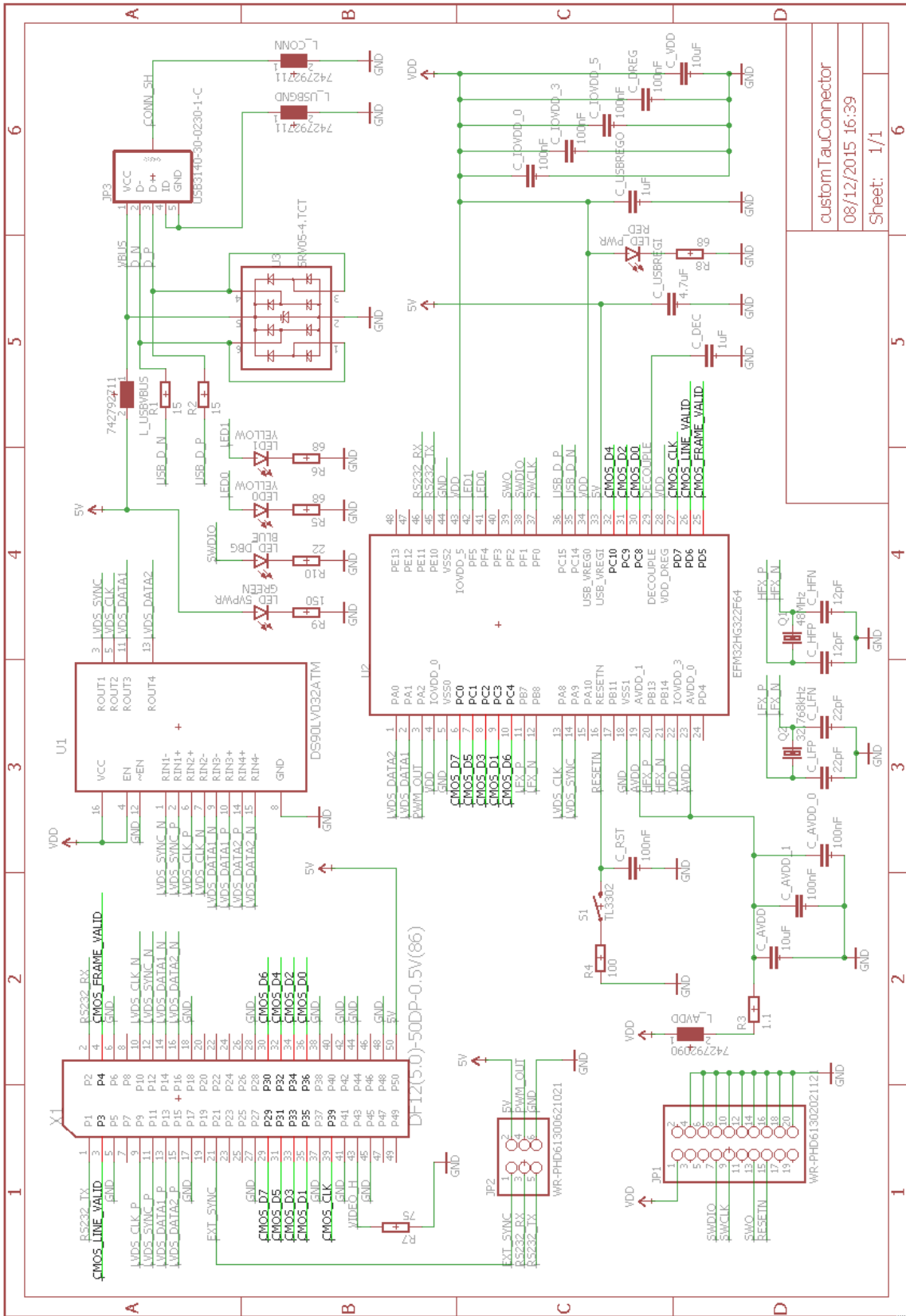


Figure C.1: Custom PCB Schematic

customTauConnector
08/12/2015 16:39
Sheet: 1/1

Part	Value	Package	MF	MPN	OC_FARNELL
C_AVDD	10uF	0402-CAP	TDK	C1005X5R0J106M050BC	2346872
C_AVDD_0	100nF	0402-CAP	AVX	UT026D104MAC2F	2099805
C_AVDD_1	100nF	0402-CAP	AVX	UT026D104MAC2F	2099805
C_DEC	1uF	0402-CAP	AVX	04026D105MAT2A	2332538
C_DREG	100nF	0402-CAP	AVX	UT026D104MAC2F	2099805
C_HFN	12pF	0402-CAP	VISHAY	VJ0402D120JXCAJ	2133989
C_HFP	12pF	0402-CAP	VISHAY	VJ0402D120JXCAJ	2133989
C_IOVDD_0	100nF	0402-CAP	AVX	UT026D104MAC2F	2099805
C_IOVDD_3	100nF	0402-CAP	AVX	UT026D104MAC2F	2099805
C_IOVDD_5	100nF	0402-CAP	AVX	UT026D104MAC2F	2099805
C_LFN	22pF	0402-CAP	AVX	04025A220FAT2A	2332500
C_LFP	22pF	0402-CAP	AVX	04025A220FAT2A	2332500
C_RST	100nF	0402-CAP	AVX	UT026D104MAC2F	2099805
C_USBREGI	4.7uF	0402-CAP	TDK	C1005X5R0J475M050BC	2309027
C_USBREGO	1uF	0402-CAP	AVX	04026D105MAT2A	2332538
C_VDD	10uF	0402-CAP	TDK	C1005X5R0J106M050BC	2346872
JP1	WR-PHD61302021121	2X10	WURTH ELEKTRONIK	61302021121	2356138
JP2	WR-PHD61300621021	2X03	WURTH ELEKTRONIK	61300621121	2356131
JP3	USB3140-30-0230-1-C	USB-B-MICRO-SMD-VERTICAL	GLOBAL CONNECTOR TECHNOLOGY	USB3140-30-0230-1-C	2443139
LED0	YELLOW	LED-0603	KINGBRIGHT	KP-1608SYCK	2290330
LED1	YELLOW	LED-0603	KINGBRIGHT	KP-1608SYCK	2290330
LED_5VPWR	GREEN	LED-0603	KINGBRIGHT	KP-1608MGC	8529825
LED_DBG	BLUE	LED-0603	ROHM	SMLE12BC7TT86Q	1685096
LED_PWR	RED	LED-0603	KINGBRIGHT	KPT-1608SRC-J4	2314401
L_AVDD	742792090	0805	WURTH ELEKTRONIK	74279209	1635722
L_CONN	742792711	0402	WURTH ELEKTRONIK	74279271	1635676
L_USBGND	742792711	0402	WURTH ELEKTRONIK	74279271	1635676
L_USBVBUS	742792711	0402	WURTH ELEKTRONIK	74279271	1635676
Q1	48MHz	CRYSTAL-OSC-SMD-6X3.5	ABRACON	ABM7-48.000MHZ-D2Y-F-T	2101335
Q2	32.768kHz	CRYSTAL-OSC-SMD-4.9X1.8	ABRACON	ABS10-32.768KHZ-1-T	2467869
R1	15	R0402	TE CONNECTIVITY / NEOHM	CPF0402B15RE1	1697294
R2	15	R0402	TE CONNECTIVITY / NEOHM	CPF0402B15RE1	1697294
R3	1.1	R0402	PANASONIC ELECTRONIC COMPONENTS	ERJ2GE1R1X	2324278
R4	100	R0402	WELWYN	PCF0402PR-100RBT1	1768757
R5	68	R0402	WELWYN	ASC0402-68RFT10	2078853
R6	68	R0402	WELWYN	ASC0402-68RFT10	2078853
R7	75	R0402	TE CONNECTIVITY / NEOHM	CPF0402B75RE1	1697305
R8	68	R0402	WELWYN	ASC0402-68RFT10	2078853
R9	150	R0402	PANASONIC ELECTRONIC COMPONENTS	ERA2AED151X	2324722
R10	22	R0402	YAGEO (PHYCOMP)	RC2512JK-7W22RL	1376981
S1	TL3302	TACTILE_SWITCH_TALL	MULTICOMP	MC32882	2396053
U1	DS90LV032ATM	SOIC127P600X175-16N	TEXAS INSTRUMENTS	DS90LV032ATM/NOPB	1469003
U2	EFM32HG322F64	TQFP48	SILICON LABS	EFM32HG322F64G-A-QFP48	2475991
U3	SRV05-4.TCT	SOT-23	SEMTECH	SRV05-4.TCT	1456415
X1	DF12(5.0)-50DP-0.5V(86)	DF12(5.0)-50DP-0.5V(86)	HIROSE(HRS)	DF12(5.0)-50DP-0.5V(86)	1324647

Figure C.2: Custom PCB Bill of Materials

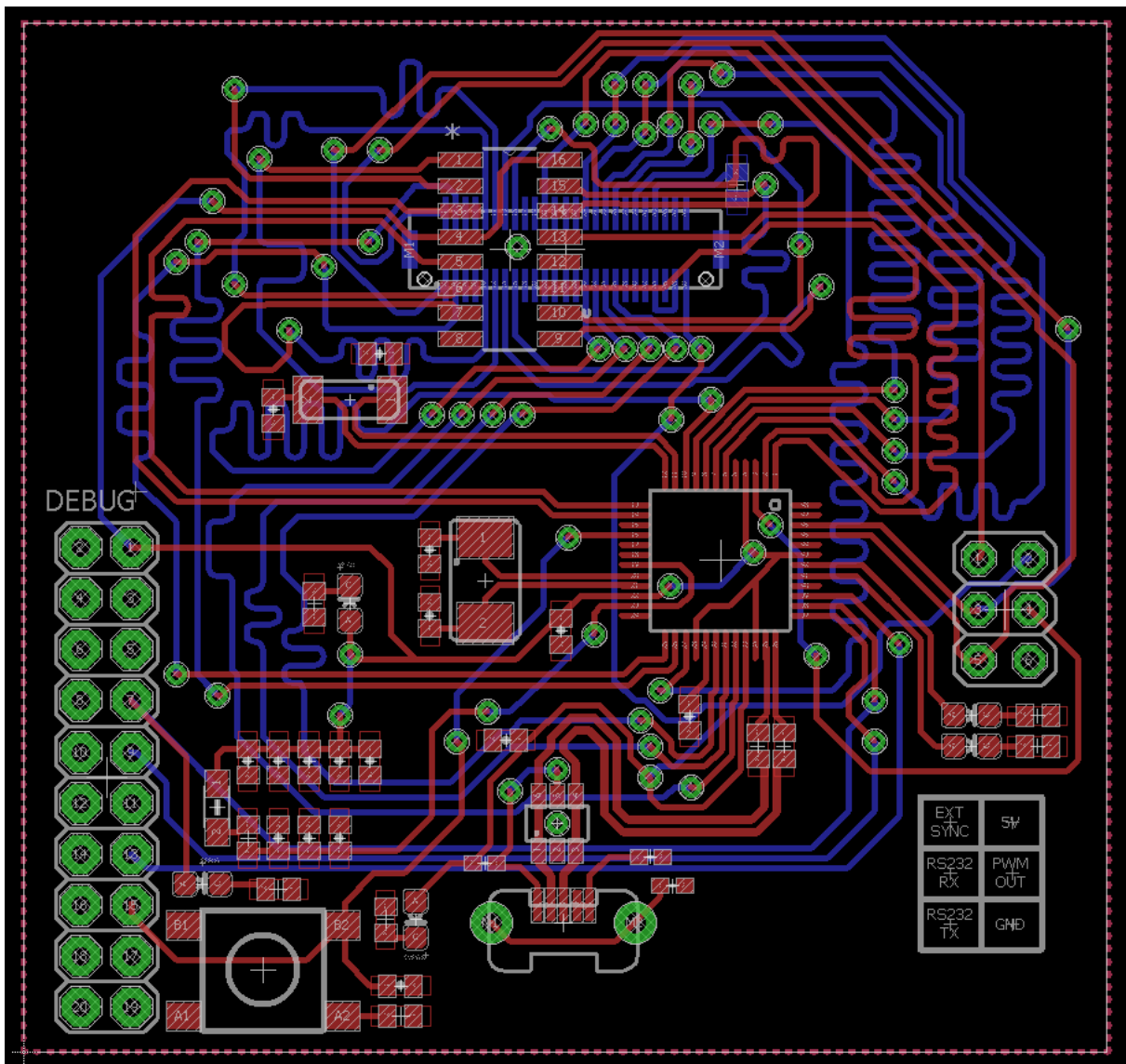


Figure C.3: Custom PCB Layout without ground plane polygons filled.

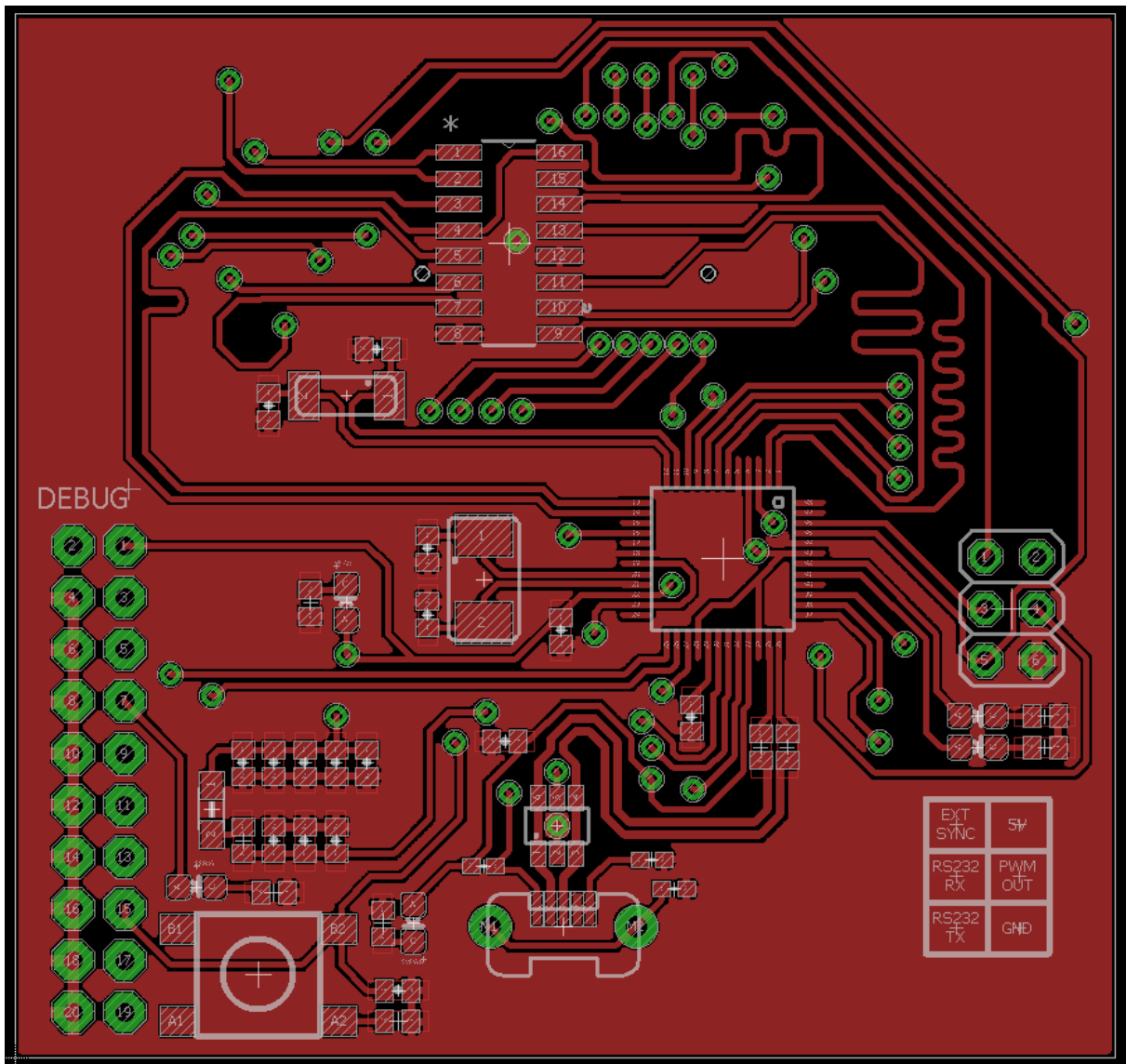


Figure C.4: Custom PCB front layout with ground plane.

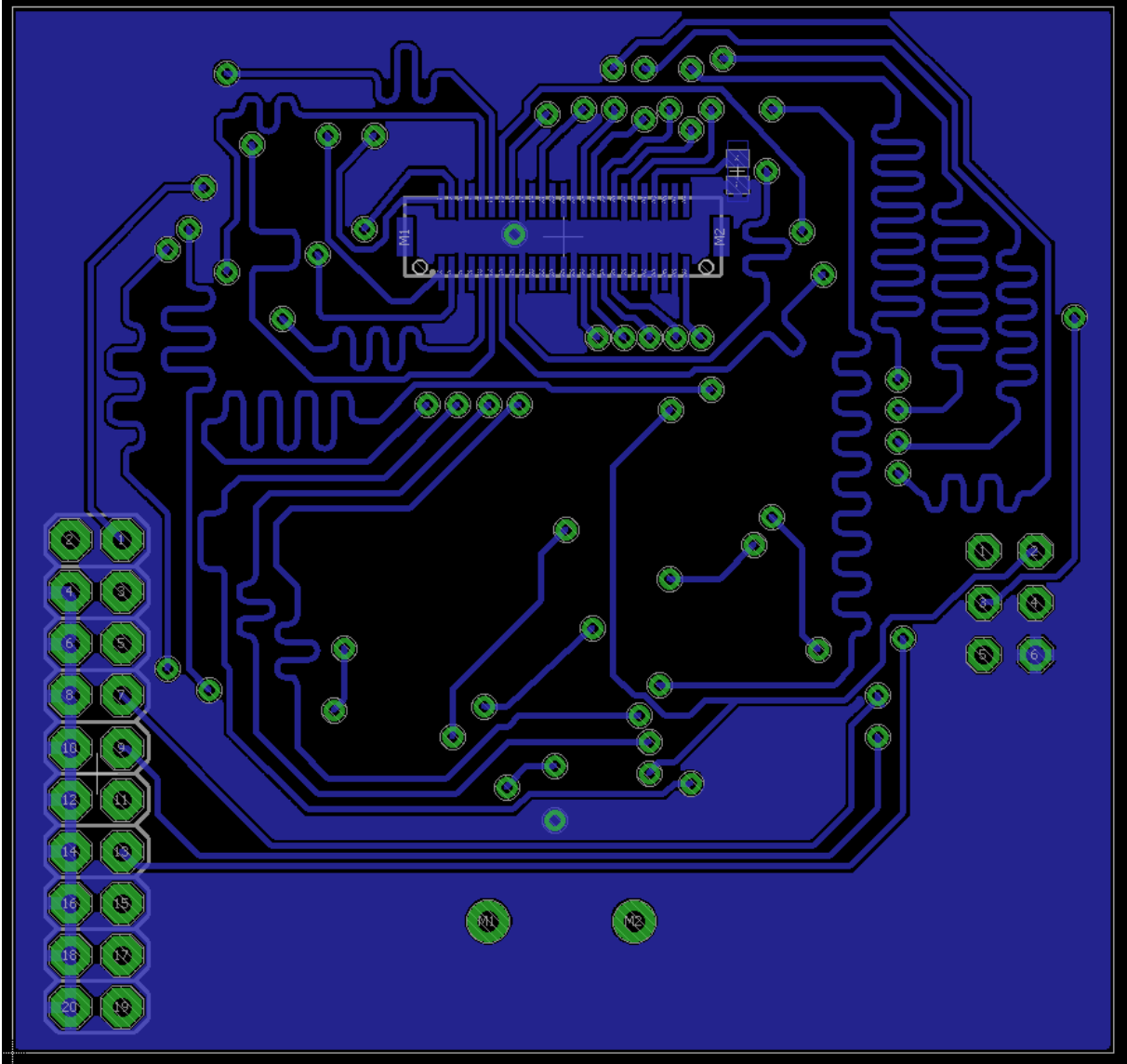


Figure C.5: Custom PCB back layout ground plane.

Appendix D

Demos and Accompanying Files

Full toolchain software containing modifications for the Camera Control Interface can be found on the uavlab git server:

git@uavlab.itk.ntnu.no:uavlab

in the repositories called dune, imc, imcjava and neptus.

(branch name in all: **feature/flir-serial**)

The DUNE version for Digital Data Retrieval can be found in the dune repository at the same location.

(branch name: **feature/flir-digital**)

Both the entire software systems (for demonstration) and single created code files (for code inspection) have been included in the accompanying .zip file. The following files were included to indicate all code produced during development of the Camera Control Interface:

- *1IMCThermalCam*: contains an altered definition of IMC to add a message to carry camera control commands (message ID 1991, second from last).
- *2DUNEAlgorithmForCamCRC*: contains C++ files created for the Camera Control Interface version of DUNE to calculate the cyclic redundancy check (CCITT) values for serial communication.
- *3DUNEfeatureFLIRSerialTask*: contains the C++ DUNE Task created that is responsible for receiving camera control IMC messages, encoding them into cam-

era serial protocol, calculating relevant CRC values, transmitting them to the camera and handling the camera reply.

- *4NeptusThermalCamPackage*: contains the Java Neptus package responsible for the added section of the ground station to allow sending of camera control commands.
- *5FullSystemFeatureFlirSerial*: contains full toolchain relevant to the Camera Control Interface. Run neptus.exe in the neptus folder and open a console (e.g. uav-light) -> Tools -> FLIR Tau 2 Camera Control GUI to see GUI.

The following files were included to indicate work done towards the Digital Data Retrieval:

- *6EAGLEFiles*: contains files for viewing with CadSoft Eagle PCB layout software. The file called customPCB.lbr is a library file containing all library packages manually defined during the custom PCB design. The folder called custom-Tau-connector is the Eagle project folder, containing the schematic (customTauConnector.sch) and the board layout (customTauConnector.brd) files.
- *7MCUProgram*: contains files for use with Silicon Labs Simplicity Studio IDE. The folder contains a first prototype C project (CustomFLIRTau2PCB) for the custom PCB microcontroller. USB configuration information can be found in inc/usbconfig.h and inc/descriptors.h, with USB callbacks defined in src/callbacks.c. Digital data retrieval code can be found in src/digdata.c. Minimal testing completed.
- *8DUNETaskFeatureFLIRDigital*: contains the first prototype C++ DUNE Task created that is responsible for receiving uncompressed digital image frames from the custom PCB, compressing them into JPEG format and sending them as an IMC frame message to the ground station. Minimal testing completed.
- *9DUNEFullFeatureFlirDigital*: full DUNE software system with task for retrieval of digital data included.

Bibliography

- [1] Abracon. *32.768kHz Low Profile Crystal ABS10*, 2010.
- [2] Abracon. *Ceramic SMD Micro Miniature Microprocessor Crystal ABM7*, 2010.
- [3] Ricardo Bencatel. DUNE wiki. <https://github.com/LSTS/dune/wiki>, 2015.
- [4] CadSoft. EAGLE Version 7.4.0. www.cadsoftusa.com, 2015.
- [5] R Martins; J Braga; P.D. Calado. DUNE code (master branch). <https://github.com/LSTS/dune>, 2015.
- [6] Paul E. Berg & co. *USB Device Class Definition for Video Devices*, rev 1.5 edition, 2012.
- [7] J Pinto; P Dias. Neptus code (develop branch). <https://github.com/LSTS/neptus>, 2015.
- [8] FLIR. *FLIR Camera Controller GUI User's Guide*, 2014.
- [9] FLIR. *FLIR Tau 2 Electrical Interface Description Document*. FLIR, 102-ps242-41 edition, June 2014.
- [10] FLIR. FLIR Tau Camera Link Expansion Board. <https://github.com/LSTS/dune/wiki>, 2015.
- [11] FLIR. FLIR Tau PCB Wearsaver with Solder Pads. <https://www.flirshop.com/product/tau-pcb-wearsaver/Tau320-Accessories>, 2015.
- [12] FLIR. FLIR Tau VPC Module. <https://www.flirshop.com/product/tau-vpc-module/Tau320-Accessories>, 2015.
- [13] FLIR. *FLIR Tau2 & Quark Software Interface Description Document*, 2015.

- [14] FLIR. Tau 2 LWIR Thermal Imaging Camera Cores. www.flir.com/cores/display/?id=54717, 2015.
- [15] The Eclipse Foundation. Eclipse IDE for C/C++ Developers. www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/mars1, 2015.
- [16] The Eclipse Foundation. Eclipse IDE for Java Developers. www.eclipse.org/downloads/packages/eclipse-ide-java-developers/mars1, 2015.
- [17] Thor I. Fossen & Tor Arne Johansen Frederik S. Leira, Kenan Trnka. A Light-Weight Thermal Camera Payload with Georeferencing Capabilities for Small Fixed-Wing UAVs. Technical report, NTNU, 2015.
- [18] Hirose. *0.5mm Pitch SMT Board to Board Connector*, 2015.
- [19] Texas Instruments. *4-Channel ESD Solution for High-Speed Differential Interface*, 2013.
- [20] Texas Instruments. *DS90LV032A 3V LVDS Quad CMOS Differential Line Receiver Technical Datasheet*, 2013.
- [21] Stephen Kempainen. Low-Voltage Differential Signaling (LVDS). Technical report, National Semiconductor, 2002.
- [22] Silicon Labs. *AN0004: Clock Management Unit*, 2013.
- [23] Silicon Labs. *AN0016: Oscillator Design Considerations*, 2013.
- [24] Silicon Labs. *AN0046: USB Hardware Design Guide*, 2013.
- [25] Silicon Labs. *AN0013: Direct Memory Access*, 2014.
- [26] Silicon Labs. *AN0014: TIMER*, 2014.
- [27] Silicon Labs. *AN0025: Peripheral Reflex System*, 2014.
- [28] Silicon Labs. *AN0039: EFM32 Interrupt Handling*, 2014.
- [29] Silicon Labs. *AN0002: Hardware Design Considerations*, 2015.
- [30] Silicon Labs. *AN0012: General Purpose Input Output*, 2015.

- [31] Silicon Labs. *AN0042: USB/UART Bootloader*, 2015.
- [32] Silicon Labs. *AN0065: EFM32 as a USB Device*, 2015.
- [33] Silicon Labs. EFM32 Happy Gecko Starter Kit Schematic, April 2015. Document number BRD2012A.
- [34] Silicon Labs. *EFM32HG Reference Manual*, 2015.
- [35] Silicon Labs. *EFM32HG322 Datasheet*, 2015.
- [36] Silicon Labs. EFM32HG322F64 Hardware Reference, February 2015. Schematic.
- [37] Silicon Labs. Simplicity Studio. www.silabs.com/products/mcu/Pages/simplicity-studio.aspx, 2015.
- [38] LSTS. DUNE: Unified Navigaiton Environment. <http://www.lsts.pt/toolchain/dune>, 2015.
- [39] LSTS. IMC: Inter-Module Communication Protocol. www.lsts.pt/toolchain/imc, 2015.
- [40] LSTS. Neptus: Command and Control Software. www.lsts.pt/toolchain/neptus, 2015.
- [41] LSTS. Our Toolchain. <http://www.lsts.pt/toolchain>, 2015.
- [42] R Martins. IMC: Intermodule Communication API (master branch). <https://github.com/LSTS/imc>, 2015.
- [43] Jason Nash. *Tactile Switch Ta-1143 Technical Data Sheet*, 2009.
- [44] Odroid. Odroid XU4 Hardware. odroid.com/dokuwiki/doku.php?id=en:xu4_hardware, 2015.
- [45] Oracle. *Creating a GUI with JFC/Swing*, 2015.
- [46] Oracle. *Learning the Java Language*, 2015.
- [47] J Pinto. IMC Java code (master branch). <https://github.com/LSTS/imcjava>, 2015.

- [48] Segger. Debug Probes - J-Link and J-Trace. <https://www.segger.com/jlink-debug-probes.html>, 2016.
- [49] Nathan Seidle. *Designing PCBs: Advanced SMD*, 2015.
- [50] Nathan Seidle. *Designing PCBs: SMD Footprints*, 2015.
- [51] Global Connector Technology. *micro USB Receptacle Type B Surface Mount Vertical with Through Hole Shell Stakes*, 2015.
- [52] L Tremblay. Tau XP - Wearsaver Board Schematic, April 2009. DWG NO 250-0350-03 Proprietary information contained.
- [53] Kenan Trnka. Localization and Tracking of Ships and Objects Using a UAV Mounted Thermal Imaging Camera. Master's thesis, NTNU, January 2015.
- [54] USB-IF. *USB 2.0 Specification*, 2000.
- [55] USB-IF. USB Class Codes. http://www.usb.org/developers/defined_class, 2016.