CHALMERS

Architecture and Performance Evaluation of the Space Communication Protocol Proximity-1

Master of Science Thesis in Embedded Electronic System Design

MALIN ELIASSON
JOHAN HASSEL

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2015

Architecture and Performance Evaluation of the
Space Communication Protocol Proximity-1

MALIN ELIASSON.
JOHAN HASSEL.

Examiner: PER LARSSON-EDEFORS

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Abstract

The amount of data generated during missions to other planets is ever-increasing. To accommodate the requirements on communication equipment, relay communication such as lander-satellite-earth links are used. For short-range communication between a lander and a satellite, protocols such as Proximity-1 are used.

This report describes the implementation, testing and evaluation of the Data link layer of the Proximity-1 space communication protocol. The implementation, called Proximity-1 module, can establish a communication session with a remote Proximity-1 transceiver, transfer data, and reconfigure the radio link if needed.

The Proximity-1 module is entirely implemented in hardware in order to off-load work from the On-Board Computer (OBC) of the system in which the module is used. The Proximity-1 module is designed to be as flexible as possible in order to ease the task of incorporating it into a host system.

The internal design of the Proximity-1 module is divided into blocks which correspond to the sublayers defined in the Proximity-1 standard. Each block is connected using streaming handshake interface. The well-defined block functionality and handshaking interfaces means that blocks implementing additional functionality can easily be added into the design for future expansions.

In order to ensure an efficient design, the module is designed to provide an evenly distributed workload throughout the implementation. In addition, a performance enhancing acknowledgment strategy which is not part of the Proximity-1 standard is proposed and implemented. The configuration can be used to reduce the number of acknowledgment messages sent in a communication session and hence increase the effective data throughput.

Acronyms

ARQ	Automated Repeat Queuing
ASM	Attached Synchronization Marker
CCSDS	Consultative Committee for Space Data Systems
COP	Communication Operations Procedure
CRC	Cyclic Redundancy Check
DFC ID	Data Field Construction Identifier
ESA	European Space Agency
FARM	Frame Acceptance and Reporting Mechanism
FOP	Frame Operation Procedure
MAC	Medium Access Control
MER	Mars Exploration Rover
MEX	Mars Express
MIB	Management Information Base
OBC	On-Board Computer
OSI	Open Systems Interconnection
P-frame	Protocol Data Frame
PCID	Physical Channel Identifier
PLCW	Proximity Link Control Word
PLTU	Proximity Link Transmission Unit
QoS	Quality of Service
SCID	Spacecraft Identifier
U-frame	User Data Frame

Definitions

Communication channel (Also referred to as *communication link.*) Physical connection between two transceivers in a communication session.

Communication session Time period during which two transceivers are communicating with each other.

Full duplex operation Both the transmitter and receiver is active at the same time in a transceiver.

Half duplex operation A transceiver switches between using its transmitter and receiver during a communication session.

On-Board Computer The computer system that the Proximity-1 module is connected to and which configures the Proximity-1 module.

Protocol data Information and commands consumed within the Proximity-1 protocol.

Protocol data packet Data packet which contains protocol data.

Protocol frame (P-frame) Transfer frame containing protocol data.

Proximity Link Control Word (PLCW) Protocol data unit which is used in the COP process to send the status of received Sequence Controlled frames.

Remote transceiver The system that the local Proximity-1 module is sending to and receiving from.

Simplex operation Only the transmitter or only the receiver is active in a transceiver during a communication session.

Version-3 Transfer Frame A Version-3 Transfer Frame consists of a frame header and payload data, which is used to transfer data between two Proximity-1 transceivers.

User data Data provided to the Proximity-1 module from the local On-Board Computer and sent to the remote On-Board Computer.

User data packet Data packet which contains user data.

User frame (U-frame) Transfer frame containing user data.

Contents

1

Introduction

With the upcoming Mars missions, there is a need for improved radio communications. In particular, surface-to-orbit communications are required to manage data transfer between planetary rovers and satellites [1]. In previous space systems, the rover on a planet sent data directly to Earth. Since a rover has limited resources regarding the transmission rate and power consumption, data could only be sent relatively slowly and the signal was weak. A new communication protocol, called Proximity-1, was developed to improve these shortcomings [1]. Proximity-1 is a communication protocol established between two spacecrafts relatively close to each other in space, for example a rover and a satellite. Using Proximity-1, the rover can send its data to a satellite orbiting the planet, and the satellite can send the data back to Earth. Compared to rovers, satellites generally have less strict requirements on power and can afford higher transmission rates, thus increasing the amount of data received on Earth. To facilitate interoperability between protocol implementations, the Proximity-1 protocol is defined in a standard managed by the Consultative Committee for Space Data Systems (CCSDS) [2]. In future Mars missions where the importance of relay communication will grow, the Proximity-1 Space Link protocol will be an important asset for satellites and rovers at Mars [2], [3].

The Proximity-1 protocol consist of two *layers*, the Data Link Layer and the Physical layer [2]. The Data Link Layer responsible for reading data to be transmitted, attach protocol information to the data (such as the ID of vehicle that is to receive the data) and output the result as a serial bit stream to the Physical layer. The Physical layer transforms the serial bit stream into a physical signal that can be transmitted by an antenna to a remote receiver. Although CCSDS has developed a standard to make implementations of the protocol similar and compatible, there has been some interoperability and performance issues between different implementations in the past [2]. These issues often arise from differences in the Data Link Layer implementation [2]. Due to this, there is a need to develop and evaluate a more general and flexible Data Link Layer that can avoid these problems.

1.1 Purpose

This project aims to design and evaluate an implementation of the Proximity-1 Space Link protocol Data Link Layer in European Space Agency (ESA)'s next generation of rover and satellite systems. The module is designed as an IP-core that may be implemented in a wide range of systems, but it will be connected to an existing system for testing. All requirements of the Proximity-1 Data Link Layer will be met, with the exception of timing services offered by the protocol, see section 1.3. This means that the design could be used in any vehicle that needs to comply with the Proximity-1 standard.

In order to avoid efficiency and interoperability issues, the implementation should provide as much flexibility and configurability as possible and also allow for future expansions. However, it is simultaneously preferable with a hardware implementation over software, since hardware can offload work from the system's On-Board Computer (OBC). Therefore, as much as possible should be implemented in hardware which in turn may limit flexibility. The challenge therefore lies in providing flexibility and configurability, and at the same time make the implementation as self-contained as possible.

1.2 Research Questions

The project will aim to answer the following questions:

1. How can the Proximity-1 protocol Data Link layer be implemented in hardware, in a performance efficient way, and still provide flexibility to the system in terms of operation and future expansions?
2. What functionality of the Data Link Layer is preferably placed in software and why?
3. How can Data Link Layer interoperability and performance issues be avoided while the Data Link Layer is mainly implemented in hardware?

The third research question deals with known interoperability and performance issues of the Data Link Layer. These issues, which have been identified by CCSDS, are described in section 2.3.

1.3 Limitations

The focus of this project is mainly to implement and analyze the Data Link Layer of the Proximity-1 protocol, with focus on the known interoperability and efficiency issues of the layer [2]. The Physical Layer of the protocol will not be implemented, but the interface between the Data Link layer and Physical layer will be designed. In addition, the timing services offered by the protocol will not be implemented, (see [4] for details about the timing services offered by the protocol). The timing services provide data time-stamping and time-synchronization functionality, but this is not essential for evaluating the efficiency of transmission and reception of data, neither are they considered a limiting factor in terms of flexibility. However, the interfaces and registers needed by the timing services will be implemented in the design in order to make it easier to add the functionality in the future. While the timing services are considered a requirement in Proximity-1 implementations, there is no requirement that they are used in communication sessions. The data transfer functionality of Proximity-1 functions independently of these services.

While power is in limited supply in space applications, the power dissipation of the module is assumed to be several magnitudes lower compared to the antennas. For example, the Mars Exploration Rover has a transmit power of 12 W [5]. This means that a more efficient data transfer link that reduces the operation time of the antennas will yield lower power dissipation. Therefore, focusing on optimizing the efficiency of the protocol implementation will yield reduced power, more-or-less regardless of the power dissipation of the module when active. Furthermore, when not in use, the power dissipation of the module is assumed to be zero since the module can be clock-gated. Thus, the power dissipation of the module implementation is not considered in the design.

1.4 Thesis Outline

In chapter 2, an overview of the Proximity-1 protocol and details of the Data Link Layer are given. Chapter 3 details the methods used in the implementation of the Data Link Layer such as coding style and tools. After this, in chapter 4, the Data Link Layer implementation is described. In chapter 5 the testing and evaluation of the functionality and performance of the implementation can be found. Chapter 6 provides a discussion about the implementation and relates back to the research questions of the thesis work. Chapter 7 concludes the paper and acknowledgments can be found in chapter 8.

2

Theory - Proximity-1 Protocol Standard

This chapter begins with an overview of the Proximity-1 protocol and its advantages. The rest of the chapter describes the Data Link Layer of the protocol, which is the focus for this report.

2.1 Proximity-1 Background

Proximity-1 is a communication protocol used for communication between space vehicles that are relatively close to each other in space [2]. Proximity-1 is used to set up and close down a *communication session*, and to transmit and receive data during the session. A communication session is the time period during which two transceivers are communicating with each other. Data to be transmitted during the session is supplied to the Proximity-1 module by a system referred to as the *user*. In this project, the user is assumed to be an On-Board Computer (OBC), and hence this document will mainly refer to the user as the OBC. The Proximity-1 protocol can be viewed as consisting of two parts, one part which handles the transmission of data and one part which handles the reception of data.

In the transmission part, the Proximity-1 implementation receives data from its OBC and attaches a header, a checksum and other protocol-specific information to the data. A header is attached to the beginning of the data, and specifies the data format, data length and an identification of the intended receiver of the data. The data is serialized and transmitted to the remote transceiver on the receiving vehicle.

In the reception part, the Proximity-1 implementation receives data from a remote transceiver. Proximity-1 validates the data using the header and checksum and sends it to its OBC [2].

A typical user scenario for Proximity-1 is when a rover on Mars is going to transfer data back to Earth. Instead of communicating directly with Earth, the rover transfers data to a satellite orbiting Mars. The satellite then propagates the data to Earth. The satellite can also receive commands issued from Earth and forward these to the rover. This type of communication is referred to as relay communication, as the satellite relays data between the rover and Earth.

In 2004, the Proximity-1 protocol was used for the first time to transfer data between NASA's Mars Exploration Rover (MER) Spirit and ESA Mars Express (MEX) orbiter at Mars [1]. Since then, the Proximity-1 protocol has provided the functionality for relay communications between rovers and satellites around Mars [6], [2].

The benefits of using a relay communication protocol such as Proximity-1 instead of transmitting data directly back to Earth are many [3]. From the rovers point of view, the performance of a communication link to a nearby satellite is significantly better than a long distance communication link to Earth. The relation between the performance of a communication link and the distance, R, between the two communication points is given by 2.1 [3]

$$Performance \propto \frac{1}{R^2} \tag{2.1}$$

Equation 2.1 shows how difficult long-range communications are, since the performance decreases quadratically with increasing distance. To enable a reliable communication link between Earth and a vehicle at Mars, heavy and power-hungry communication equipment is necessary. However, the larger the mass of a rover, the more difficult and expensive it becomes to shuttle and land on the Mars surface. This limits the rover's capability of having efficient communication equipment.

Satellites, on the other hand, can have a larger mass and higher power consumption than rovers, and therefore can afford the more efficient communication equipment [3]. Since the distance between the satellite and the rover is much shorter than the distance between the rover and Earth, the rover can obtain an efficient communication link with the satellite using equipment with smaller mass and power consumption. The satellite, being less restricted by mass and power, can then transfer data back to Earth. In the future, relay communication is expected to be even more important as the instruments that collect data on the rovers are expected to be more efficient and collect even more data. A reliable and efficient communication link which transfers the collected data back to Earth will be increasingly important to be able to handle all the collected data. Another important benefit of using relay communication is that the availability of communication and data transfer increases for the rover. Depending on where the rover is situated on Mars, Earth might not be in view. However, the satellite will always be known to pass above the rover at regular intervals [3].

Proximity-1 allows for several rovers to communicate with the same satellite [2]. This is beneficial from a cost perspective since many rovers can use the same satellite. Proximity-1 has many configurable parameters which allows for a flexible design. However, the implementation of some fundamental functions are left to the designer, which means that implementations of the protocol may differ [2]. For example, an implementation is required to support *full duplex* operation, i.e. both the transmitter and receiver is active at the same time [4]. However, it is also possible for an implementation to additionally support *half duplex* (switching between transmitting and receiving operations during a communication session) and *simplex operation* (allowing only transmission or only reception of data during a communication session) [4]. In addition, not all functionality nor interfaces of the Proximity-1 protocol are specified in detail which also can create differences between implementations. For example, the Proximity-1 standard gives some freedom to the designer to decide some of the features of the interface between Proximity-1 and the OBC.

2.2 Proximity-1 Functional Overview

The functionality and services provided by the Proximity-1 protocol are described in CCSDS recommended standards [4], [7], [8]. The Proximity-1 protocol implements the two lowest layers of the Open Systems Interconnection (OSI) model communication standard, the Data Link Layer and Physical Layer [2]. According to the OSI model, the Data Link Layer handles the conversion of data packets into data frames which can be sent and received by the system [9]. Data packets contains data which is to be sent to the remote receiver. Protocol specific information is added to the data packets to form data frames. A data frame hence contains data from data packets and some additional information, the added information depends on the protocol used. In addition, the Data Link Layer is responsible for the configuration of parameters such as physical addressing and synchronization between the communicating transceivers. The Physical Layer of the OSI model manages the transmission and reception of data over a physical medium [9].

The Data Link Layer in the Proximity-1 protocol consist of five sublayers; the I/O Sublayer, the Data Service Sublayer, the Frame Sublayer, the Coding and Synchronization Sublayer and the Medium Access Control (MAC) sublayer [2]. Figure 2.1 shows a layout view of the protocol layers.

To the right in Figure 2.1 the data format used in each sublayer is specified. Data packets are supplied to the Proximity-1 module by the OBC. The I/O Sublayer creates Version-3 Transfer Frames (or transfer frames in short) from the supplied data packets. A transfer frame contains data and a transfer header with information about how the transfer frame should be handled in the remote transceiver. The transfer frames are transported from the I/O Sublayer through the Data Service Sublayer and Frame Sublayer until it arrives in the Coding and Synchronization Sublayer. In the Coding and Synchronization Sublayer, an identifier (called Attached Synchronization Marker (ASM)) and a checksum (CRC) is attached to the transfer frame. The ASM, checksum and transfer frame forms a data unit called PLTU. The Coding and Synchronization Sublayer outputs PLTUs as a serial bit stream to the Physical Layer. The Physical Layer takes the data bits and transforms them into a physical signal that can be sent by an antenna. A summary of the contents of each data unit is in Section 2.2.6 and Section 2.2.7.

As seen in Figure 2.1, the OBC can communicate and interact with the Proximity-1 protocol through the I/O Sublayer and the MAC Sublayer. The OBC sends and receives

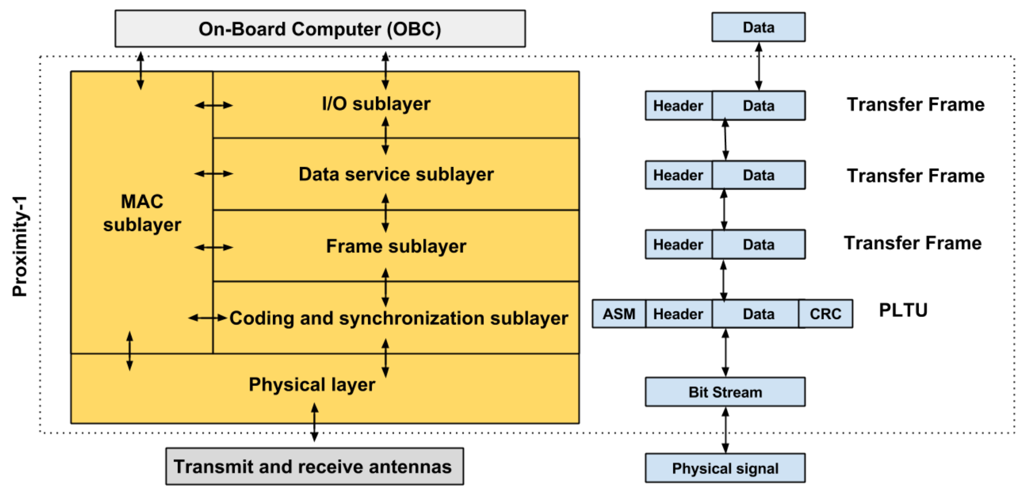

Figure 2.1: Proximity-1 protocol layers and data formats.

data through the I/O Sublayer, while commands and configurations of the protocol can be sent to the MAC Sublayer. The MAC controls the operation of the protocol and hence interacts with all the other layers. The Physical Layer transmits and receives data through transmit and receive antennas. The subsequent subsections will describe each sublayer of the protocol in more detail. Since the protocol deals with both the transmission and reception of data, the explanation of each sublayer is divided into a *transmission part* and a *reception part*.

2.2.1 I/O Sublayer

The transmission part of the I/O Sublayer provides an interface to the local OBC where data to be sent over the Proximity-1 link can be supplied [4]. The OBC supplies data packets and the information needed to transmit the packets to the remote transceiver. For instance, the ID of the module in the remote transceiver that is to receive the packets (called Port ID) is supplied by the OBC. Data packets that should be delivered to the remote transceiver's OBC is referred to as *user data packets*. Data packets that contain configurations to be used inside the Proximity-1 protocol at the remote transceiver are referred to as *protocol data packets*. The I/O Sublayer forms Version-3 Transfer Frames, or simply transfer frames, from the supplied data packets and information. A transfer frame contains the supplied data and also a transfer header. The header contains the information supplied by the OBC (for example, the Port ID) and some additional information formed by the I/O Sublayer, such as the number of bytes the frame contains. The header information hence provides information about the frame for the remote transceiver. See section 2.2.6 for a detailed description of the Version-3 Transfer Frame. Large data packets can be divided into multiple transfer frames, and several small data packets can be sent in one transfer frame [4].

The Proximity-1 protocol provides two different Quality of Service (QoS) modes for the data to be sent, Sequence Controlled and Expedited. The Sequence Controlled service provides reliable data transfer through a Go-back-n Automated Repeat Queuing (ARQ) algorithm. In this algorithm, acknowledgment of received data and retransmission of lost data is used to ensure that data is received correctly. The ARQ algorithm is handled in the Data Service Sublayer, see section 2.2.2. When the Expedited service is used, no retransmission of data is provided and the protocol implementation does not receive any acknowledgment when (or if) the data reaches the remote transceiver. Expedited data packets have priority over Sequence Controlled data packets.

The I/O Sublayer places the Expedited transfer frames and Sequence Controlled transfer frames in separate buffers where they stay until they are issued for transmission. Frames are taken from the buffers by the lower Data Service Sublayer.

The transmission part of the I/O Sublayer is also responsible for notifying the OBC of the transmission status of data packets. The OBC is notified when a complete Expedited data packet has been transmitted by the Physical Layer and when a complete Sequence Controlled data packet has been acknowledged by the remote transceiver.

The reception part of the I/O Sublayer receives frames from the Data Service Sublayer. The frames are assembled into the original data packets and supplied to the OBC. Only complete data packets are delivered to the OBC. This could become an issue for data packets that use the Expedited QoS, since frames that are lost are not re-transmitted. However, data packets using the Expedited QoS are typically managed by communication layers above the Data Link Layer of Proximity-1 [4].

2.2.2 Data Service Sublayer

The transmission part of the Data Service Sublayer receives Expedited and Sequence Controlled frames from the I/O Sublayer via the Sequence Controlled and Expedited buffers [4]. The Data Service Sublayer handles the Go-back-n ARQ algorithm for the Sequence Controlled frames using a Communication Operations Procedure (COP) process. The COP consists of two processes, the Frame Operation Procedure (FOP) process which handles the transmission of frames in the transmission part and the Frame Acceptance and Reporting Mechanism (FARM) process which generates acknowledgments for received frames in the receiver part.

The FOP process keeps track of sent Sequence Controlled frames and their acknowledgments. Sequence Controlled frames are identified using their frame sequence number contained in the header of the frame. An acknowledgment is a protocol frame called Proximity Link Control Word (PLCW) which contains the expected frame sequence number, that is, the sequence number of the last acknowledged frame plus one. Frames which are sent but not acknowledged are saved in a buffer. If a frame need to be resent it can be sent again from the buffer, and if a frame is acknowledged by the remote transceivers FARM process it can be removed from the buffer. The maximum number of frames that can be in flight at the same time without acknowledgment is determined by the COP window size set by a Management Information Base (MIB) parameter (see section 2.2.9).

The FARM process checks that Sequence Controlled frames arrive in-order. If a frame arrives out-of-order, it will be discarded. The FARM process informs the remote transceiver's FOP process of the frames it has accepted by sending an acknowledgment in the form of a PLCW. The PLCW contains the sequence number of the next Sequence Controlled frame the FARM expects, that is, sequence number of the frame that should be transmitted or retransmitted next.

The FOP process notifies the OBC when a Sequence Controlled frame has been acknowledged or a Expedited frame has been transmitted [4].

2.2.3 Frame Sublayer

The transmission part of the Frame Sublayer is responsible for prioritizing which frame that should be sent next and supplies the selected frame to the underlying Coding and Synchronization Sublayer [4]. The Frame Sublayer prioritizes between Sequence Controlled frames, Expedited frames, PLCWs and MAC generated protocol frames.

In the reception part, the Frame Sublayer receives frames from the Coding and Synchronization Sublayer. The frame is validated by checking that it is a valid Version-3 Transfer Frame, and the source and destination identifier in the frame header. Based on the content of the frame, the Frame Sublayer sends the frame to the Data Service Sublayer or MAC Sublayer [4]. Frames that contain user data are sent to the Data Service Sublayer. Frames that contain protocol data, i.e protocol information or configuration, are sent to the MAC Sublayer.

2.2.4 Coding and Synchronization Sublayer

In the transmission part, the Coding and Synchronization Sublayer receives a transfer frame from the Frame Sublayer [8]. A Cyclic Redundancy Check (CRC) and ASM are attached to the frame to form a Proximity Link Transmission Unit (PLTU) (see section 2.2.7). The PLTUs are then supplied to the Physical Layer, which expects a constant input data rate. When no new PLTUs are available, a bitstream consisting of PLTUs and idle data is generated by the Coding and Synchronization Sublayer. The idle data is the static hexadecimal sequence ”352EF853”, and is inserted into the bitstream when no PLTUs are available in order to provide the constant data rate required by the Physical Layer. Idle data is also inserted at the start and end of a communication session. The generated bitstream can optionally be encoded using LDPC code or Convolutional code before it is sent to the Physical Layer.

In the reception part, the Coding and Synchronization Sublayer reads a stream of data from the Physical Layer. The Coding and Synchronization Sublayer identifies and delimits frames in the data stream by searching for the ASM. When a frame has been delimited, the CRC is checked to validate that the frame contains no transmission errors before it is sent to the Frame Sublayer.

In addition to the procedures described above, the Coding and Synchronization Sublayer is responsible for storing the timestamps of sent and received transfer frames when this is enabled for timing synchronization services. An implementation can also choose to add an error-correcting code, such as Reed-Solomon, to the data [4].

2.2.5 Physical Layer

In the transmitting part, the Physical Layer receives a stream of data from the Coding and Synchronization Sublayer and modulates it onto a radiated carrier for transmission to the remote transceiver [7]. In the reception part, the Physical Layer receives a data stream from the remote transceiver over the physical channel and outputs this data stream to the Coding and Synchronization Sublayer.

Transceiver configurations used in the Physical Layer such as frequency, modulation and data rate are set by parameters in the MAC Sublayer. Before a communication session is started, the Physical Layer is responsible for obtaining a carrier signal. The Physical Layer informs the MAC Sublayer when a carrier has been acquired and when a data stream is being received successfully on the communication channel [7].

2.2.6 Version-3 Transfer Frame

Figure 2.2 shows the Version-3 Transfer Frame used in the Proximity-1 protocol. The maximum total size of a transfer frame is 2048 bytes, where 5 bytes are reserved for the header and the rest is data [4]. The maximum size of a transfer frame can be varied using a MIB parameter (see section 2.2.9). In Figure 2.3 the header of the transfer frame is specified in more detail.

Header (5 bytes)	Data (maximum 2043 bytes)

Figure 2.2: Version-3 Transfer Frame

	Transfer frame version number	Quality of Service (QoS) Indicator	PDU Type ID	Data Field Construction Identifier (DFC ID)	Spacecraft Identifier (SCID)	Physical Channel Identifier (PCID)	Port Identifier	Source/ Destination Identifier	Frame Length	Frame Sequence Number
Number of bits	2	1	1	2	10	1	3	1	11	8

Figure 2.3: Version-3 Transfer Frame header

The different fields of the transfer frame header are as follows [4]:

- ***Transfer Frame Version Number***. This number is set to the binary value '10'. It is used to identify the frames as Version-3 Transfer Frames.
- ***Quality of Service (QoS)*** Indicates the QoS of the frame; Expedited or Sequence Controlled.
- ***PDU Type ID*** Indicates if the data in the frame contains user data or protocol data.
- ***Data Field Construction Identifier (DFC ID)*** Indicates the data format of the frame, for example if the data is a segment of a larger data packet or contains an entire data packet.
- ***Spacecraft Identifier (SCID)*** Identifies the source or destination spacecraft, depending on the value of the Source-or-Destination Identifier field.
- ***Physical Channel Identifier (PCID)*** Determines which physical transceiver the frame should be sent to. Allows two transceivers to be in operation simultaneously.
- ***Port ID*** Determines what output port in the remote transceiver the I/O Sublayer should supply the received data packet to.
- ***Source-or-Destination Identifier*** Tells if the SCID field of the header contains the source or destination spacecraft identifier.
- ***Frame Length*** Length, in bytes, of the transfer frame (header and data)
- ***Frame Sequence Number*** The sequence number is a number assigned to each frame. The number assignment starts at zero for the first frame and then increments by one for each frame. The sequence number is used in the ARQ algorithm to identify frames and validate that they arrive in order.

2.2.7 Proximity Link Transmission Unit

In Figure 2.4 a Proximity Link Transmission Unit (PLTU) can be seen. The PLTU consist of a Version-3 Transfer Frame, a CRC checksum and an ASM [8].

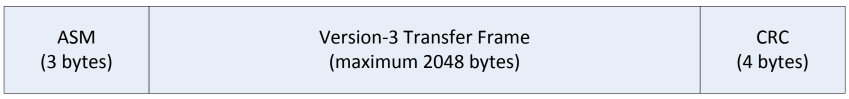

Figure 2.4: PLTU

The CRC is calculated based on the whole Version-3 Transfer Frame, the calculations and formulas can be found in annex C of [8]. Upon reception of a PLTU, the CRC is used to do an error check of the received data. The ASM is the static hexadecimal sequence "FAF320", which is used to find frames in the received bitstream.

2.2.8 MAC Sublayer

The MAC sublayer controls the operation of the Proximity-1 protocol during a communication session. It handles *directives* (commands) received from the OBC or the remote Proximity-1 transceiver. It is responsible for the establishment and termination of a communication session and it generates control signals for the other Data link sublayers and the Physical Layer [4]. The MAC also controls and uses many of the configurable parameters of the Proximity-1 protocol. The configurations can be divided into two sets, one set of configurations that only can be changed by the local OBC (called MIB parameters, see section 2.2.9) and one set that both the local and remote OBC can modify. The MIB parameters configures the local Proximity-1 module, they are for example used for timing different activities within the MAC Sublayer (see section 2.2.9). Almost all the MIB parameters should be kept static during a communication session and hence need to be set by the local OBC before the session is initialized. The configuration parameters that both the local and remote OBC can modify are transmitter and receiver configurations, such as data rate and choice of encoding of the data. These parameters are session dynamic (i.e they can change during a communication session) and are set using directives (see section 2.2.9).

The MAC Sublayer also contains buffers to store transfer frames send and receive times. These time values are part of the timing services provided by the Proximity-1 protocol [4].

2.2.9 MAC Persistent Activities

The MAC Sublayer uses *persistent activities* in order to provide reliable data transfer between two Proximity-1 transceivers when initializing a communication session or when changing communication parameters of a communication session [4]. These actions requires some directives (commands) to be exchanged between the two Proximity-1 transceivers. A reliable data transfer is needed since the MAC uses the unreliable Expedited service to send protocol frames with directives.

A persistent activity provides reliable data transfer since a directive sent in a persistent activity requires a response message from the receiver of the directive. The MAC Sublayer handles three persistent activities:

1. ***Hailing***. Establishment of a communication session.
2. ***Comm Change***. Perform transmitter and receiver parameter changes during a communication session.

3. ***Resynchronization***. Resynchronization of the FOP and FARM processes when the sender and receiver of Sequence Controlled frames disagree on the frame sequence number.

When the MAC sends a directive in a persistent activity it waits for a response message from the remote transceiver for a certain time called *waiting period* (the value of the waiting period of a persistent activity is configured via MIB parameters). If the response is not received within the given waiting period, the activity is retried if the *lifetime* of the activity not has expired (the value of the lifetime of a persistent activity is also given in MIB parameters). The lifetime value can either be a time which is counted downwards to zero, or a number which indicates how many times an activity should be retried. If no response is received within the lifetime of an activity, the activity is terminated and the local OBC is notified that the activity did not succeed. If a response is obtained during the lifetime of the activity, the OBC is notified of the success of the persistent activity [4].

Management Information Base Parameters

Proximity-1 uses Management Information Base (MIB) parameters to control several operational configurations of the protocol [4]. Examples of MIB parameters are COP transmission window size, local spacecraft ID, and waiting periods and lifetimes of different persistent activities. The MIB parameters are set by the local OBC. Many of the MIB parameters are session static and hence need to be set by the local OBC before a communication session is established. All the MIB parameters defined for the Proximity-1 protocol can be found in Annex C of [4].

Local Directives

The commands sent from the local OBC to the local Proximity-1 module are called *local directives* [4]. The MAC Sublayer receives these commands and is responsible for configuring the other local sublayers based on the command, and if necessary send a directive with configurations to the remote transceiver. The local directives that can be issued from the local OBC are listed below.

- ***Set Mode***. The Set Mode directive commands the Proximity-1 protocol to go to a certain operational state. There are four options available:
  - ***Inactive***. Put the Proximity-1 into an inactive state where no data is sent or received. The MAC also resets many internal variables in this state.
  - ***Active***. Put the Proximity-1 into an active state where data can be sent and received.
  - ***ConnectingL.*** Put the Proximity-1 into a listening state where the Proximity-1 module waits for a hailing message (communication establishment message) from a remote transceiver.

  - ***ConnectingT***. Put the Proximity-1 into a transmitting state where the Proximity-1 module tries to initialize a communication session by sending a hailing frame (i.e. a communication establishment frame) to a remote transceiver. The hailing message contains communication session configurations for the transmitter and receiver. It is made up of the directives Set Transmitter Parameters, Set Receiver Parameters and, optionally, Set PL Extensions (see section 2.2.9). The hailing frame is transmitted over a default link configuration.

- ***Set Initialize Mode***. When the MAC receives this directive, it goes to its Inactive state and resets the ARQ COP processes.
- ***Load Communication Value Buffer***. The OBC uses this directive to set the configuration of the transmitter and receiver. When Hailing or executing a Comm Change, these communication values are sent in remote directives to the remote Proximity-1 transceiver in order to configure it. The directives that are sent to the remote transceiver are called Set Transmitter Parameters, Set Receiver Parameters and Set PL Extensions.
- ***Local Comm Change***. This directive initializes a Comm Change activity (i.e an activity performed in order to change communication session parameters such as data rate and transmission frequency). The MAC generates remote directives containing the new communication session parameters and sends these to the remote Proximity-1 transceiver. The directives sent are Set Transmitter Parameters, Set Receiver Parameters and Set PL Extensions (see section 2.2.9).
- ***Local No More Data***. The OBC sends this directive to the MAC when it has no more data to send. The MAC propagates the information to the remote Proximity-1 by sending the remote directive Remote No More Data.
- ***Set Duplex***. This directive configures the local Proximity-1 module to use either full duplex, half duplex or simplex operation.
- ***Set Receiving SCID buffer***. This directive makes it possible for the local OBC to set a Spacecraft ID (SCID) which can be used to validate received frames. The SCID contained in a received frame header may be compared to the SCID value loaded by the OBC in order to validate the frame.
- ***Read Status***. This directive enables the OBC to read the status of different Proximity-1 configurations and operations.

Remote Directives

Commands that are sent to a remote Proximity-1 transceiver are called *remote directives* [4]. The remote directives are formed within the MAC Sublayer and are sent to the remote transceiver where the remote MAC Sublayer handles them. The remote directives available in the Proximity-1 protocol are listed below.

- ***Set Transmitter Parameters***. In this directive the configurations for frequency, data rate, encoding and modulation of the remote transmitter are sent.
- ***Set Receiver Parameters***. In this directive the configurations for frequency, data rate, encoding and modulation of the remote receiver are sent.
- ***Set PL Extensions***. The Set PL Extensions directive contains transceiver configurations that are needed by transceivers which have additional functionality not required by the Proximity-1 standard. The directive can hence be used to set those optional extra configurations. For example, the directive can be used to indicate if error correcting code such as Reed-Solomon should be used when sending and receiving frames.
- ***Set Control Parameters***. This directive is used to send three types of information to the remote Proximity-1 transceiver. First, it is used to configure and switch between full duplex, half duplex and simplex operation. Second, it is used to notify the remote transceiver that the local OBC has no more data to send. Third, it can be used to initialize timing sampling of frames, in which the send and receive time of frames are stored.

2.3 Potential Functionality and Performance Issues of Proximity-1

Some potential functionality, interoperability and performance issues of Proximity-1 have been identified by CCSDS [2]. Since this report focuses on the Data Link Layer of the protocol, only the issues related to the Data Link Layer will be covered.

2.3.1 Choice of Data Packet Size and Quality of Service

One thing to keep in mind when using the protocol is the choice of data packet size and Quality of Service [2]. For example, if a large data packet is sent using the Expedited service the performance can drop. The reason is that if one or more frames are lost, the whole data packet is lost since only complete data packets are delivered to the OBC. To avoid the problem, the OBC should take the maximum frame length into account when deciding the size of data packets. It is preferable to segment data into several data packets instead of sending everything as one large data packet for the Expedited service [2]. If large data packets need to be sent, the Sequence Controlled service is likely a better choice since it automatically retransmits lost frames.

2.3.2 Completion of Hailing Activity With No Data to Send

Another implementation issue concerns the hailing process. Many implementations signal the completion of a hailing activity when the first transfer frame is received over the communication channel [2]. In cases where the responder of the hailing activity does not have any data to send, it will only radiate a single transfer frame at the end of the hailing activity. If the initiator of the hail has not locked onto the data stream in time to receive the single frame, it will never receive the frame and the hail will fail. To avoid this problem, implementations should make sure that enough time is spent waiting before the first transfer frame is sent in order to make sure the initiator have locked onto the data stream. Another option is to configure the waiting time during the hailing activity and to send two transfer frames instead of one at the end of a hailing activity [2].

2.3.3 Frame Size, Window Size and Data Rate Combinations.

When certain combinations of transfer frame size, transmission window size, transmission data rate and reception data rate are used, the efficiency of the link can suffer significantly [2]. The problem arises when Sequence Controlled frames are sent, and when one side of the communication link sends slower than the other during full duplex operation. The fast side can send frames without receiving any acknowledgment up until it reaches the transmission window size. At this point, the sender stops sending new frames and instead starts retransmitting already sent frames until it receives an acknowledgment from the slower side. This can happen if the slow sending side is busy sending a large frame, and due to the low data rate, it will take a long time before a PLCW can be sent acknowledging the received Sequence Controlled frames.

This problem can be avoided if the transmission data rate, reception data rate, frame size and window size are carefully chosen in order to make sure that the acknowledgment arrives before frames start to be retransmitted. In addition, an implementation that supports the acknowledgment of several frames in one PLCW can reduce the impact. In a basic implementation, one PLCW acknowledges one frame. This would make the problem described above even worse, since a PLCW needs to be sent from the slow-sending side for every frame received. In a more advanced implementation, the most recently received frame should be the one acknowledged in the PLCW, thus acknowledging all previously received frames as well [2]. Hence, referring back to the example described above, all the frames that are received on the slow-sending side while it is transmitting a large frame can be acknowledged together in only one PLCW after the large frame has been sent.

2.3.4 Test the whole Communication Chain

CCSDS recommends that a Proximity-1 implementation should be tested together with the whole communication chain that it is supposed to be part of [2]. If this is done, problems in the interfaces between Proximity-1, the OBC and the transceiver can be detected.

3

Method

In this chapter, the functional goals that the Data Link Layer implementation should fulfill are described and motivated. A description of the coding style adopted and the tools used to implement and test the design can also be found in this chapter.

3.1 Implementation Goals

To be able to motivate design decisions during the development of the Proximity-1 module prototype, we present a number of goals that the system should fulfill which relates to the research questions in 1.2. The system should:

1. Have a light footprint on the OBC.
2. Achieve efficient data transmission and reception links.
3. Be flexible in terms of configuration and expansion.

The first goal is intended to minimize performance hits on the OBC while the Proximity-1 module is in use. This is very important in order to ensure module flexibility, since it in turn minimizes interference with other system processes. The prototype should thus remain as standalone as possible with respect to the rest of the OBC and host system. This means that that our design is implemented solely in hardware using VHDL. An exception to this is the packet assembly of the I/O Sublayer, see section 4.4.11.

While the second goal heavily relies on transmission factors such as data rate and transmission error rate, it is important to consider common issues of the Proximity-1 protocol such as the ones discussed in Section 2.3. An efficient data link should ideally only be constrained by the transmission medium (i.e. data rate and packet loss), while the underlying transceiver system should not be a bottleneck. While some pitfalls must be taken into consideration and avoided by the module user, other issues can be mitigated by architectural choices during the system design.

The third goal ensures compatibility with existing implementations of Proximity-1, but also a forward-compatibility for future missions. The MIB parameters in Section 2.2.9 provide Proximity-1 implementations with high configurability and thus compatibility with other implementations. To further improve forward-compatibility especially, the module should be designed with expansion in mind. That way, special features required for certain missions such as data encoding and error correction can be added with low effort from designers.

3.2 Coding Guidelines for Space Application

Since the implemented module is to be used in space, certain considerations are needed regarding the coding style. In space, one must take into account the probability that radiation and high-energy particles interfere with the hardware and for example cause bit flips in registers, interrupt a signal or trigger a signal. The design must take these events into consideration, and the primary goal is that an outside trigger should not cause the hardware module to get stuck in an undefined state. With this in mind, we have adopted several coding guidelines recommended by RUAG Space AB. These coding guidelines make the VHDL code more robust and less sensitive to such unwanted triggers. The coding guidelines are outlined in the following sections.

3.2.1 Define All States of a State Machine

Because the states of a state machine are generally defined by a number of bits, the number of states should be a power of two in order to avoid the possibility of the state machine ending up in an undefined state. This ensures that all bits in the state value are utilized, and thus that all possible values correspond to a defined state. If the number of defined states is not a power of two, the state machine could end up in an undefined state. For example, radiation may result in an unwanted bit flip that makes the state register assume a value which does not correspond to a defined state. This could in turn cause the state machine to become stuck in an undefined state. However, if all states are defined, the state machine will always end up in a known state. It might be the wrong state, but at least it does not give undefined behavior.

Since a state machine is restricted to using 2^N states, some states might not be of any functional use. In our implementation, these unused states only force the state machine to return to its initial state. This means that if a bit flip makes the state machine end up in one of the unused states, the state machine will go back to its initial state.

3.2.2 Handshake Interfaces Between Blocks

When different blocks in the implementation communicates with each other, it is important to use a robust communication interface. One common example of communication between blocks is when one block is to transfer data to another block. The block sending the data might place the data in one register and then set a signal (lets call it ”Data Valid”) to true to notify the other block that new data is available. If the Data Valid signal only is *pulsed*, i.e set to true for one clock cycle, the other block might miss the signal if for example radiation interrupts the signal. The sending block is not aware of that the receiving block has missed the Data Valid pulse, and might overwrite the data with new data.

To avoid this problem, handshaking interfaces are preferred between blocks. In a handshaking interface one more signal is needed, we call this signal ”Data Ready”. The Data Ready signal is controlled by the receiving block. The Data Ready signal is set to true by the receiving block when it has received the Data Valid true signal. A typical transmission of data in this handshaking interface is carried out as follows. When the sending block has new data to send, it places the data in the data register and sets it Data Valid signal to true. When the receiving block sees the Data Valid true signal, it sets its Data Ready signal to true. When the sending block sees the Data Ready true signal it knows that the receiving block has acknowledged the Data Valid true signal, and the sender sets its Data Valid signal back to false. When the receiving block has collected the data from the data register it sets its Data Ready signal to false to signal that it is ready to receive new data. In order to avoid the sending block to send new data before the receiving block has handled the previous data, the sending block must check that the Data Ready signal has transitioned back to false before it writes new data to the data register. A more in-depth description of these types of interfaces is in Section 4.2.1.

3.3 Tools

The protocol implementation module is developed in hardware using VHDL. The code is verified using VHDL test benches in ModelSim 10.3 and synthesized using Synopsys Synplify (version F-2012.03-SP2) and Xilinx PlanAhead 14.7. The final module prototype is connected to a host system with a processor acting as the OBC, and the prototype and OBC is synthesized to a Xilinx Virtex-6 FPGA. The scripting language Tcl is used to write tests for the FPGA prototype.

4

System Implementation

This chapter details the implementation of the Proximity-1 Data Link Layer prototype. The prototype is also referred to as Proximity-1 module. As mentioned in section 3.1, the implementation is done solely in hardware using VHDL. The design is divided into several hardware blocks. The functionality of each block and their interfaces are described in this chapter. The interfaces between the Proximity-1 module and the On-Board Computer (OBC) are also described.

4.1 OBC Interface

There are three types of interfaces used for communication between the Proximity-1 prototype and the OBC: a data input, a data output and a register interface. These interfaces are described in the following sections. An example of how the Proximity-1 module can be connected to a host rover system can be seen in Figure 4.1.

4.1.1 Data Input

The data input is the interface through which the OBC supplies data to the Proximity-1 module to be sent to the remote transceiver. Data to be sent is supplied as a packet. A packet consists of the data to be sent and a packet header which is prepended to the data. The header contains information needed by the Proximity-1 module to form the frame header (see section 2.2.6). The information supplied in the packet header can be divided into the PDU Type ID, the DFC ID, the Port ID and the Source-or-Destination Identifier (see section 2.2.6 for an explanation of these). Additional information needed by the Proximity-1 module to form the frame header, such as Spacecraft ID, are part of the configuration registers (see section 4.1.3).

The data packet is formatted according to CCSDS packet transfer protocol [10]. Such a packet consist of four header bytes followed by data, see figure 4.2. The three first header bytes are not used within the Proximity-1 prototype and are therefore don't

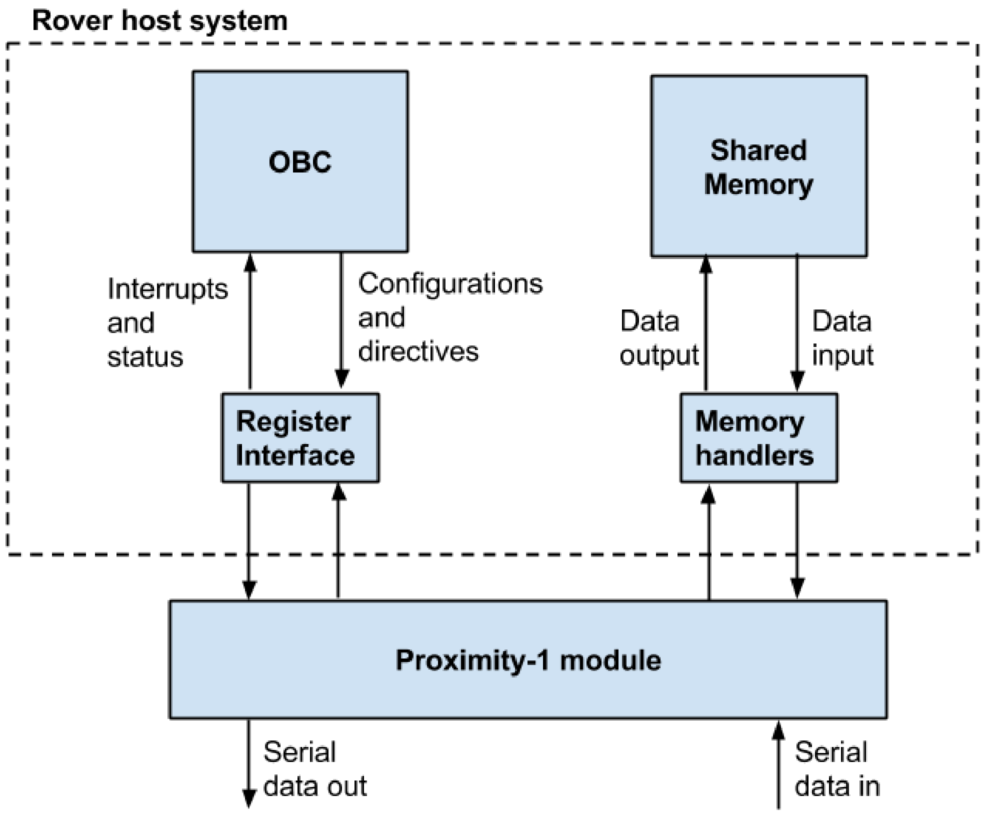

Figure 4.1: Proximity-1 module connected to host rover system

cares. The fourth header byte, the application byte, contains the header information used within the Proximity-1 module.

A typical set up that can be used to send data packets to the Proximity-1 module includes a memory-handling module and some data formatting of the OBC. The OBC places data to be sent into data packets formatted according to figure 4.2. The packets are placed in memory. When the time comes for the data packets to be transmitted, the OBC issues a send request to a memory-handling module which is responsible for reading the data packets from memory and sending them to the Proximity-1 module.

4.1.2 Data Output

The data output is the interface through which the OBC collects data from the Proximity-1 module which have been received from the remote transceiver. The output data is Version-3 transfer frames (see section 2.2.6, with an additional *router byte* prepended to each frame. The router byte is a one-byte number which is intended to be used

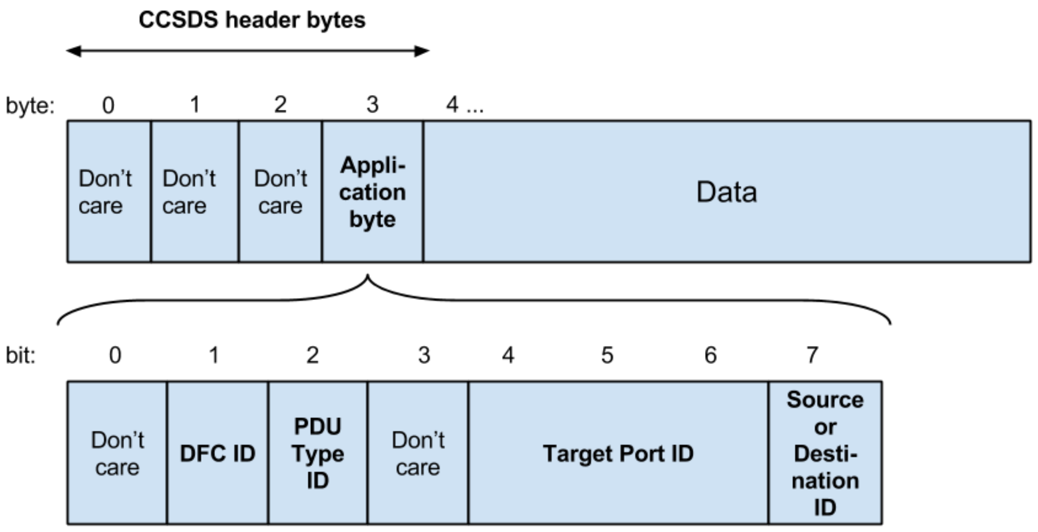

Figure 4.2: CCSDS data packet

in a router network outside of the Proximity-1 module to select a target module that should receive the frame. The router byte is determined by mapping the Port ID of the frame header to a value in a look-up table. This look-up table is configured in the configuration registers (see section 4.1.3). The router byte can for example be used to address a memory-handling module, which collects frames from the Proximity-1 module and places them in memory. The OBC can then read the data from memory.

Since the output data is formatted as transfer frames, the OBC software is responsible for assembling data segmented over several frames into the original data packets. This design decision is discussed further in section 4.4.11.

4.1.3 Register Interface

The register interface between the Proximity-1 module and the OBC consist of a set of shared registers (all the registers are listed in section A.4). The registers can be grouped into four different types.

Configuration Registers

The local OBC uses these registers to configure the Proximity-1 module. The configuration registers include all MIB parameters (see section 2.2.9) and the transmitter and receiver configurations. Any new values written to these registers by the OBC are immediately used by the Proximity-1 module.

Directive Registers

The OBC uses directive registers to issue local directives to the MAC (for a list of local directives see section 2.2.9). When the OBC writes to one of the local directive registers, the MAC is notified. The directive registers contain the following local directives:

- ***Set Mode***. Available options are Inactive, ConnectingL and ConnectingT. The action taken upon reception of one of these directives are as described in section 2.2.9. Set Mode Active can not be chosen by the OBC. The Active mode is only tracked and handled internally in the MAC block. This directive is mainly used to initiate a session to a remote transceiver.

- ***Local Comm Change*** This directive starts a local Comm Change. The MAC uses the values in the transmitter and receiver configuration registers to form Set Transmitter Parameters and Set Receiver Parameters to be sent to the remote transceiver. One bit in the Local Comm Change register decides if PL Extension are to be sent in the Comm Change activity. Since the MAC uses the values in the transmitter and receiver configuration registers, it is both required and assumed that the OBC writes the values it wants to use to these registers before issuing the Comm Change directive.

- ***Local No More Data***. This directive tells the MAC that the local OBC has no more data to send. The MAC generates a Set Control Parameters directive which contains the Remote No More Data information. The directive is sent to the remote transceiver to notify it that no more data will be sent from the local transceiver.

The three directives in the bullet list above are the local directives that the MAC is notified by and supposed to take action upon when received from the OBC. In section 2.2.9, several other local directives are described which are not mentioned here. The MAC does not need to take any special action for those other directives, so they are implemented as values accessible via configuration register. A complete list of all the local directives used in the implementation can be found in section 4.7.1.

Status Registers

The status registers contains the value of different internal signals and registers within the Proximity-1 module. These registers can be read by the OBC in order to access the state of the Proximity-1 module and what operations it currently is performing. Examples of status registers are the MAC state, current transmitter and receiver configurations, and number of acknowledged Sequence Controlled frames.

Interrupt Registers

Interrupts are sent from the Proximity-1 module to the OBC to notify it about important events. Interrupts are for example generated when a persistent activity succeed or fail;

when a Sequence Controlled frame has been acknowledged; when an Expedited frame is transmitted; or if the carrier signal in the Physical layer is lost.

4.2 Functionality Overview

An overview of the system is shown in Figure 4.3. All blue (gray) arrows use a streaming data interface described in Section 4.2.1. All thin black arrows are serial data. The wide black arrow depict configuration registers that can be written to and read from the OBC. The system is split into two main parts, the transmit and receive stages. Furthermore, the MAC block controls the operation of the entire module, thus it does not belong into either stage exclusively and is therefore viewed as a separate part.

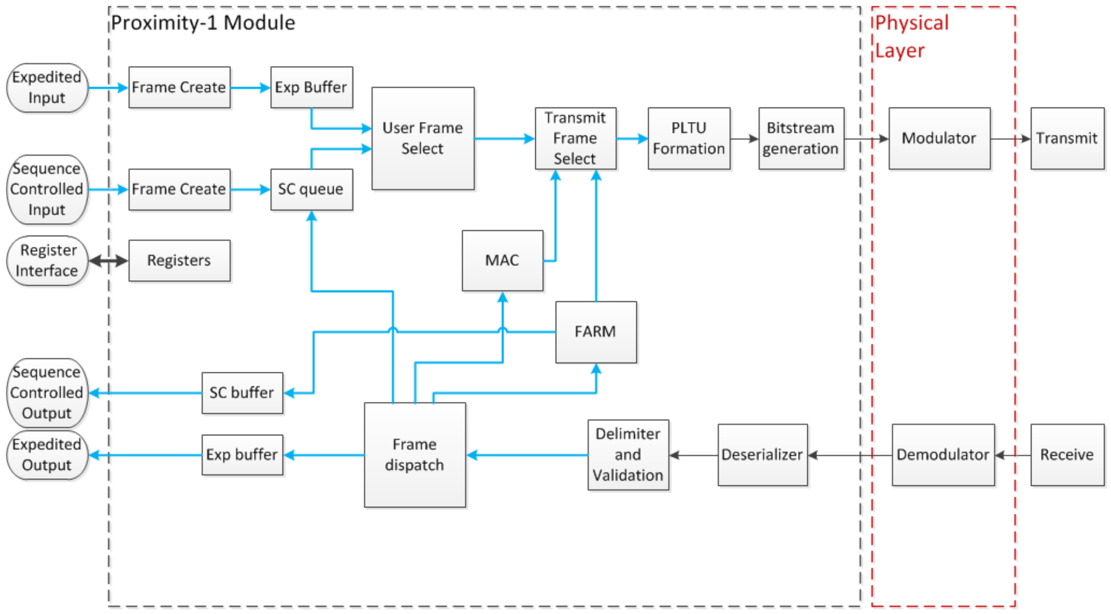

Figure 4.3: Block diagram of the highest level system blocks.

4.2.1 Streaming Data Interfaces

To ensure that the module remains flexible and expandable, it is important to use consistent interfaces between the blocks in the design. This is especially true for data paths that have high throughput requirements, since a well-defined interface yields a deterministic data flow.

One interface that allows such a data flow is the data streaming interface. In such an interface, the data consists of a stream of data words that is propagated through the data path. The size of the data word directly influences the possible throughput, a 32-bit data stream yields twice the throughput of a 16-bit one.

Since some blocks may need to process data over several clock cycles, it must be possible to throttle incoming and outgoing data, i.e. temporarily reduce the data rate.

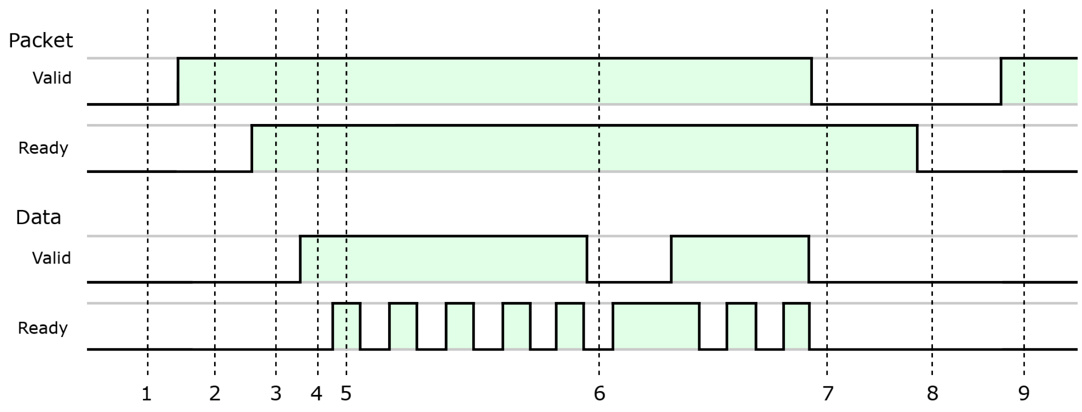

Figure 4.4: Example of a streaming handshake protocol where a number of data words are transferred between two blocks.

Other blocks, such as buffers, may be full and require stopping new data until they are flushed. These two types of flow control can be achieved by a simple handshake protocol, where the data must be provided by one block, and acknowledged by the other. One simple way to implement such a protocol is by using two boolean flag signals, or handshake flag pair: Data Valid and Data Ready. The data producer sets the Data Valid flag true when it is providing valid data. The data receiver sets the Data Ready flag true when it is ready to consume data. When both flags are true, the data is transferred between the blocks.

For a streaming data interface, the data packet is split into several data words that are sequentially sent. The data transferred under one handshake is one data word, and several handshakes are needed to transfer the entire data packet. It is thus useful to be able to distinguish between sets of data words that constitute a data packet. This can be achieved by a separate handshake flag pair: Packet Valid and Packet Ready. During the transfer of a data packet, the data producer sets the Packet Valid flag true, and keeps it true for the duration of the data packet transfer. The data consumer sets Packet Ready true to accept the transfer of the packet, and only sets it to false after the producer has signaled the packet end by setting Packet Valid to false. The producer may only start a new packet when the consumer Packet Ready is false. This allows both sides to throttle packets.

The streaming handshake interface provides great flexibility to the blocks with its throttling functionality, while remaining fairly simple to implement. In addition, the interface is in accordance with the handshaking interface guidelines described in section 3.2.2. An overview of the signal flags is shown in Table 4.1.

Using two handshake flag pairs, the data producer or the data consumer can throttle both data words and data packets. An example of this is shown in Figure 4.4. The following events occur in the example figure:

1. Idle, no data is transferred between the blocks.

Table 4.1: Boolean flags used by the data stream handshake interface.

Signal	**Driven by**	**When false**	**When true**
Data Valid	Producer	Producer provides no data	Producer provides valid data
Data Ready	Consumer	Consumer cannot consume data	Consumer can consume data
Packet Valid	Producer	Producer provides no data packet	Producer provides data packet
Packet Ready	Consumer	Consumer is ready for data packet	Consumer is consuming data packet

2. Data provider signals Packet Valid, implying that a new data packet is available for transfer. The provider must remain in this state until the consumer signals Packet Ready, allowing the consumer to throttle the new packet.
3. Data consumer signals Packet Ready, implying that the data packet is ready to be consumed. The provider may now send data words.
4. Data provider signals Data Valid, implying that the data output is ready to be read by the consumer, the consumer may wait before signaling Data Ready, throttling the data.
5. Data consumer signals Data Ready. Because both Data Valid and Data Ready is true during this clock cycle, data is read by the consumer, and the producer updates the data output.
6. Data producer stops signaling Data Valid, temporarily throttling the data output. Since the Packet Valid signal remains true this is not an end of packet, instead the producer needs some cycles to continue providing data for the same packet.
7. Data producer stops signaling Data Valid and Packet Valid, implying that the entire packet has been transmitted. The data consumer still signals Packet Ready, implying that the last packet is still being processed and preventing a new packet from being sent.
8. Data consumer stops signaling Packet Ready, implying that it is ready to consume a new packet.
9. Data producer signals Packet Valid again, a new packet is available for transfer.

4.2.2 Data Buffers

A number of buffers are present in the design. These are mainly used to ensure that data is available for processing evenly throughout the system, thus spreading the workload. At certain locations, they also ensure that blocks requiring a constant stream of data are never starved. One example is the serializer block, which requires new data every serial clock cycle. If the data would not reach the serializer in time, the output bit stream would become corrupted. There are two ways to store data using a buffer: in a *shared memory* or a *dedicated memory*.

A dedicated memory handles write and read requests from one block only. This means that the read and write times are deterministic and fast, and are very well suited for buffers that feed data to blocks requiring constant data streams.

A shared memory may be accessed by multiple blocks and even the OBC or other peripheral modules. Dedicated memories are generally quite flexible in size, and are well suited for buffers that require larger memory spaces. They are especially useful for larger, more complex systems, since the system designer can decide what modules share what memory, reducing the total number of memory arbiters and adding system flexibility. Because the memory is shared between multiple users, the read and write times are non-deterministic in the module context.

All buffers, regardless if their memories are dedicated or shared, use the streaming data interface, and may thus stop new data frames if no slot is available via the Packet Ready signal. Furthermore, buffers implemented using shared memory also throttle incoming and outgoing data words because of their non-deterministic read and write times.

The system uses two types of buffering, single buffering and double buffering. A single buffer has one memory slot that holds one data word. This means that the buffered value must be read in its entirety before a new value can be written to it. Double buffering on the other hand provides two memory slots, one reading and one writing slot. A new value can thus be written while an old value is read. When the old value has been read, the slots are flipped, and the recently written value is available for reading. This improves workload distribution, since different sides of the buffer may operate on different data at the same time. The usage of data buffers in the design is discussed in more detail in Section 4.5.

4.2.3 Clocks

Two clocks are assumed to be provided to the module:

1. Bus clock - Clocks the module logic, assumed to be 32 MHz or higher.

2. Serial clock - Clocks the serializer and deserializer, 2048 kHz.

The serial clock can be clock divided into all possible data rates defined by the Proximity-1 standard. Since the transmit and receive links may have different data rates, two clock dividers provide a serial transmit clock and a serial receive clock internally.

4.3 Transmit Stage

Figure 4.5 shows the data path of transmitted data packets. The sublayer of the Proximity-1 standard that the blocks belong to is also indicated in the figure. The block division have been done to mimic the sublayer layout and functionality as far as possible. Each sublayer is implemented as one or more blocks in the design. This is intended to facilitate adding functionality to the design in the future, as it clarifies which block corresponds to what sublayer functionality.

To execute a packet transmission, a data packet is supplied to either the Sequence Controlled input or Expedited input. The data packet is read by the Frame Create block and split into frames. Each frame is then propagated through the transmission chain, until it is finally serialized and radiated by an antenna.

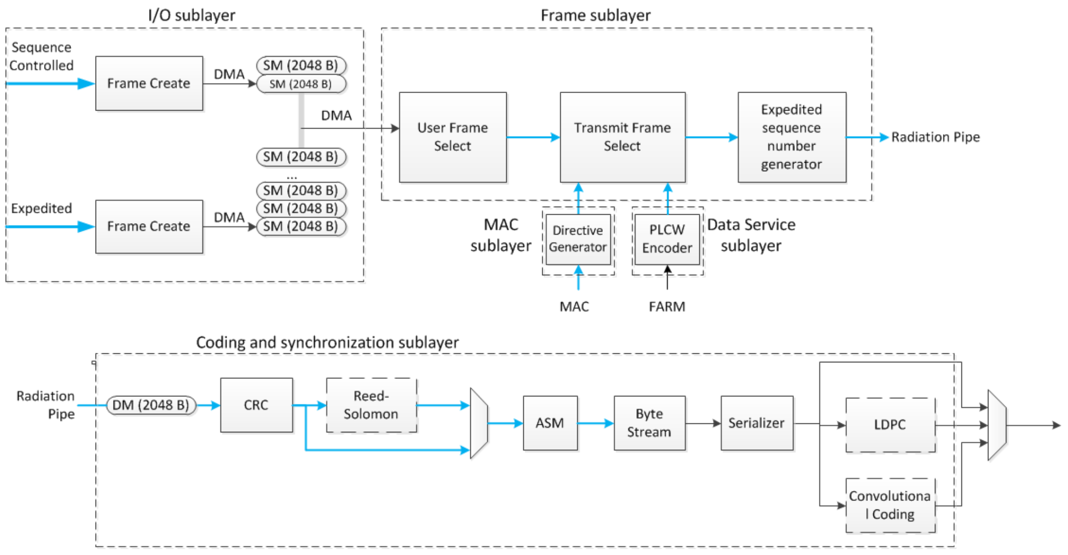

Figure 4.5: Block diagram of the transmit stage. Dashed blocks are optional and left out in the prototype system.

4.3.1 Frame Creator

The Frame Creator block receives data packets from an outside block. The packet data is provided over a streaming interface described in section 4.2.1. The data packets are put into transfer frames. The OBC can commit three categories of packets for transmission. For all categories of packets, the data is converted into one or several transfer frames. This is done by prepending the data with a transfer frame header. All resulting transfer frames are stored in a frame buffer in memory. The three available packet categories are:

1. Single-data packet
2. Multiple-data packet

3. Protocol-data packet

Single-data packets denote larger chunks of data that can fill one or several transfer frames. When packets of this category is read by the Frame Creator block, the data is automatically segmented into multiple frames if it cannot fit into a single transfer frame. The resulting frames are buffered by one of the input frame buffer blocks.

Multiple-data packets denote a number of smaller data packets that all can be contained in one transfer frame. It is up to the OBC to guarantee that this is the case when it submits data packets of this category. Since this packet category always results in a single transfer frame, no special segmentation handling is required.

Protocol-data packets contains directives which are transmitted to the remote transceiver. These are guaranteed to fit in a single transfer frame, so no special segmentation handling is required.

Two Frame Creator blocks are instantiated in the design, one for Sequence Controlled data, and one for Expedited data. By sending a packet to either input, the OBC controls whether a particular message is sent using Sequence Controlled, i.e. reliable, or Expedited, i.e. unreliable QoS. Two inputs are used as it should be possible to send Expedited data packets even if Sequence Controlled data packets are stalled. Sequence Controlled data packets could for example be stalled when the Proximity-1 module has sent a couple of Sequence Controlled frames and is waiting for a PLCW acknowledgment before it sends any new Sequence Controlled frames.

4.3.2 Input Frame Buffers

Two blocks buffer input frames coming from either Frame Create block, one for each QoS. The size of the Sequence Controlled Frame Buffer depends on the Window Size MIB parameter, which corresponds to the number of frames that must be stored in the buffer. The theoretical maximum of this parameter is 128 (although a lower maximum value would likely be set for most missions), requiring 128 frames to fit in this buffer at one time. The Expedited Frame Buffer has a constant size independent of MIB-parameters. To allow a potentially large memory size but still provide an identical interface between the Frame Create and buffer blocks, both buffers use shared memory accesses. This is done by adding an interface to an external bus (a DMA channel) to each frame buffer block, through which a shared memory can be accessed.

Expedited Frame Buffer

Any Expedited frames that are ready for transmission are stored in the Expedited Frame Buffer block. To allow simultaneous reading and writing from the buffer, it is implemented as a double buffer. This is intended to decrease the impact of message congestion when many Expedited messages are committed for transmission at the same time.

Upon reading a frame from the Expedited Frame Buffer, that frame is also flushed from the buffer, allowing a new frame to be buffered. Whenever all frames of a data packet have been transmitted, the OBC is notified by the module, see Section 4.1.

Sequence Controlled Frame Buffer

Since Sequence Controlled frames may need to be retransmitted, they cannot be flushed once they have been transmitted. Instead, they must remain available until they are acknowledged by the remote transceiver in accordance with the ARQ-algorithm. This FOP process (Section 2.2.2), is implemented in the Sequence Controlled Frame Buffer block.

Whenever Sequence Controlled frames are transmitted, rather than being flushed from the buffer, they are marked as sent and stored in the retransmission queue. The retransmission queue contains all transmitted but unacknowledged Sequence Controlled frames. Whenever an acknowledgment frame, PLCW, is received from the remote transceiver, all acknowledged frames are flushed from the buffer. If the PLCW requests frame retransmission, any frames left in the retransmission queue are retransmitted before any new frames are accepted into the buffer.

Whenever all frames of a data packet have been acknowledged, the OBC is notified, see Section 4.1.

4.3.3 User Frame Select

The User Frame Select block selects whether to transmit frames from the Expedited or Sequence Controlled frame buffer. When available, Expedited frames are always prioritized before Sequence Controlled ones, following the QoS priority rules defined by the Proximity-1 standard [4].

4.3.4 Transmit Frame Select

The Transmit Frame Select block selects either a frame from the User Frame Select block or a protocol frame from the MAC or FARM blocks. The frames are selected based on a priority list:

1. MAC Frame (protocol data)
2. FARM Frame (PLCW) if the latest transmitted frame was a frame from the User Frame Select block
3. Expedited frame from the Frame Select Block
4. Sequence Controlled frame from the Frame Select Block
5. FARM Frame (PLCW)

4.3.5 Expedited Sequence Number Generator

Since Expedited frames can be generated by the MAC, FARM and Expedited Frame Creator blocks, the sequence number of these frames must be counted and generated after the Transmit Frame Select block. This is done by the Expedited Sequence Number

Generator block, which is placed after the Transmit Frame Select block (see Figure 4.5). When propagating Expedited frames, the Expedited Sequence Number Generator block increments the Expedited sequence number and writes the result to the frame header.

4.3.6 CRC

The CRC block calculates the CRC-32 of the incoming transfer frame and appends it to the frame.

4.3.7 ASM

The ASM block prepends a static ASM to the transfer frame.

4.3.8 Byte Stream

The Byte Stream block serializes frames to bytes. When no frame is available for transmission, the Byte Stream block outputs a static idle stream, see Section 2.2.4. Whenever a frame is available, this block requires a constant data stream to avoid corruption of the output byte stream. The output byte stream is clocked by the serial transmit clock divided by 8, providing a data rate of one bit per serial clock cycle to the serializer.

4.3.9 Serializer

The Serializer block serializes the byte stream to a bit stream. This block requires a constant byte stream input to avoid corruption of the output bit stream, however this constraint is already met by the output of the Byte Stream block. The output bit stream is clocked by the serial transmit clock. This is the bit stream that is to be modulated and radiated by the Physical layer.

4.3.10 PLCW Encoder

The PLCW encoder forms PLCWs based on the state of the FARM process in the FARM block. The following FARM state variables are contained in a PCLW frame:

- Expected Sequence Controlled sequence number
- Retransmission of Sequence Controlled frames required

Whenever the content of the PLCW changes, the new PLCW must be retransmitted. This means that if any of the FARM state variables above change, the PLCW encoder block will signal that a new PLCW frame is available for transmission to the Transmit Frame Select block. Since the ARQ-algorithm only requires that the most recent PCLW is transmitted, the actual PCLW data is not generated until the Transmit Frame Select block commits the PLCW frame for transmission. This ensures that PLCWs are always generated based on the most up-to-date FARM state.

In the Proximity-1 standard recommendation, PLCWs are generated as soon as a Sequence Controlled frame is validated and accepted by the FARM process. If many small, valid Sequence Controlled frames are received in a communication session, a considerable amount of PLCWs are generated and transmitted. This may take up a significant portion of the transmission time, reducing the overall link efficiency. Therefore, other strategies can be employed that reduce the impact of PLCWs on the data throughput, such as acknowledging several Sequence Controlled frames in one PLCW. Acknowledging multiple frames through PLCWs is already supported by the standard, though the PLCW transmitter must be configured to transmit PLCWs less often. In our design, a configuration register allows the OBC to select two options of when PLCWs should be generated:

- Generate a PLCW as soon as an in-order Sequence Controlled frame arrives. This follows the recommended Proximity-1 standard.
- Generate a PLCW when $(Transmission\_Window\_Size - 1)$ in-order Sequence Controlled frames have arrived. In this setup, a PLCW is generated only after a number of frames equal to the current window size minus one has been received. If an out-of-order Sequence Controlled frame is received, the PLCW is generated immediately.

The second option is especially feasible when many small Sequence Controlled frames are sent, since this would otherwise generate a significant number of PLCWs that may be close to the size of the actual data frames. Both options are evaluated in Section 5.3.

Furthermore, PLCW transmission may also be triggered by a PLCW retransmission timer. This timer exists to make sure that the peer transceiver always receives a PLCW at some point, even if one PLCW is lost due to packet losses in the radio link. The timer resets whenever a PLCW is transmitted, and its retransmission period value can be configured by using the configuration registers of the module. When using an acknowledgment interval larger than one, the retransmission period may need to be configured accordingly. For example, if only two frames can be received during the retransmission period time due to a low data rate, each PLCW will acknowledge those two frames, regardless of the acknowledgment interval being higher.

4.3.11 Directive Generator

The Directive Generator block is responsible for generating remote directives to be sent to the remote Proximity-1 transceiver. The MAC block signals to the Directive Generator whenever a directive is to be sent and what type of directive that is to be generated. The available remote directives are listed in section 2.2.9.

4.4 Receive Stage

Figure 4.6 shows the data path and related blocks on the receiving side of the Proximity-1 protocol implementation. The sublayer of the Proximity-1 standard that the blocks belong to is also indicated in the figure. An exception to this is the Delimiter and Validation block. In this block, the frame validation is included, which is part of the Frame sublayer functionality (see section 2.2.3). The reason for having the frame validation in this early stage is for the simplicity of having all the frame verification in the same place, i.e. both the frame validation and CRC check.

The received bitstream is collected in a deserializer block which converts the bit stream to bytes. The received transfer frame is then validated using information in the transfer frame header and a CRC check is done. If the frame passes the validation and CRC check, the frame is placed in a double buffer. The Frame Dispatch block takes frames from the double buffer and dispatch them depending on their type to internal blocks in the Proximity-1 module or to the OBC through double buffers.

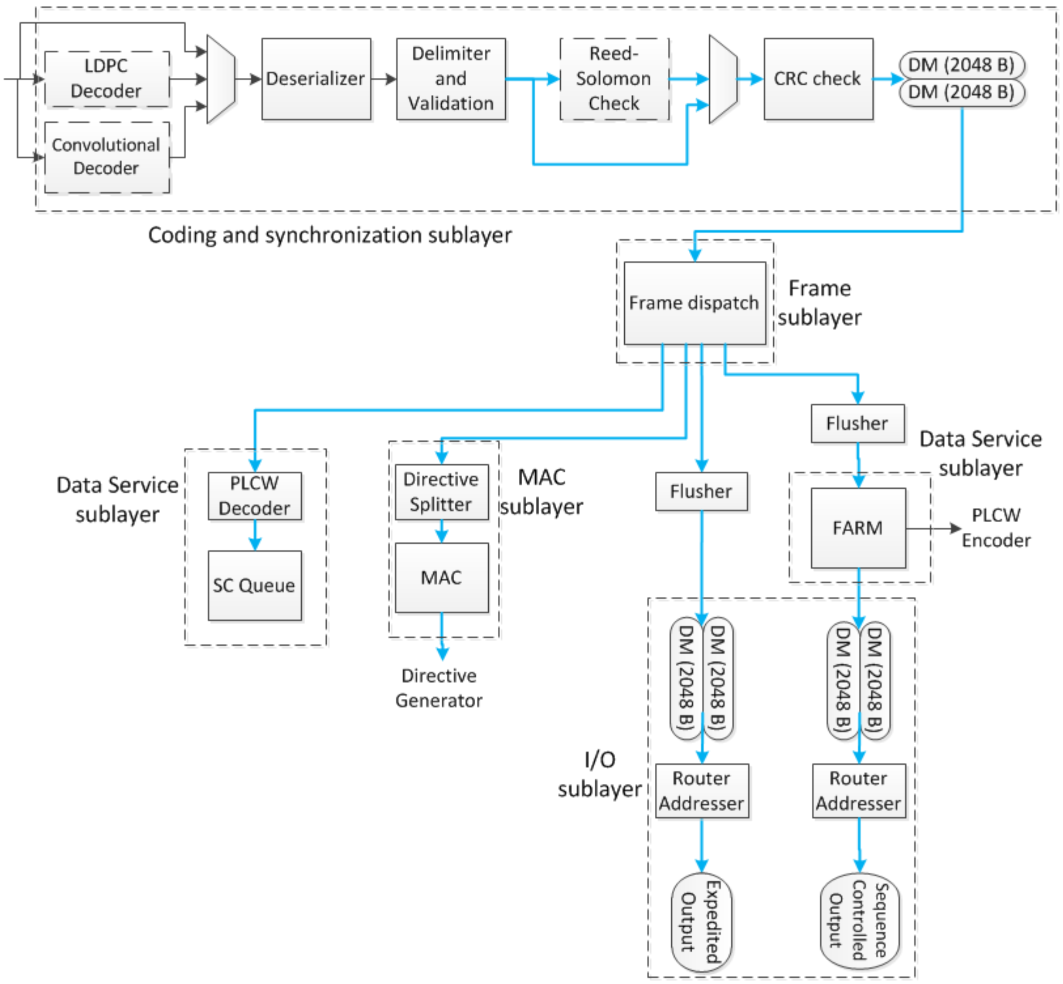

Figure 4.6: Block diagram of the receive stage. Dashed blocks are optional and left out in the prototype system.

4.4.1 Deserializer

The deserializer block receives a bitstream at a constant rate from the underlying Physical layer. The input bits are pipelined to a depth of 3 bytes, and when the pipeline contains an ASM, the Deserializer aligns its byte output stream to the bit stream and signals the start of a new frame. The deserializer remains aligned to the bit stream until the Delimiter and Verification block signals the end of a frame, after which the deserializer starts searching for an ASM again.

In the Proximity-1 standard documents it is described how different implementations of the modulation in the Physical layer can result in that the data received is the inverse of the data sent [2]. Therefore, it is recommended that an implementation should search for the inverse ASM, and if it is found, the rest of the data received should be inverted to obtain the original data. Our deserializer follows this recommendation, and hence the deserializer searches for the inverse of the ASM as well as the normal ASM. If an inverse ASM is found, the data following it is inverted.

4.4.2 Delimiter and Verification

When the Deserializer signals the start of a new frame, the Delimiter and Verification block checks the incoming byte stream. In order to verify that the received frame is of the correct type and the local vehicle is the intended receiver of the frame, the frame header fields *transfer frame version number*, *source-or-destination identifier* and *spacecraft identifier* are verified. The block then reads the frame length and, if all checks passed, outputs all frame bytes including the frame header and CRC until the specified frame length is reached. When the end of the frame is reached, the Delimiter and Verification block signals the Deserializer that the end of frame has been reached.

If any of the verification checks fail, the delimiter does not immediately signal end of frame, but rather assumes that the read frame-length value is correct. We assume that if the checks fail, it is not likely to be caused by noise being misinterpreted as an ASM and thus a frame, instead it is more likely because a frame intended for another transceiver was received. If we were to immediately signal end of frame and thus start listening for new ASMs, there is a risk that data could be interpreted as an ASM. Since data is much more likely to be identified as an ASM compared to noise (the data would need to contain the ASM sequence, but the sequence may be unaligned with respect to the data bytes which significantly increases the risk), we would rather wait until the supposed frame is done before we start listening for new ASMs.

4.4.3 CRC Check

The CRC check block calculates the CRC-32 of the received frame and validates that the frame does not have any bit errors. If the frame passes the check, it is stored in a double buffer. Otherwise it is discarded.

4.4.4 Frame Dispatch

The Frame Dispatch block dispatches frames depending on their type:

- A User Data Frame (U-frame) using the Expedited QoS is dispatched to the Expedited output double buffer.
- A Sequence Controlled U-frame is dispatched to the FARM block in order to validate that the frames arrive in order. If the frame is accepted by the FARM block, it is dispatched to the output Sequence Controlled double buffer, otherwise the frame is discarded.
- A Protocol Data Frame (P-frame) that contains a PLCW is dispatched to the Sequence Controlled frame buffer block which handles the FOP process.
- A P-frame that does not contain a PLCW is dispatched to the MAC block.

4.4.5 PLCW Decoder

The PLCW decoder receives a PLCW from the frame dispatch block, extracts the contained sequence number and retransmission flag and sends this to the FOP process in the Sequence Controlled frame buffer block.

4.4.6 Directive Splitter

The Directive Splitter block receives the frame data of received protocol data frames (i.e frames containing directives sent by the remote transceiver), and splits them into 2-byte directives. These directives are then sent to the MAC. When a directive has been acknowledged, the next 2-byte directive is sent and so on, until the entire protocol data frame has been read. This block is needed since a frame can contain more than one directive, and to make the implementation of the MAC easier the MAC only handles one directive at a time.

4.4.7 FARM

The FARM block runs the FARM process. It checks the sequence number of the received frame to see if it is equal to the expected sequence number. If this is true, the frame is dispatched to the output Sequence Controlled double buffer and the PLCW encoder is told to generate a PLCW to acknowledge the received frame. If the frame sequence number of the received frame is higher than expected, the PLCW encoder is requested to generate a PLCW which contains the expected sequence number. If a frame with a sequence number that has already been acknowledged arrives, the frame is discarded.

4.4.8 Output Double Buffers

Two output double buffers are present in the design, one for Sequence Controlled frames and one for Expedited frames. The OBC is responsible for ensuring that frames are read from the buffers. However, the OBC does not have to handle the reading of frames itself, but can instead command another module of the rover system to take frames from the Proximity-1 module and for example place them in memory, as described in section 4.1.2.

4.4.9 Flusher

The flusher blocks check if the output double buffers are full, and if so, frames that arrive are discarded. This is done in order to never stall the receive link logic. Note that the flusher managing Sequence Controlled frames is placed before the FARM block. This discards frames that cannot be stored before they are seen by the FARM process. This means that the frames will not be acknowledged and hence are resent by the remote transceiver.

4.4.10 Router Addresser

This block prepends a router byte to outgoing frames. The byte is selected based on the port id specified in the frame header. 8 different port IDs (0 to 7) may be specified by the transmitter, and these are mapped to one of eight bytes that can be specified via the configuration registers.

4.4.11 Arrange Frames Into the Original Data Packets

As described in section 2.2.1, the Proximity-1 protocol is responsible for arranging the received user frames back into the original user data packets before they are sent to the OBC. If large data packets were to be assembled in our hardware module, a considerable amount of memory would be required to buffer the frames constituting the packet.

Since the software of the OBC can be more flexible regarding the allocation of memory, the task of assembling frames into data packets is preferably placed in software. Since the task is a minor one of the Proximity-1 protocol functionality, this software functionality is not implemented as a part of the prototype in this project. It should also be mentioned that this is the only part of the Proximity-1 Data Link layer that we have chosen to place in software.

4.5 Data flow

As mentioned in Section 4.2.2, data buffers are required to ensure constant data streams for blocks that require them. On the transmit side, the Byte Stream and Serializer blocks require constant data streams. Since the shared memories of the Input Frame Buffer blocks have non-deterministic read times, we need an additional dedicated memory buffer

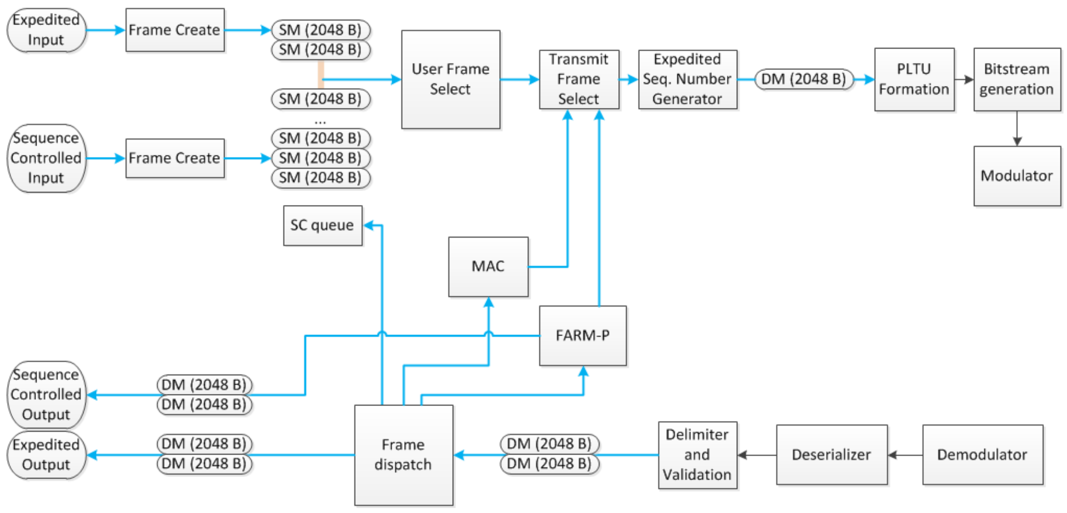

Figure 4.7: The main data flow of data and frames in the module.

to guarantee a constant data stream. In the prototype, this buffer is placed between the Expedited Sequence Number Generator block and the ASM block. This is done under the assumption that the blocks between the buffer and the blocks requiring constant data streams have deterministic data propagation time, that is that data is guaranteed to always reach the constrained blocks on time. The buffer could also be placed after the ASM and CRC blocks, but this would require larger buffers since the ASM and CRC bytes would also need to be stored. Note that while the difference in size would be just a few bytes, the maximum data size would no longer be the 2048 bytes that may be neatly aligned in memory.

Another important thing to note is that contrary to most other buffers in the module, this buffer is implemented as a single buffer rather than a double buffer. A single buffer has a shorter propagation time than a double buffer, meaning that more up-to-date frames will be sent. While the propagation time hardly matters for data frames, we consider it important to ensure that PLCWs are as up to date as possible.

The only drawback of using a single buffer to feed the serializer is that gaps may occur in the data stream between frames. This happens since a buffer write cannot happen in parallel with a buffer load. Recall that the serializer is fed with one byte at a time (Section 4.3.8). If the buffer write cannot complete before the serialization of the last byte of the previous frame, the serializer will insert idle bytes between the frames. We consider this to be a very minor concern since it follows the standard, and thus use a single buffer in the prototype.

A double buffer is placed before each data output. This is done since the reading rate of the OBC (or another hardware module) is non-deterministic, and the Deserializer and Delimiter blocks require a constant output data stream. These double buffers could be implemented as queues instead, which would reduce the risk of having to flush valid received frames in the flusher blocks. This could yield a more efficient radio link, but we assume that the OBC reads data from the buffer outputs fast enough so that the double buffer can always handle new received frames, and thus never stops the input data.

4.6 Timers

Four different timers are used in the design to keep track of the timing of events. The four timers in the design correspond to the timers described in the Proximity-1 standard [4], and they are described in the subsections below.

4.6.1 PLCW Timer

This timer is loaded with a value contained in a MIB parameter called “PLCW Repeat Interval”. The timer is used to regularly send PLCWs. When it times out, a PLCW is sent and the timer is restarted.

4.6.2 Carrier Loss Timer

This timer is loaded with the MIB Parameter value called “Carrier Loss Timer Duration”. If the carrier is lost during a communication session, the Carrier Loss Timer is started and counts downwards. If the carrier is acquired again before the timer times out, the timer is reset to its original value and stopped. This is done in order to give room for short interruptions in the session. However, if the timer times out, the MAC is notified that the carrier is lost, and the MAC should take appropriate action (the actions taken by the MAC block is described in the MAC block section 4.7.6).

4.6.3 Wait Timer

This timer is used by the MAC block to time different activities. The MAC loads a value into the timer and starts it, and if the timer times out the MAC is notified and takes appropriate action. For example, during session establishment, the MAC loads the MIB Parameter value ”Carrier Only Duration” into the timer. When the timer times out, the MAC is notified and knows that the time that only the carrier should be radiated has ended, and it can move on to the next state.

4.6.4 Sync Timer

This timer is loaded with the value in the MIB Parameter ”Sync Timeout”. If an invalid PLCW is received, the Sync Timer is started. If it times out, an interrupt tells the OBC that the ARQ algorithm process in the sender and receiver does not agree and needs resynchronization. The MAC is also notified via a Resync signal which is set to true when the timer times out. When the Resync signal is set to true, a resynchronization of the FOP and FARM is done.

4.7 MAC Sublayer Block

The MAC block implements the MAC sublayer of the Proximity-1 Data Link layer. As described in section 2.2.8, the MAC controls the operation of the Proximity-1 protocol and handles directives (commands) from the local OBC and remote Proximity-1 transceiver. The MAC block takes as input status signals from the other Data link sublayers, status signals from the Physical layer and commands received from the local OBC and remote Proximity-1 transceiver. Using these inputs, the MAC monitors the operation of the Proximity-1 protocol and controls the other Data link sublayers through control signals and configuration registers.

4.7.1 Local Directive Implementation

The different local directives described by the Proximity-1 standard can be seen in section 2.2.9. Below is a list of the different directives and how they have been implemented in our design.

- ***Set Mode***. Implemented through registers accessed by the OBC and MAC block, see section 4.1.3
- ***Set Initialize Mode***. This directive is in our implementation equivalent to issuing a Set Mode Directive with the value Inactive (see section 4.1.3 and 2.2.9).
- ***Load Communication Value Buffer***. This directive should load the remote transceiver configurations used in a Hail or Comm Change activity according to the Proximity-1 standard [4]. However, since these values are implemented as configuration registers in our implementation (see section 4.1.3), the OBC is supposed to have written the related registers before issuing a Hail or Comm Change. When the MAC performs a Hail or Comm Change, it simply uses the values in the configuration registers and hence assumes that the OBC has written the correct values there prior to the Hail or Comm Change.
- ***Local Comm Change***. Implemented through registers accessed by the OBC and MAC block, see section 4.1.3
- ***Local No More Data***. Implemented through registers accessed by the OBC and MAC block, see section 4.1.3
- ***Set Duplex***. Since our implementation only supports full duplex operation, this local directive is not implemented. However, a directive register has been created in the design which is supposed to be used for the Set Duplex directive. This has been done to make it possible to add the half-duplex and simplex operation modes in future expansions of the module. In the current implementation, the Set Duplex register is not used.

- ***Set Receiving SCID buffer***. Similar to the Load Communication Value Buffer directive, this directive is implemented as a configuration register. The Proximity-1 module assumes that the correct value is written to the register before its value is used (for example, the register can be written before a communication session is started).
- ***Read Status***. The values of states, buffers and current configurations of the Proximity-1 module that can be of interest to the local OBC are placed in status registers in our design (see section 4.1.3). These status registers are updated as soon as the related internal Proximity-1 value changes, and hence the OBC can get access to the value anytime by reading the related register.

4.7.2 Interface to Other Proximity-1 Blocks

The MAC communicates with several other blocks in the Proximity-1 module. In general, blocks either receives commands/data from the MAC or outputs commands/data to the MAC. The interfaces between the MAC and the blocks are all implemented in accordance with the handshaking rules described in 3.2.2. The following blocks delivers commands to the MAC:

- ***OBC directive registers***. Delivers local directives from the OBC to the MAC.
- ***Directive Splitter***. Forwards directives to the MAC received from the remote transceiver.

The following blocks handles commands issued by the MAC:

- ***PLCW Encoder***. The MAC can force the PLCW Encoder to send a PLCW. This is for example done during the Hailing Activity. When the responder of the hail has received the directives sent by the caller of the hail (Set Transmitter Parameters directive and Set Receiver Parameters directive), it is supposed to send a response to the caller. This response is simply a PLCW, and hence the MAC commands the PLCW Encoder to send a PLCW.
- ***Directive Generator***. When the MAC wants to send a directive to the remote Proximity-1 transceiver, it commands the Directive Generator to form and send the directives.
- ***FARM***. During resynchronization of the FOP and FARM processes, the MAC can set the Sequence Controlled frame sequence number that the FARM expects to receive next. This is done in order to resynchronize the FARM to a known sequence number when the sender and the receiver of Sequence Controlled frames disagree on the next sequence number.
- ***Time Tagging Block***. The MAC can tell the Time Tagging Block to start to store send and receive time of frames. The functionality of this block is not implemented

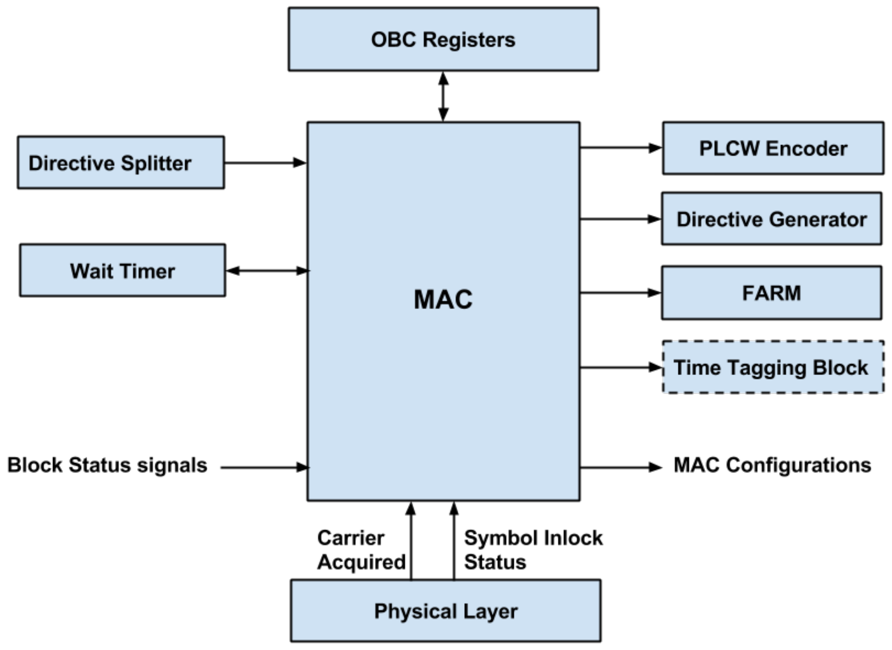

Figure 4.8: Interfaces between the MAC and other blocks

since the timing services is outside the scope of this project. However the interface to the block is implemented in order to make it easier to add it in future.

In addition to sending and receiving commands to other blocks, the MAC receives status signals from several blocks. The status signals includes for example a signal which indicates if the single output buffer is empty, if a frame has been received and if there are frames waiting to sent. The MAC also receives two status signals from the psychical layer, ”Carrier Acquired” and ”Symbol Inlock Status”. The Carrier Acquired signal is true when a carrier is detected on the transceiver. The Symbol Inlock Status is true when a serial stream of data has been received and is delivered to the Deserilizer block.

The MAC contains a number of internal status registers which for example indicate if the Proximity-1 module is transmitting data, which mode the module is in and the current receiver and transmitter configurations. These status registers are needed by other blocks in the design and the Physical layer in order to operate correctly. Hence these status registers are driven by the MAC. These registers are referred to as MAC Configurations.

In Figure 4.8, an overview of the MAC interfaces can be seen.

4.7.3 MAC State Machine

The state diagram in Figure 4.9 shows an overview of the major states of the MAC block. The Hail Sequence state and Comm Change state contain several substates which are implemented as separate states of the MAC, but in the overview state diagram they are grouped into one state for simplicity.

The state machine of the MAC block is similar to the one specified in the Proximity-1 standard [4]. However, a Resync state and Session Termination state (not shown in the overview state diagram) have been added to fit our VHDL implementation . In addition, several “Waiting States” (not shown in the Overview state diagram) have been added in order to ensure that the blocks the MAC communicates with acknowledge different commands sent by the MAC before moving on to the next MAC state. For example, when the MAC receives the hail directives in the ”Waiting for Hail” state, it should move on to the ”Radiate Carrier Only” state according to the Proximity-1 standard [4]. However, the MAC should also send a PLCW as a response to the hail and start the Wait Timer. In order to do this with the handshaking interfaces in our design, the MAC goes from the ”Waiting For Hail” state to special ”Waiting States” where it waits for the PLCW Encoder and Wait Timer to acknowledge the commands. After the PLCW Encoder and Wait Timer acknowledgments have been received, the MAC can go to the Radiate Carrier Only state.

4.7.4 Initialize the MAC

The OBC places the MAC in the starting, Idle state by writing to the Set Mode directive register and giving the value “Inactive”. In addition to initializing its own internal variables in the Idle state, the MAC also initializes other Proximity-1 blocks by sending out a reset signal that goes to all blocks in the design. After the MAC has been placed in the Idle state, and before a communication session can start, the OBC is responsible for writing suitable values to configuration registers, both the MIB parameter values and the transmitter and receiver configurations.

4.7.5 Communication Session Establishment

After suitable configuration values has been written by the OBC, the Proximity-1 module is ready for a communication session. A communication session is started when the OBC writes either the value ConnectingL or ConnectingT to the Set Mode directive register:

- ***ConnectingL***. This mode puts the Proximity-1 module in a listening state, where the MAC blocks waits for a hail directive to be received from a remote transceiver. The Hail directive consist of the directives Set Transmitter Parameters and Set Receiver Parameters (optionally, Set PL Extensions directive can be included as well). When the hail is received, the MAC sets the registers holding the current receiver and transmitter configurations to the values received in the hail and sends a hail response (PLCW) on the new transmitter configurations. The choice of the

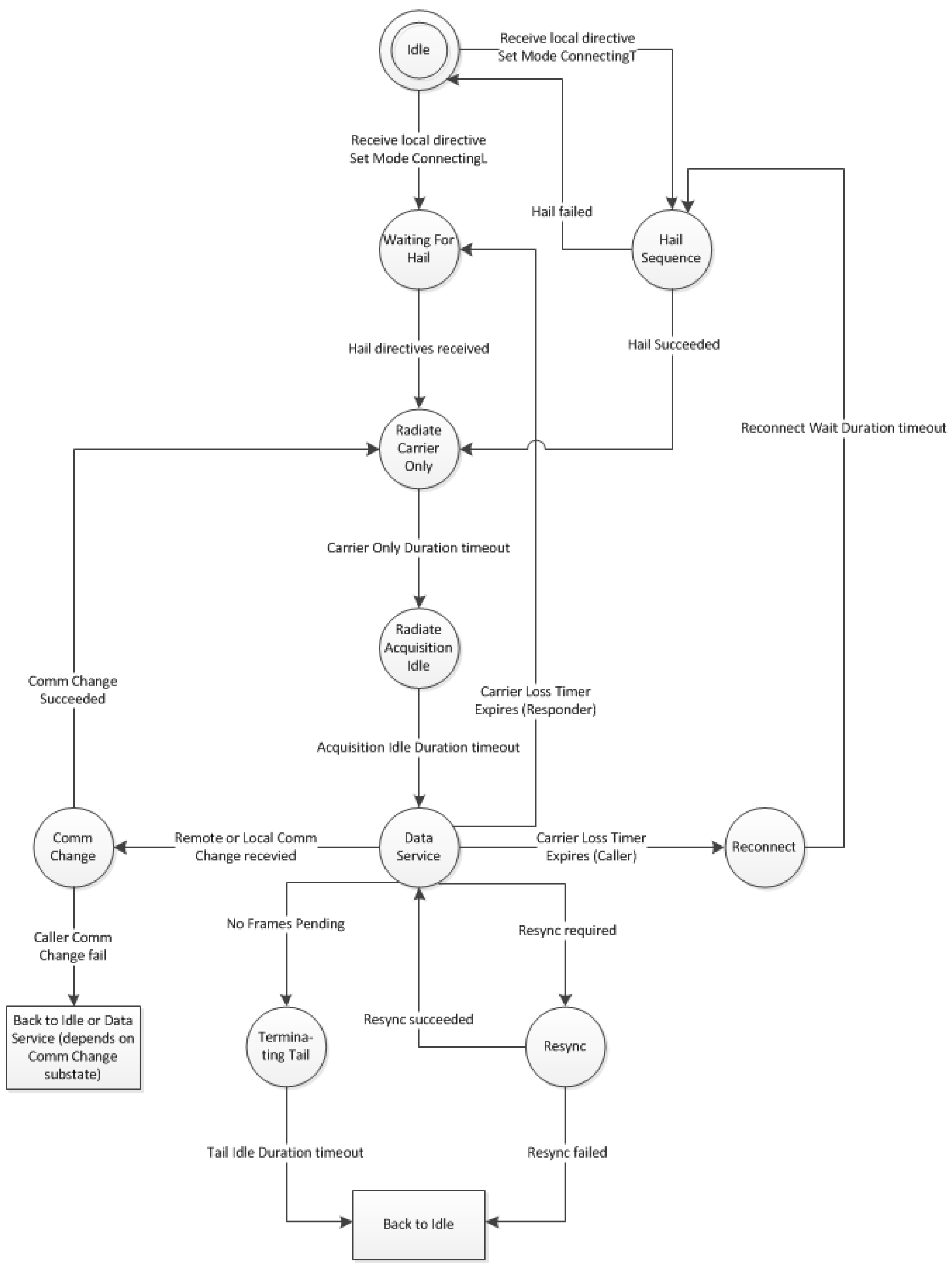
Idle
Receive local directive Set Mode ConnectingT
Receive local directive Set Mode ConnectingL
Hail failed
Waiting For Hail
Hail Sequence
Hail directives received
Hail Succeeded
Reconnect Wait Duration timeout
Radiate Carrier Only
Carrier Only Duration timeout
Radiate Acquisition Idle
Comm Change Succeeded
Carrier Loss Timer Expires (Responder)
Acquisition Idle Duration timeout
Comm Change
Remote or Local Comm Change recevied
Data Service
Carrier Loss Timer Expires (Caller)
Reconnect
Caller Comm Change fail
No Frames Pending
Resync required
Back to Idle or Data Service (depends on Comm Change substate)
Termina-ting Tail
Resync succeeded
Resync
Tail Idle Duration timeout
Resync failed
Back to Idle

Figure 4.9: MAC Overview State Diagram

ConnectingL mode means that the Proximity-1 module will act as the *Responder* of the session.

- ***ConnectingT***. This mode puts the Proximity-1 module in a transmitting state. The Hailing sequence is initialized in the MAC, which is a chain of MAC state transitions and actions which are made in order to send hailing directives to a remote transceiver and await a response. The choice of this mode means that the Proximity-1 module will act as the *Caller* of the session. The details of the hailing sequence is given in appendix A.1.

When hail directives have been received while in mode ConnectingL or a hail response have been received while in mode ConnectingT, the Wait Timer is loaded with the MIB Parameter register value “Carrier Only Duration” and the MAC transition to the Radiate Carrier Only state. In this state, the MAC drives signals which informs the Physical layer that a carrier signal should be transmitted but no data is modulated onto the carrier. When the Wait Timer times out, the MAC transitions to the Radiate Acquisition Idle state, in which modulation is on and idle data is radiated for the time given in the MIB Parameter register “Acquisition Idle Duration”. The operations performed in the Radiate Carrier Only and Acquisition Idle state are needed to set up a physical communication link between the two transceivers. After the Acquisition Idle state, the MAC moves into the Data Service state, where transfer frames are sent and received.

4.7.6 Actions and Events During an Active Communication Session

In the Data Service state, the Proximity-1 module is up and running and transfer frames are sent and received across the Proximity-1 Link. While in the Data Service State, several events can make the MAC transition to another state. The events are listed below:

- ***All frames has been sent***. When all frames has been sent and both the local and remote OBC have issued and sent the No More Data directive, the MAC prepares to terminate the session by transition to the state ”Terminating Tail”. The Proximity-1 module radiates a data idle sequence for a time defined in the MIB Parameter register ”Tail Idle Duration” and then the MAC goes to the Idle state.

- ***Local or Remote Comm Change directive received***. The local OBC issues a Comm Change request or a remote Comm Change request is received from the remote transceiver. The MAC transitions to the Comm Change state, which contains several substates in which the transmitter and receiver parameters are configured in both transceivers. The Comm Change sub-states and actions are described in more detail in the sections A.2 and A.3.

- ***Resync***. When the Synch Timer (see section 4.6) times out, it triggers a Resync signal to become true. This signal tells the MAC that the sender and receiver do not agree on the frame sequence numbers, and a resynchronisation procedure is needed. The resync events are described further in 4.7.7.

- ***Carrier Lost***. If the Carrier Loss Timer (see section 4.6) times out, the MAC is notified that the carrier is lost. Depending on if the local Proximity-1 module is the Caller or Responder of the session (see section 4.7.5) the MAC moves to different states. If the local Proximity-1 module is the Caller of the session, the MAC moves to the Reconnect state (and after this, to the Hail Sequence state). If the module is the Responder, the MAC moves to the Waiting For Hail state. This is done in order to try to set up the connection again through hailing.

4.7.7 Resync

When the Resync signal is true, the MAC knows that a resynchronization of FOP and FARM is needed. A MIB Parameter decides if the MAC should handle the resynchronization or if the OBC should handle it. If the resynchronization is not handled by the MAC, it stays in the Data Service state. Otherwise, the MAC generates a "Set VR" directive to be sent to the remote transceiver. The Set VR directive contains the Sequence Controlled frame sequence number that the remote transceiver sequence number should be set to. This number corresponds to the last sequence number that both transceivers agreed on (i.e the latest acknowledged sequence number). This will make the local and remote transceiver agree on the sequence number. When the directive has been created, the MAC moves to the Resync state where it waits for the Resync signal to become false (which means that the Resync has succeeded). If this does not happen within a predetermined time set by the MIB Parameter register "Resync Waiting Period", a new attempt to resynchronize is done by sending the Set VR directive again. If the resynchronization does not succeed in a certain number of attempts (set by the MIB Parameter register "Resync Lifetime"), the MAC transitions to Idle and the Proximity-1 module is reset, i.e the communication session is terminated. It should be noted that it is not clear in the Proximity-1 standard about what should be done if the resynchronization fails. The decision to reset the Proximity-1 module by going to the MAC Idle state might not be the optimal solution, since Expedited frames should still be able to be sent and received correctly. Another option could be that if the resynchronization fails, the MAC should notify the OBC and go back to the Data Service state.

4.7.8 Reset the MAC - Set Mode Inactive

In any state, the OBC can reset the MAC and Proximity-1 module by writing the value Inactive to the directive register Set Mode. This makes the MAC transition to the Session Termination state, in which the MAC acknowledges the OBC directive, sends an interrupt to the OBC indicating that the session terminated, and then moves to the Idle state where the MAC reset signal goes high and resets all blocks of the Proximity-1 module.

4.7.9 MAC and Timing Service Interfaces

The MAC can receive “take time sample” directives from the remote transceiver, but can not generate such directives itself since it is outside the scope of this project. There are directive registers available for time sampling usage, but code handling those directives are needed in the MAC in the Data Service state.

5

Results

This chapter contains a description of the functional testing of the Proximity-1 module. The performance of the module is also evaluated in terms of data transmission efficiency.

5.1 Test and Verification of the Prototype Functionality

In this section the testing and verification of the functionality of the Proximity-1 module is described. Several tests on different levels of the design have been done. All the major blocks of the design were tested separately. The whole Proximity-1 module was tested and the communication between two instantiations of the Proximity-1 module was tested. Finally, the Proximity-1 module was connected to a host rover computer and tested. This section will mainly focus on the tests that were carried out on higher levels of the design, i.e on the Proximity-1 module level and when the module was connected to the host rover system, since this gives a more complete view of the overall functionality of the design.

5.1.1 Module Functional Test

The functionality of each block was tested separately in Modelsim using VHDL test benches. Different input data was supplied and the block outputs and internal variables were verified.

After the functionality had been verified on the block level, the blocks were connected together to form the complete Proximity-1 module. Different data was input and different configurations were set and the behavior of the module and data outputs were verified in Modelsim.

In order to make it possible to simulate a communication session between two Proximity-1 modules, a VHDL test bench was created where two instantiations of the Proximity-1 module were used. The serial data output of one module was connected to the serial input of the other module, and vice versa. This means that the data sent from one Proximity-1 instantiation was sent to the other Proximity-1 instantiations serial input. The subsections below contain the different scenarios and operations that were tested and verified using the setup with two Proximity-1 module instantiations.

Establish Communication Session

The hailing (session establishment) activity was tested by placing one module instantiation in the *ConnectingL* mode and the other in the *ConnectingT* mode, using the directive Set Mode. The instantiations successfully established a communication session and the MAC blocks ended up in the Data Service state as expected (see section 4.7.5 for the details about the hailing activity).

Send and Receive Sequence Controlled and Expedited Data

While in the Data Service state, different forms of data packets were sent to the Frame Create blocks of both module instantiations. The transmission of data from one module to the other was verified by comparing the data input at the Frame Create blocks in one module with the data output from the Router Addresser blocks of the other module. Both data using the Expedited and Sequence Controlled QoS was issued for transmission. The transmission of data packets of different size was also tested and validated, i.e. both data packets fitting into one frame as well as data packets segmented over several frames.

Comm Change

The Comm Change activity was tested by issuing a local Comm Change directive to one of the module instantiations. Prior to the Comm Change directive, new values were written into the transmitter and receiver configuration registers of the module. The module receiving the local Comm Change directive sent a remote Comm Change directive to the other module instantiation with the new transmitter and receiver configurations, and the modules successfully reconfigured to the new values.

Carrier Lost

The Carrier Lost scenario (described in section 4.7.6) was tested by forcing the Physical layer Carrier Acquired input signal to false when both modules was in the Data Service state. This caused them to try to set up a new communication session through hailing as described in section 4.7.6). The Carrier Acquired signal was forced to true again, and the modules successfully reestablished the session.

Resync

The Resync operation is triggered when the Proximity-1 module receives an invalid PLCW and the ARQ COP-P process requires resynchronization (see section 4.6.1 and 4.7.7 for further details). An invalid PLCW was formed and sent as a protocol packet to one of the module instantiations Expedited Frame Create block. This PLCW was handled as a normal PLCW in the receiver of the protocol frame, and hence the Resync operation was triggered in the other module instantiation when it received the invalid PLCW. After the Resync operation was triggered in the receiving module, it performed the resynchronization operations detailed in section 4.7.7, and a resynchronization was successfully performed.

PLCW Generation Register

Both options for PLCW generation described in section 4.3.10 were tested using the PLCW generation configuration register. Both the option of generating a PLCW as soon as a Sequence Controlled frame is received and the option of waiting until $WindowSize - 1$ frames has been received were verified.

Terminate Communication Session

Session termination was tested in two ways. In the first test, both modules were given the directive Local No More Data. This initiates actions which makes both modules aware of that neither of them have any more data to send. The session terminated as expected. In the second termination test, the Set Mode Inactive directive was issued to force the MAC block to reset the Proximity-1 module and close the communication session. This also worked as expected.

5.1.2 Proximity-1 Module and Host Rover System Functional Tests

The Proximity-1 module was connected to an existing rover host system (provided by RUAG Space AB) in order to verify the interfaces between the Proximity-1 module and a rover OBC. The rover host system contains memory handling blocks that can be used to write and read data packets to the Proximity-1 modules data input and data output ports.

Two instantiations of the Proximity-1 module were connected to the same rover host system in order to simulate and test a communication session. Although the modules were connected to the same rover host system, they used separate memory handling modules and separate configuration register interfaces. Hence, the modules did not affect each other. The memory handlers were connected to the same shared memory, but this did not pose a problem since the Proximity-1 module would have to compete with other modules for shared memory access in a real rover host system as well.

Similar to the simulation set up, the serial transmit and receive pins of the two module instantiations were wired directly to each other, yielding a communication link with no packet loss. A figure of the rover host system set up containing the Proximity-1 modules can be seen in Figure 5.1.

The register interface blocks are used by the OBC to send configurations and directives to the Proximity-1 modules, and to receive status and interrupts from the Proximity-1 module. Each Proximity-1 module uses a set of memory handlers, grouped

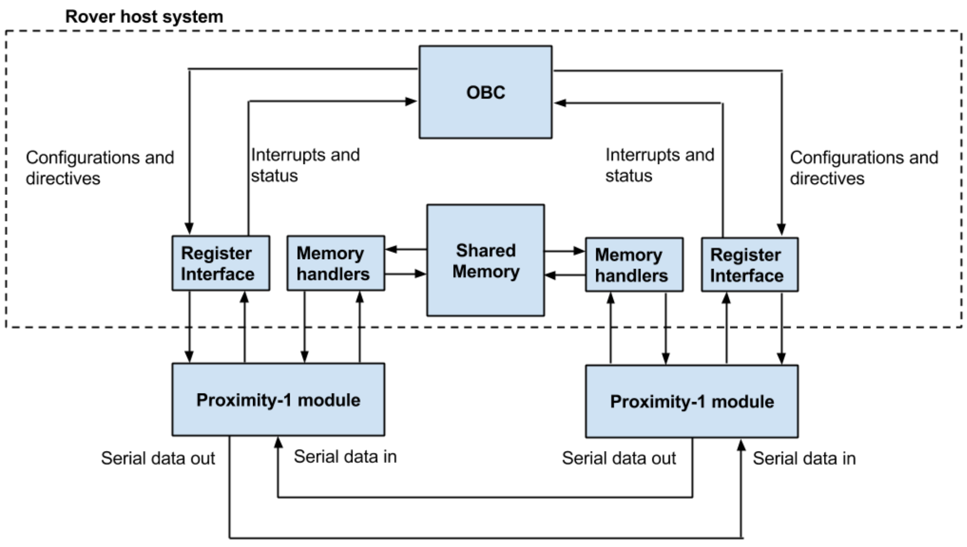

Figure 5.1: Rover host system and Proximity-1 module set up

into one block in Figure 5.1. The memory handlers are used to send Expedited and Sequence Controlled data packets from the rover memory to the inputs of the Proximity-1 module. The memory handlers also reads Sequence Controlled and Expedited frames from the Proximity-1 module outputs and places them into memory. The router byte attached to each outgoing frame is used to address the correct memory handler.

The same simulation tests as described in section 5.1.1 were performed on the set up in Figure 5.1. However, the Comm Change and Carrier Lost tests were not performed, since these tests require control over the Physical layer Carrier Acquired signal. This signal could not be controlled from outside when the Proximity-1 module was incorporated into the host rover system.

In the simulations it was not the OBC that issued register writes to the register interface since this functionality does not exist in the OBC software yet. Instead, register writes were performed in a simulation test to mimic the writes an OBC would do in a communication session. Data packets to be sent were not placed by the OBC in memory, but was instead generated by the simulation test and placed in memory.

After the rover host system and Proximity-1 module set up had been verified by the simulation tests, the complete set up was synthesized onto a FPGA to further verify functionality. The memory content, memory handlers and register operations in the synthesized rover host system could be controlled and monitored by an external computer over Ethernet.

The aim of testing the set up on FPGA was to verify that the basic functionality of the prototype worked after synthesis. The majority of the tests that were performed on the rover host system in simulations were ported to the scripting language TCL, which was used to control memory and register operations in the FPGA. The TCL scripts ran on an external computer, controlling the rover host system over Ethernet. The Resync test was not ported to TCL since it was considered as corner case functionality, and the aim of the FPGA test was to verify the fundamental functionality of the prototype. The more exhaustive tests were performed in simulation as described earlier in this section.

5.2 Module Synthesis Results

The FPGA synthesis results for the Proximity-1 module can be seen in table 5.1. Note that the values are the synthesis results for one Proximity-1 module, and not the complete host rover system.

Table 5.1: FPGA synthesis results for the Proximity-1 module

Type	**Number**
LUTs	1746
Slice registers	5836

5.3 Data Throughput Evaluation

To evaluate the performance of the protocol and module, a data throughput test is done in which we consider a scenario where both the transmit and receive links are fully loaded while transmitting a data set. Since the Sequence Controlled data service requires the transmission of acknowledgments, PLCWs, transmitting data over the Sequence Controlled service generates more traffic than transmitting the same data over the Expedited service. We thus consider a scenario where both transceivers of a proximity link simultaneously transmit a set of data, a payload, of a predetermined size using Sequence Controlled quality of service (QoS). To simplify matters, we assume that both links use the same data rate, and that the transmission of both data sets start at the same time. We also assume no packet loss.

Because PLCW transmission is triggered by received Sequence Controlled data frames, we must consider how the payload is segmented into frames. The simplest way to control this is to use unsegmented data frames, and divide the payload data into a number of packets. Using this type of segmentation, each data packet can be transmitted in one frame. For simplicity, the rest of this test section will use the words packet and frame interchangeably since one packet fits into one frame.

Since both the transmit and receive links send the same data using the same data rate, they are entirely symmetrical in terms of transmitted data. The only difference between the links is the order of data packets and PLCWs, since PLCWs transmission is triggered by the reception of data packets. However, this does not impact data throughput significantly, especially when a large number of packets is used. We thus consider the transmission time of both links identical, and only consider one link in the evaluation.

The transmission time t_{tx} of a payload depends on the transmitted payload data size $s_{payload,tx}$, the transmitted PLCW data size $s_{plcw,tx}$, and the data rate r_{data} according to (5.1). The transmitted payload data size corresponds to the total data size required to transmit the payload, including the overhead caused by the frame segmentation (5.2). The overhead of a frame $s_{frame,overhead}$ is everything added to the frame on top of the data, i.e. the frame header, ASM and CRC. Similarly, the transmitted PLCW data size is the accumulated data size of all PLCWs, that is the number of PLCWs n_{plcw} times the size of each PLCW s_{plcw} (5.3).

$$t_{tx} = \frac{s_{total,tx}}{r_{data}} = \frac{s_{payload,tx} + s_{plcw,tx}}{r_{data}} \tag{5.1}$$

$$s_{payload,tx} = s_{payload} + s_{frame,overhead} n_{packets} \tag{5.2}$$

$$s_{plcw,tx} = s_{plcw} n_{plcw} \tag{5.3}$$

Combining the expressions for the two types of data being transmitted over the link yields the following expression for the transmission time:

$$t_{tx} = \frac{s_{payload} + s_{frame,overhead} n_{packets} + s_{plcw} n_{plcw}}{r_{data}} \tag{5.4}$$

Because of the assumptions that both links transmit the same payload using identical data rates and segmentation (i.e. identical r_{data} and $n_{packets}$), we can express the number of PLCWs n_{plcws} as a function of the number of data packets $n_{packets}$. Recall that the proximity standard recommends that a PLCW is transmitted upon the reception of each valid Sequence Controlled frame, i.e. each received data frame generates a PLCW frame on the transmit link. This yields the relationship in (5.5). We also consider the optimized PLCWs proposed in Section 4.3.10, where each PLCW acknowledges a number of data frames n_{acked} at the same time, yielding (5.6). $n_{acked} = 1$ is the same as the Proximity-1 standard recommendation, as each received frame triggers the generation of a PLCW. All other values of n_{acked} correspond to the option of using the added configuration register where n_{acked} corresponds to the $WindowSize - 1$. Note that if $n_{acked} = 1$, both relationships are identical, we thus use the more general expression in (5.6) in our model, and expose n_{acked} as a parameter. This gives us the model equation (5.7).

$$n_{plcw} = n_{packets} \tag{5.5}$$

$$n_{plcw,optimized} = \left\lceil \frac{n_{packets}}{n_{acked}} \right\rceil \tag{5.6}$$

$$t_{tx} = \frac{s_{data} + s_{frame,overhead} n_{packets} + s_{plcw} \left\lceil \frac{n_{packets}}{n_{acked}} \right\rceil}{r_{data}} \tag{5.7}$$

To validate this model, we simulate the scenario using two proximity modules that send data to each other, with the parameters $s_{payload} = 10240B$, $r_{data} = 256kB/s$. We

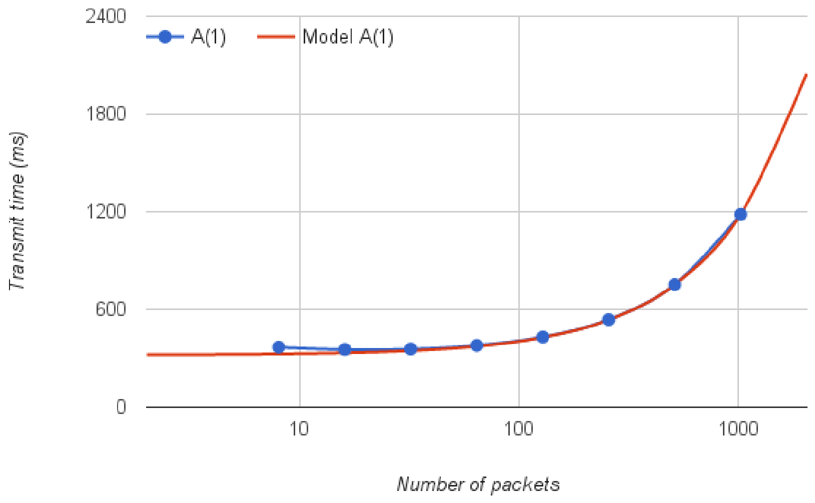

Figure 5.2: Simulated transmit time and analytical model for the proximity standard implementation, data size is 10240 bytes, data rate is 256 Kb/s.

sweep the number of packets as $[8, 16, 32...1024]$, and also use different PLCW acknowledgment strategies. In the rest of the text, we call our analytical model given by equation (5.7) $A(n_{acked})$ where the parameter n_{acked} denotes how many frames each PLCW acknowledges. The models and the simulation results are plotted in Figure 5.2, Figure 5.3 and Figure 5.4.

The simulation results lie very closely to the analytical model, especially for A(1), acknowledging every frame (which is the Proximity-1 standard recommendation). For A(3) and A(15) (which uses our added configuration register to acknowledge several frames), the simulation results is consistently slightly offset from the model. This is most likely due to the last PLCW being triggered by a timer rather than a data frame. Since the number of packets (and thus frames) is never divisible with n_{acked}, the acknowledgement of the very last data frames are never automatically triggered, and must wait for the PLCW timer before it is transmitted. This gives a slight increase in transmission time compared to the analytical model.

Another interesting observation is the slight increase in simulated transmission time when the number of packets approaches 0. This is not predicted by the model, but can be attributed to delays caused by the streaming of packets internally in the module. For the leftmost point, the number of packets is 8, giving a packet size of 1280 bytes. This frame is too large to be streamed to the serializer block quick enough to fully fill the link's symbol stream, resulting in the transmission of one or several idle bytes (see Section 4.3.8). Since we assume that all payload packets are the same size, this happens consistently for each data packet. This is analogous to considering idle bytes

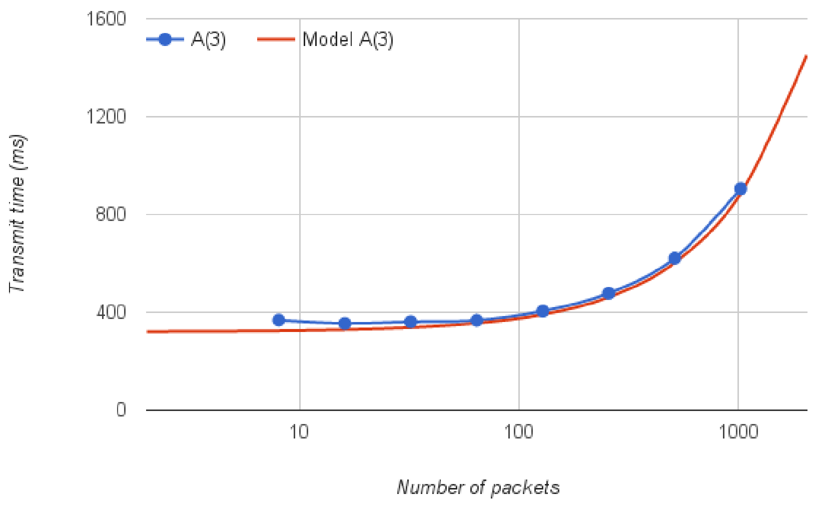

Figure 5.3: Simulated transmit time and analytical model for throughput optimization of window size 4, data size is 10240 bytes, data rate is 256 Kb/s.

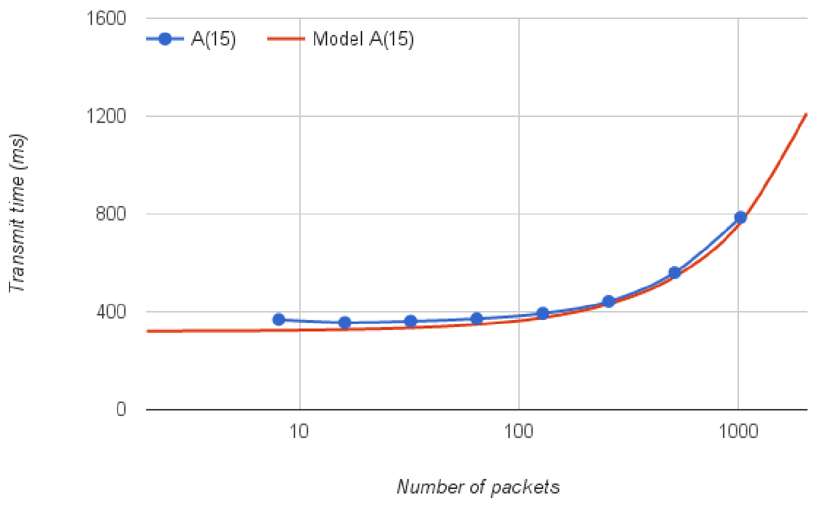

Figure 5.4: Simulated transmit time and analytical model for throughput optimization of window size 16, data size is 10240 bytes, data rate is 256 Kb/s.

as overhead bytes in (5.2). Note that this effect is largely implementation-based, and a lower data rate will increase the maximum packet size that can be sent without requiring idle frames. This effect may thus be avoided entirely by reducing the data rate.

5.3.1 PLCW Acknowledgment Optimization

Let us evaluate the effects of acknowledging many frames in each PLCW. We start by defining a performance metric that does not depend on the data rate, the link efficiency (LE). The link efficiency is the percentage of the data transmission that is spend transmitting useful data, i.e. the payload data in the evaluation scenario. We define the link efficiency according to (5.8). Substituting t_{tx} according to (5.1) yields (5.9). Further substitution of $s_{total,tx}$ yields the expression in (5.10). We plot the link efficiency using $s_{payload} = 10240B$ in Figure 5.5.

$$LE = \frac{\frac{s_{payload}}{t_{tx}}}{r_{data}} \tag{5.8}$$

$$LE = \frac{s_{payload}}{s_{total,tx}} \tag{5.9}$$

$$LE = \frac{1}{1 + \frac{s_{frame,overhead} n_{packets} + s_{plcw} \left\lceil \frac{n_{packets}}{n_{acked}} \right\rceil}{s_{payload}}} \tag{5.10}$$

In Figure 5.5 we see that increasing the acknowledgment interval n_{acked} maximizes the link efficiency since less PLCWs are generated, but increasing it yields a diminishing return. If we had an infinite acknowledgment interval, the PLCW overhead term disappears from (5.10), which is analogous to never sending PLCWs at all. This is of course not possible in practice, where the maximum acknowledgment interval is equal to the window size. The maximum window size is 128 in the Proximity-1 standard [4]. We can now define the expression (5.11) for the maximum link efficiency when using the acknowledgment interval optimization.

$$LE = \frac{1}{1 + \frac{s_{frame,overhead} n_{packets} + s_{plcw} \left\lceil \frac{n_{packets}}{128} \right\rceil}{s_{payload}}} \tag{5.11}$$

From Figure 5.5 we can conclude that a small increase in the acknowledgment interval can offer an increase in overall data throughput under the assumption that the payload data is divided into a moderately high number of frames (100+) and that packet loss is negligible.

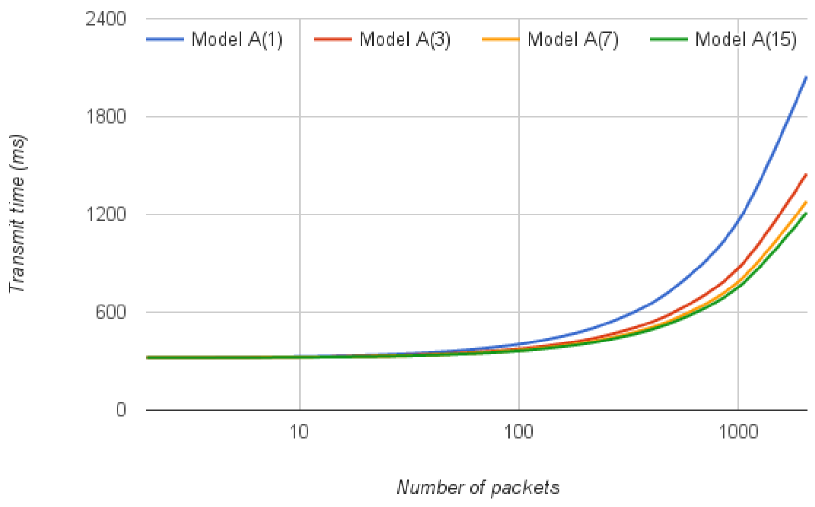

Figure 5.5: Analytical model plots for throughput optimization of different window sizes, data size is 10240 bytes, data rate is 256 Kb/s.

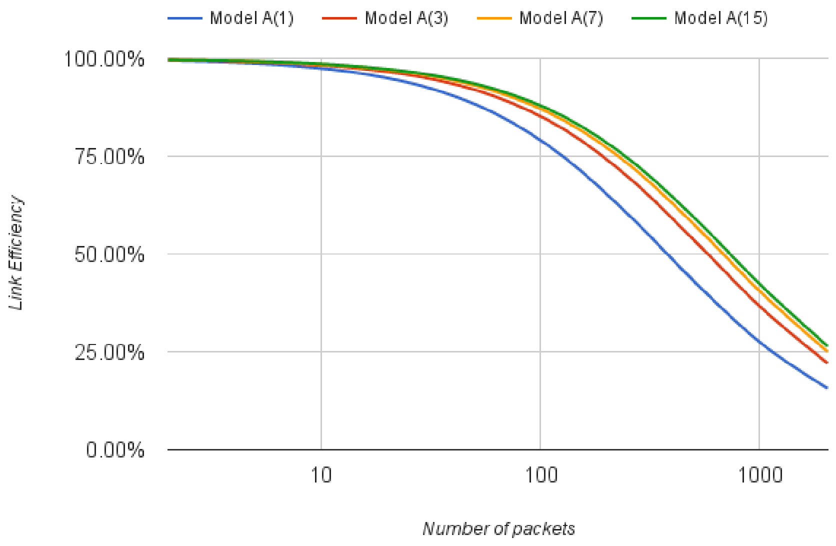

Figure 5.6: Analytical model plots for the link efficiency for different number of packets and PLCW acknowledgements.

6

Discussion

In this discussion chapter, we will compare our implementation to another Data Link Layer implementation. The performance improving PCLW acknowledging register is also discussed. In addition, the flexibility of the design is described and the extent to which our module handles the known performance issues of the Data Link Layer is discussed. At the end of the chapter, we will relate back to the research questions presented in Section 1.2 and try to answer those questions.

6.1 Comparison with Other Proximity-1 Implementation

In order to evaluate our design decisions it is interesting to compare our design to other implementations of the Proximity-1 Data Link Layer. It is however difficult to find detailed papers on other implementations, probably because many of the implementations are done in industry where company secrecy prohibits the publishing of details. Nevertheless, we managed to find one recently published paper which describes a project similar to ours. The paper is published by Sharma et al. who also implemented the Proximity-1 Data Link Layer on an FPGA [11]. Their implementation also focuses on implementing the Data Link Layer for full duplex operation, suitable for communication between a satellite and rover. Since the Proximity-1 standard documents describe the functionality of the protocol in much detail there are naturally many similarities in the functional design when comparing ours and the Sharma et al. implementation. However, differences in interfaces, block partitioning and storage of data are evident.

In our implementation, the design is divided into three major parts; transmit stage, receive stage and MAC sublayer. Sharma et al. have done the same partitioning. The subsections below will discuss some of the similarities and differences of each part.

6.1.1 Transmit Stage Comparison

At the top of the transmit stage is the interface between the I/O sublayer and the user. In our implementation, the header information about Quality of Service, Port ID etc. is sent together with the data packet. Sharma et al. has instead decided to place the header information as input signals into the I/O sublayer, and the data is supplied through a separate interface. This means that the user must change the header signals dynamically when data is sent into the I/O sublayer. Our implementation relieves the user from dynamically having to control inputs to I/O sublayer since the header information is part of the data sent into the I/O sublayer. After the OBC has created the data packet, it does not have to give any other input information about the packet to our Proximity-1 module. This means that the OBC can create a lot of packets to be sent, place them in memory and then continue with other operations. When the data packets are to be sent, the OBC can just tell a memory handling module to send the data packets to the Proximity-1 module. This is in accordance with one of our design goals; to make the Proximity-1 module as independent as possible from the OBC.

Both in our and the Sharma et al. design the Expedited and Sequence Controlled data has separate inputs to the I/O sublayer in order to make it possible to send Expedited data when the Sequence Controlled data queues are full, and vice versa. Another similarity is that both designs store the whole frame in memory, i.e both the header and the data. As identified by Sharma et al., this makes the retransmission of frames more efficient since the frame header does not have to be reformed if data has to be resent [11].

The failure of a resync activity is handled somewhat differently in our and the Sharma et al. designs. If the resync activity fails in our implementation, the MAC goes back to the idle state, i.e, the communication session is ended. Sharma et al. has taken a more advanced approach. When the resync activity fails in their design the variables of the COP-P process is modified to correspond to the last sequence number that the two transceivers agreed upon and stored unacknowledged Sequence Controlled frames are discarded [11]. This means that the COP-P process is reset to the last known, working state. This approach is something that could be implemented in our design as well in a future update.

6.1.2 Receive Stage Comparison

In our design, we have chosen to divide the design into blocks corresponding to the different sublayers of the Data Link Layer and their functionality. It was done since it seemed as a suitable division to mirror the protocol layers in the block design. This has been done on both the receive and transmission parts. Sharma et al. however decided to implement the receiving part as one large block instead of several small ones. Their motivation for this is that it removes much of the communication interfaces needed between the sublayers of the protocol [11]. If several blocks are used to implement each sublayer, more communication interfaces are needed. Since the receiving side contains less complex functionality compared to the transmitting side, it is possible to implement the receiving side as one large block containing all the sublayers functionality [11]. Even though the Sharma et al. receiving side implementation might reduce interface communication and make the design more compact, our implementation with many small blocks makes it easier to relate each block functionality with the corresponding protocol sublayer functionality. This in return will probably make it easier to change features of the Proximity-1 module in future expansions, since it is easy to know which block that implements which protocol functionality.

6.1.3 MAC Sublayer Comparison

In both our and the Sharma et al. design there is only one interface between the Proximity-1 module and the user (OBC) where configurations and directives can enter the module, and status registers and notifications are sent to the user. In the Sharma et al. design, the MAC handles this interface in contrast to our implementation where there is a register interface.

In the Sharma et al. implementation, only the MAC can create and send protocol configurations and commands (i.e protocol data). Their I/O sublayer implementation does not have an input which indicates if the supplied data packet contains protocol or user data, and hence the I/O sublayer only supports user data packets as input. In our implementation, both the MAC and the OBC can send protocol data. When the OBC wants to send protocol data packets in our implementation, it simply sends it as an Expedited data packet which will enter at the I/O sublayer input (the Frame Create block in our design). However, there are no functional requirements that the OBC have to send a protocol data packet, since the MAC can send all the protocol commands as well in our design. The possibility of the OBC sending a protocol data packet has been added since it might be needed in future expansions of the module. Another possible use is during error cases, when it could be useful for the OBC to send protocol directives to force the remote transceiver to perform certain operations and configuration changes.

6.2 Discussion of Known Proximity-1 Performance Issues

In Section 2.3 four potential performance issues of Proximity-1 Data Link Layer were described. In the subsections below, each issue will be discussed in relation to our implementation.

6.2.1 Combination of Data Size and Quality of Service

The first problem, i.e the choice of data size and QoS (Section 2.3.1), is an issue that the user of the system is responsible for handling. The problem cannot be handled within the Proximity-1 module since it depends on the supplied data and frame header information which are both set by the OBC.

6.2.2 Loss of Response PLCW in Hailing

The second problem, related to the potential loss of response message in the hailing activity (Section 2.3.2), could be be addressed within the Proximity-1 module since an implementation could decide to send two PLCWs instead of only one as a response to the hail. However, it can not be guaranteed that the second PLCW is not missed as well. Therefore, our implementation does not support the transmission of two PLCWs as a response to a hail. It is again up to the user of the system to do as suggested in Section 2.3.2 and carefully select the MIB Parameter timing values related to the hailing activity.

6.2.3 Combination of Frame Size, Window Size and Data Rate

The third issue, which consist of the relation between the frame size, window size, data rate and PLCW acknowledgements (Section 2.3.3), is to some degree handled in our Proximity-1 module. In our design, the frame sequence number that is sent in a PLCW is always the number of the most recently received frame. Therefore, if a PLCW is stalled because another frame is currently transmitted, and a new Sequence Controlled frame is received, the frame sequence number of the PLCW is overwritten to correspond to the most recently received frame. This reduces the number of PLCWs that need to be sent in a communication session and improves the link efficiency as described in Section 2.3.3.

However, the PLCW generation implementation that we have adopted only solves half of the problem described in Section 2.3.3. The user of the Proximity-1 module is still responsible for selecting suitable combinations of window size, transmission data rate and reception data rate in order to maximize the link efficiency. Since these parameters only can be set by the OBC, it is not within the scope of the Proximity-1 module.

6.2.4 Interfaces to the Proximity-1 Module

The fourth issue, which is related to potential interface problems between the Proximity-1 module, the OBC, the Physical layer and transceiver (Section 2.3.4), has to some degree been evaluated in our tests and simulations. Our Proximity-1 module was incorporated into an existing rover host system, and the register interface between the host system and Proximity-1 module was tested and validated in both simulation and on FPGA. Hence, the interface between our module and the OBC has been verified to work. However, the scope of this project did not cover the implementation of a Physical layer, and hence the interface between the Proximity-1 module, the Physical layer and a transceiver could not be done. This is something that should be done in future developments of the system. In addition, it would be interesting (and probably necessary in a final product design) to try to set up a communication session between our implementation and another Proximity-1 Data Link Layer implementation.

6.3 Using PLCW Acknowledgment Optimization

In Section 2.3.3 and 6.2.3, the importance of the PLCW acknowledgment implementation for the protocol performance is discussed. However, these sections describe the PLCW acknowledgment in relation to the problems that can arise when the transmission and reception data rates are different. In our project we investigate another user scenario also, namely when when the transmission data rate and reception data rate are the same.

In the Proximity-1 module we added a configuration register which can be used to optimize the PLCW acknowledgements and communication link performance when the transmission and reception data rates are the same (see Section 4.3.10 and 5.3). Using the new register, the user can decide if the PLCW Encoder block should send a PLCW as soon as a Sequence Controlled frame has been received, or if it should wait until a certain number of frames has been received (equal to $(WindowSize - 1)$, as described in 4.3.10). When the first option is used, the PLCW Encoder works as defined in the Proximity-1 standard. A PLCW is generated when a Sequence Controlled frame has been received. If another frame arrives before the PLCW has been sent, the PLCW is overwritten to acknowledge the most recently received frame. The two methods and their respective performances were described using analytical models and simulations in Section 5.3.

If the first register option is used, a PLCW would be generated for every single frame received. If the Sequence Controlled frame size is around the same size as a PLCW, approximately 50% of the data sent will be PLCWs. If the second register option is used where the PLCW Encoder waits for a certain number of frames to arrive, PLCWs are sent less frequently. This option will yield a performance improvement when small Sequence Controlled frames are sent from both transceivers in a communication session. Less PLCWs are generated, and therefore the throughput of Sequence Controlled frames increases since less of the data transmitted is occupied by PLCWs.

When the Sequence Controlled frame size increases, the second option will become less effective since the ratio between the PLCW size and frame size increases. Thus most of the data sent will consist of Sequence controlled data whichever acknowledgment option that is used, since the PLCWs are very small compared to the large Sequence Controlled frames.

Another thing to keep in mind when using the option of waiting for some frames before generating a PLCW is that this option can yield a low data throughput when the sending and receiving data rates are different. If Sequence Controlled frames are received with a high data rate, but the data rate of sending frames and PLCWs are low, a PLCW should be generated as soon as possible in order to avoid retransmission of Sequence Controlled frames. If the option of waiting for some frames before generating a PLCW is used, a PLCW might be sent to seldom and unnecessary retransmissions of frames will occur since the sender does not receive a PLCW in time. Therefore, the first option of generating a PLCW whenever a frame is received is better to use in this case.

Another weakness of our new PLCW generation option is that the number of frames that need to be received before a PLCW is generated is always set to the current window size minus one. An improvement could be to provide a register where the OBC can choose a suitable value for the number of frames that should be received before a PCLW is generated. This would give the OBC more freedom to use the new option during different communication link conditions. For example, it would probably be preferable to send PLCWs more frequently when the channel signal is weak (and creates more errors in the frame transmission). Moreover, if the link creates many errors in the frames and frames are lost, it is probably better to use the option of sending a PLCW as soon as a frame arrives.

The results of the link efficiency evaluation in Section 5.3 highlights the importance of using large frame sizes when transmitting data. This reduces the impact of frame and acknowledgment (when using the Sequence Controlled service) overhead. In a real scenario, instruments on modern rovers are expected to generate large amounts of data [3], but there are also small data packets such as status reports [12]. Our optimization may prove useful if a lot of these smaller packets are to be transmitted, and the OBC has not assembled the smaller packets into one large packet. It is also important to note that our optimization significantly decreases the impact of acknowledgments, but frame overhead is still a major concern for small data frames.

6.4 Flexibility

One of the main design goals was to make the Proximity-1 Data link module as flexible as possible. In the subsections below, the areas of flexibility in our design are discussed.

6.4.1 Interface Flexibility

There are three interfaces to our design; data input, data output and register interfaces. The data input can easily be connected to any module that follows the handshake interface rules described in Section 4.2.1. The same goes for the data output. Since the Router Addresser block contains a configurable look-up table to map the Port ID of the frame header to byte addresses, a user can set the byte addresses to any value that the local system wants to use. If a user does not want to add a byte address the transfer frames, the block can easily be disconnected from the design by reconnecting some signals in the code.

Since all configurations, local directives and interrupts are collected into one well-defined register interface, it should be easy for the user to use these registers to control the Proximity-1 module.

The internal interfaces, i.e the interfaces between the blocks of the Proximity-1 module, all follow the same handshaking rules. It should therefore be easy to connect new blocks with additional functionality to the design in the future. As long as the blocks follows the handshaking rules, it is easy to connect a block between two existing blocks of the design. For example, in a future expansion of the module, the error correcting code Reed-Solomon will probably be added to the design. As shown in Figure 4.5 in Section 4.3, the functionality required for this can easily be added as a block between the CRC Generator block and ASM block.

6.4.2 Block Flexibility

The block division of our design have been made to follow the sublayer functionality of the Proximity-1 standard. This makes it easy to relate the functionality of each sublayer to the blocks in the design. This will make it easier to modify or add functionality into the blocks in future, since it defines what block is responsible for what functionality of the protocol.

6.4.3 Interfaces for Timing Services

Even though the timing services of the Proximity-1 are not implemented in our design, all required interfaces and registers are implemented. This will make it easy to add the functionality needed for the timing services in a later prototype.

6.5 Prototype Suitability for Space

As described in Section 3.2, space contains radiation and high-energy particles which can interfere with hardware and for example cause unwanted bit flips in registers. Thus, we adopted several coding guidelines in the development of our Proximity-1 module to prevent erroneous or undefined behavior caused by outside, unwanted triggers (see Section 3.2). The used coding guidelines prevent the module from using undefined variables and states. In addition, the usage of the handshaking communication interface between blocks makes the interfaces more robust. However, there are more things that can be done to further improve the design robustness as described in the subsections below.

6.5.1 Triple Registers and Voting

Registers in the design that holds important values could be tripled. This means that three registers are used to hold the same value. When the value is used, the three register values are compared and if all three registers or two registers agree on the value, the value is used. This process is called voting, since three registers votes for which value that is to be used. This set up prevents erroneous behavior caused by single bit flips, e.g. if one register is experiencing a bit flip error, the other two registers will provide the correct value. In addition, this technique can correct faulty register values, since the voted correct value is written back to all three registers after the voting. This means that if one of the registers contained a faulty value, it will be corrected. However, if all three registers would contain different values, it would mean that two bit flips have occurred in separate registers at the same time. In this case, some decision logic is needed to decide which value to use.

Since the register implementation becomes more complex and takes up more space when using triple registers and voting, it is desirable to only use this technique on registers that can cause unresolvable errors to the module behavior. In our design, it would probably be necessary to triple many of the registers contained in the MAC block, since these are used to control the operation of the whole Proximity-1 module. In addition, several of the configuration registers also contain values which are important for operation of the module and verification of transfer frames. Therefore, they would probably also benefit from being tripled.

6.5.2 Error Checking Code on Data Stored in Memory

A common method used for space applications to protect data stored in memory is to attach error checking code to the data. The code is attached to the data that is stored, and when the data is read from memory the code is used to check if the data is corrupted or not. The Hamming code is an example of an error checking code [9]. One common way to use it is to construct it to correct one error and detect two errors in the data it is applied to.

Error checking codes for data written to memory is something that could be added to our module in order to make the data written to memory less vulnerable to bit flips. All the dedicated and the shared memory used in our module would benefit from using this, but which ones are most important? Typically, the error checking codes are beneficial to use where data is stored for a long time since the probability for data corruption increases the longer the data is stored. In our design, the frames stored by the Frame Create block in the shared memory could be stored for a long time before they are selected for transmission, and hence the shared memory should use error checking code.

Data stored in the output single buffer in the transmit stage should be read from the buffer fairly regularly, since the frame kept in the buffer is transmitted as soon as possible to the PLTU Formation block. The same is true for the receiving sides double buffer (the one placed before the Frame Dispatch block), since frames should be read out from the buffer by the Frame Dispatch block as fast as possible in order not to stall the input link. However, if error checking code is applied to this buffer and an erroneous Sequence Controlled frame is detected, the frame can be discarded before it is sent to the Frame Dispatch block. This would mean that no acknowledgment is generated for the frame, and therefore the frame will be sent again.

The two output double buffers for Expedited and Sequence Controlled frames might benefit from using error checking codes, depending on how often the module that receives frames from the Proximity-1 module reads the buffers. However, these buffers should be read fairly often by an outside block, since data arriving when the data buffers are full will be thrown away by the flusher blocks.

6.6 Discussion Summary - Answers to Research Questions

As a summary to this discussion chapter, the three research questions stated in 1.2 will be answered and discussed.

6.6.1 Implementing the Data Link Layer in Hardware

Initially, the performance of the Proximity-1 module will be discussed. Notably, the functional performance of the blocks in the design is not a limiting factor in this type of module. The reason is that the clock rate used for operations within the module is much higher than the clock rate used for the serial output and input to the module. Therefore, no performance issues should arise from that the input or output serial data stream is stalled due to that a block not operates fast enough. As an example, the host rover computer that we used in our tests have a clock rate of 32MHz, while the maximum clock rate of the serial clock is 2MHz according to the Proximity-1 standard [4].

Something that on the other hand can impact the performance of the protocol is if data not flows evenly through the module and blocks that requires a constant data throughput are stalled. To avoid this from happening, several buffers are used in the Proximity-1 module. As discussed in Section 4.2.2 and 4.5, the buffers ensure that the data flows through the module in a coherent manner. In addition, the single output buffer in the transmit stage and the double receiving buffer in the receive stage ensures that the blocks in the Coding and Synchronization sublayer that requires a constant data throughput not are stalled.

As discussed in Section 6.2.3 and 2.3.3, the performance can be severely reduced when certain combinations of frame size, transmission window size, transmission data rate, receiver data rate and PLCW acknowledgment methods are used. In order to minimize the problem, the Proximity-1 module in our design implements acknowledgment of several frames in one PLCW as recommended by the Proximity-1 standard [2]. In order to maximize the throughput of data under the user scenario when the transmitter and receiver data rates are the same, our design implements an additional register which can be used to configure the Proximity-1 module to only generate acknowledgments after a certain number of frames has been received.

The flexibility of the Proximity-1 module is discussed in Section 6.4.1. To summarize, the module provides flexibility both in terms of connecting the overall module to a rover host system as well as providing flexible connections between blocks of the design. All the control information (configurations, directives, status and interrupts) that is exchanged between the Proximity-1 module and the OBC is placed in one register interface. The data input and output to the module follows a set of simple handshaking rules, and any module which follows these rules can send data to the Proximity-1 module or receive data from the Proximity-1 module.

The blocks within the Proximity-1 module follow the same handshaking rules and each block's functionality is clearly connected to the functionality of one of the protocol sublayers. This should make it easier to add additional features into the design in the future.

6.6.2 Functionality Preferably Implemented in Software

As discussed in Section 4.4.11, the assembly of frames into the original data packets are preferably done in software. The reason is that the software can offer more flexible memory handling and more easily allow for large packets.

Another functionality that also could be placed in software is the segmentation of data packets into transfer frames. Unlike the frame assembly, frame segmentation does not require a memory space that holds an entire data packet. Instead, packet bytes are streamed into the module and segmented into frames. In our module, the segmentation functionality is implemented in the Frame Create block. The implementation becomes quite complex in hardware since the Frame Create module needs to do segmentation of packets into frames and keep track of the start and end of data packets. The block also contains several internal register in order to keep track of frame sequence numbers, frame header values and segmentation of frames. Therefore, the functionality of creating frames could have been placed in software instead in order to ease the implementation process and make the hardware simpler. However, since the streaming input interface allowed for it, and the intention of this Proximity-1 module was to be as self-contained and hardware oriented as possible, the choice of having the frame formation in hardware was a necessary design decision in this implementation.

6.6.3 Avoiding Interoperability and Performance Issues

In Section 6.2 four different interoperability and performance issues identified by CCSDS were described and discussed in relation to our module. The first two problems, combination of data size and Quality of Service and the potential loss of response during hailing, are not handled in our Proximity-1 module. The third problem related to the combination of certain communication configurations such as window size, data rate and PLCW acknowledgment method, is partly handled in the Proximity-1 module by efficient PLCW acknowledgment methods. The reason that the Proximity-1 module does not handle the three first problems to a larger extent is because the problems are dependent on configurations that are supposed to be set by the user of the module. In order to preserve flexibility and configurability of the module it is important not to remove these options. However, this also puts more responsibility upon the user of the Proximity-1 module to choose configurations that avoid the problems identified by CCSDS.

Something that can be done by a user to avoid using poor configuration combinations is to implement functions and algorithms that calculate the optimal configuration values for different user scenarios. These functions could take the values that a user wants to use, and decide if the configurations are feasible or not. If not, the algorithm could propose other more suitable values. This functionality is something that could be implemented in software.

The fourth problem discussed by CCSDS, which can arise in the interfaces between the Proximity-1, the OBC and transceiver, have been handled as far as possible within the scope of this project. Since the module does not contain the Physical layer of the Proximity-1 protocol, only the interface between the Proximity-1 module and the OBC was tested and verified to work.

7 Conclusion

In this report, a flexible hardware implementation of the Data Link Layer of the communication protocol Proximity-1 has been described and evaluated. The implementation, called the Proximity-1 module, is able to set up and handle a communication session established with another Proximity-1 transceiver. The Proximity-1 module has been incorporated into an existing rover host system and been verified to work both in simulations and in functional tests on FPGA.

The Proximity-1 module provides flexibility both internally in the design and in the interfaces of the design. The same handshaking interface rules are used between major blocks internally in the design. In addition, each block has a well-defined functional task related to a Data link sublayer functionality. These two aspects will make it easy to add additional functionality into the Proximity-1 module in the future, if required.

The data input and output interfaces of the Proximity-1 module have been implemented in a way which should make it simple to connect them to any outside module in the host system. Two inputs, one for each Quality of Service, accept data packets to be transmitted, any transmission configuration is set through the data packet header or via configuration registers. On the data output, a router byte is attached to received data to allow routing in a network.

The efficiency of the data links of the Proximity-1 module has been ensured by buffering data in certain places of the design. The buffers provide a uniform data throughput throughout the whole design and ensures a constant data input where required. In addition, a new acknowledgment generation method that is especially efficient when sending small Sequence Controlled frames has been proposed, evaluated, and implemented in the design.

The Proximity-1 module was implemented in hardware to make it as stand-alone as possible from the On-Board Computer (OBC). In addition, the decision of placing the data header information together with the data in the input packets relives the OBC from dynamically having to change header information when packets are sent.

The flexible design does however put some requirements on the OBC software which configures the module. The data throughput can be low if the OBC software chooses a poor combination of configuration parameters. However, if these parameters were not configurable from the OBC, the flexibility of the design would be reduced.

Future expansions of the Proximity-1 module should include the implementation of the timing services required by the Proximity-1 standard. In addition, logic that handles the assembly of segmented packets is required for the data output from the module. Additional features such as Reed-Solomon error correction code and encoding such as LDPC could be added into the module as well.

8

Acknowledgments

This master thesis was carried out at Ruag Space AB in Sweden, and the authors would like to thank the staff at Ruag Space AB for the provision of the master thesis and for valuable support and advice throughout the development of the Proximity-1 prototype. Special thanks to our supervisor at Ruag Space AB, Tomas Larsson, who has been our primary source of guidance throughout the project. The authors would also like to thank the supervisor at Chalmers Sven Knutsson and the examiner Per Larsson-Edefors for valuable feedback concerning the academic part of the thesis work.

Bibliography

[1] CCSDS. (2004, April) Press Release:CCSDS Proximity-1 communications protocol enables high-speed communication at Mars. Accessed: 2015-02-20. [Online]. Available: http://public.ccsds.org/outreach/PressRoom/Prox-1%20Key%20to%20Mars%20Comms.pdf

[2] CCSDS, *Proximity-1 space link protocol - rationale, architecture, and scenarios.*, Dec 2013, Informational report, CCSDS 210.0-G-2, Green Book, Issue 5, Washington DC, USA.

[3] C. D. Edwards, "Relay communications for Mars exploration," *International Journal of Satellite Communications and Networking*, vol. 25, no. 2, pp. 111–145, 2007.

[4] CCSDS, *Proximity-1 space link protocol - data link layer.*, Dec 2013, Recommended Standard, CCSDS 211.0-B-5, Blue Book, Issue 2, Washington DC, USA.

[5] C. D. Edwards, Jr. and R. DePaula, "Key telecommunications technologies for increasing data return for future mars exploration," *Acta Astronautica*, vol. 61, no. 1–6, pp. 131 – 138, 2007, bringing Space Closer to People, Selected Proceedings of the 57th {IAF} Congress, Valencia, Spain, 2-6 October, 2006.

[6] C.D. Edwards, et al., "Relay support for the Mars science laboratory and the coming decade of Mars relay network evolution," 2012, IEEE Aerospace Conference, United states.

[7] CCSDS, *Proximity-1 space link protocol - physical layer.*, Dec 2013, Recommended Standard, CCSDS 211.1-B-4, Blue Book, Issue 4, Washington DC, USA.

[8] CCSDS, *Proximity-1 space link protocol - coding and synchronization sublayer.*, Dec 2013, Recommended Standard, CCSDS 211.2-B-2, Blue Book, Issue 2, Washington DC, USA.

[9] F. Hargrave, *Hargrave's communications dictionary.* New York: Wiley-IEEE Press, 2001.

[10] ECSS, *Space engineering - SpaceWire - CCSDS packet transfer protocol .*, Feb 2010, ESA Requirements and Standards Division, The Netherlands.

[11] A. Sharma, et al., "Development of CCSDS Proximity-1 protocol for ISRO's extraterrestrial missions," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference)*, Sept 2014, pp. 2813–2819.

[12] ECSS, *Space engineering - Ground systems and operations — Telemetry and telecommand packet utilization.*, Jan 2003, ESA Publications Division, The Netherlands.

A

Appendix

A.1 Hailing Sequence

The hailing sequence states are shown in Figure A.1. The states are the same as described in the Proximity-1 standard [4]. However, the special Waiting States (not shown in the state diagram) have been added in order to implement the handshaking interfaces correctly. As seen in the hail sequence state diagram, the sequence can be entered from two other states; the Idle state or the Reconnect state. The hail starts by the MAC configuring the Physical layer to send a carrier in the state ”Start Hail Action”. After this modulation is turned on and idle data is modulated onto the carrier in the state ”Send Hail Acquisition”. These two initial states are needed in order to set up a physical communication link to the remote transceiver. Then the MAC tells the Directive Generator block to transmit a hail directive (i.e Set Transmitter Parameters and Set Receiver Parameters directive) to the remote transceiver. When the directive has been sent (indicated by that the single buffer is empty), a tail sequence (idle data) is radiated and then the MAC waits for a hail response (PLCW) to be received. It waits for the time specified in the MIB Parameter register Hail wait Duration, then if no response have been received it will try the hail sequence again. If no response have been received in a certain number of attempts set by the configuration register Hail Lifetime, the hail fails and an interrupt is sent to the OBC about the failure. All the timing in the hail sequence is done using the Wait Timer.

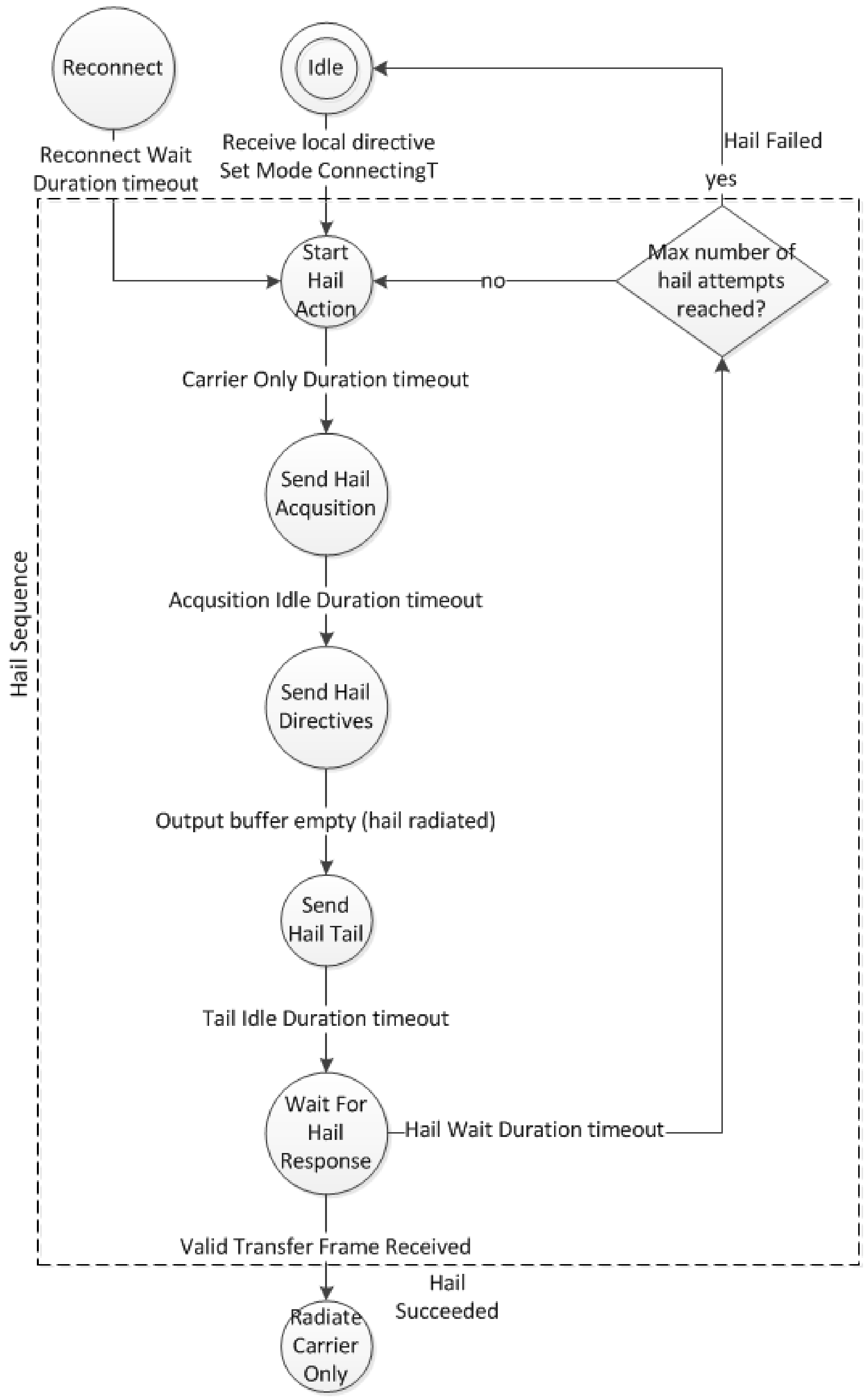
Reconnect
Idle
Reconnect Wait Duration timeout
Receive local directive Set Mode ConnectingT
Hail Failed
yes
Start Hail Action
no
Max number of hail attempts reached?
Carrier Only Duration timeout
Send Hail Acqusition
Acqusition Idle Duration timeout
Hail Sequence
Send Hail Directives
Output buffer empty (hail radiated)
Send Hail Tail
Tail Idle Duration timeout
Wait For Hail Response
Hail Wait Duration timeout
Valid Transfer Frame Received
Hail Succeeded
Radiate Carrier Only

Figure A.1: Hail Sequence states

A.2 Local Comm Change Directive and Related State Transitions

The state diagram in Figure A.2 shows the states related to a local Comm Change directive. The states are described as parameters in the Proximity-1 standard [4], but in our VHDL design it was more clear and easy to design them as separate states of the MAC state machine. Moreover, the handshaking, Waiting States (not shown in the state diagram) exist here as well. When the local Comm Change directive is received, the MAC waits for previous frames to be sent. It then tells the Directive Generator to construct and send a Comm Change directive (i. Set Transmitter Parameters and Set Receiver Parameters directive) using the values in the transmitter and receiver configuration registers. The MAC loops these actions (CommChange Y1, CommChange Y2 and CommChange Y3) until the Physical layer Symbol Inlock Status signal become false. This indicates that the remote transceiver has received the Comm Change and is changing its configuration parameters. The MAC waits in state CommChg Z1 for a valid frame to be received on the new communication channel. If a frame is received during the time given in the MIB Parameter register "Comm Change Waiting Period", the Comm Change succeeds. Otherwise, the Comm Change fails.

When a local Comm Change is performed, the value in the MIB Parameter register "Comm Change LifeTime" determines how many times the sequence containing the three first Comm Change states are performed (Y1, Y2 and Y3). Another parameter, "Comm Change Waiting Period", determines how long to wait in the state CommChange Z1 for a valid frame. If the Comm Change Lifetime expires in the first three states, the MAC goes back to the Data Service state. If Comm Change Waiting Period expiries, the MAC goes to idle state and resets. The reason for this is that no transmitter or receiver parameters are changed in the first three comm change states, hence the MAC can go back to the Data service state. However when the MAC goes to Z1 the transmission parameters are changed and the MAC can not go back to the old communication values.

All the timing is done using the Wait Timer in the local Comm Change states.

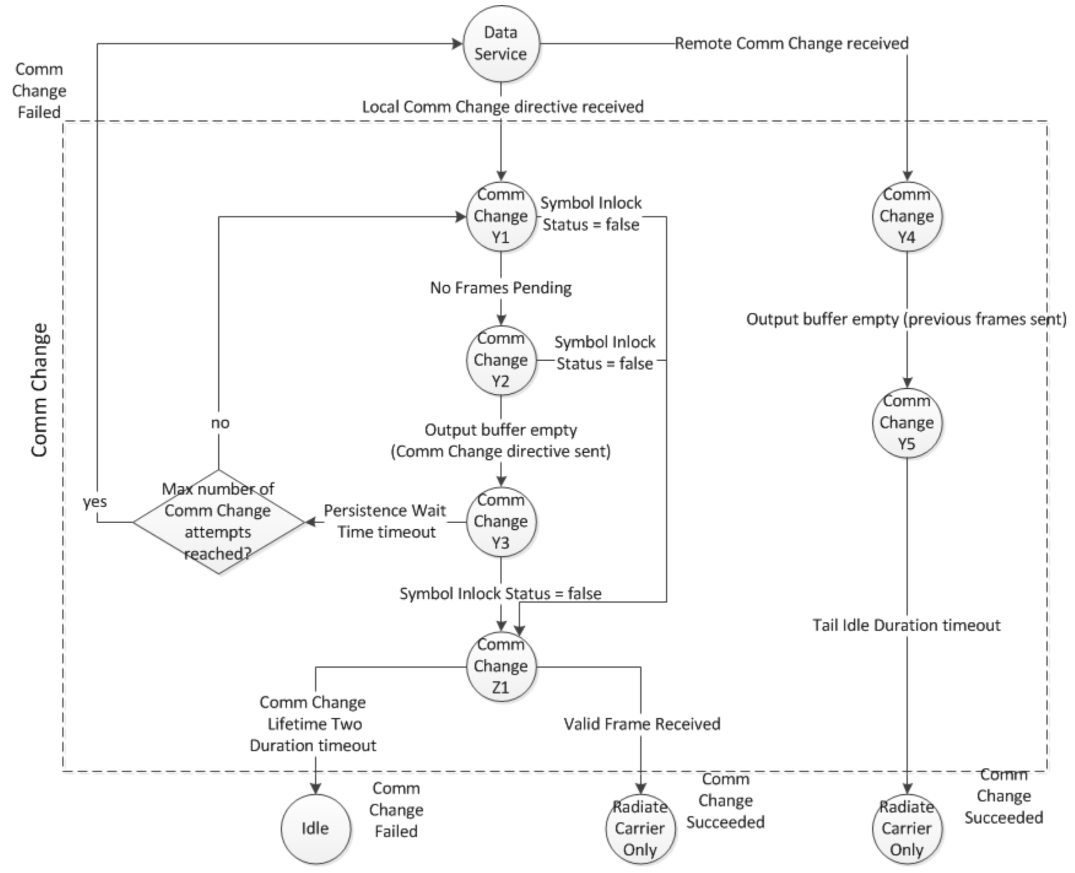
Data Service
Remote Comm Change received
Comm Change Failed
Local Comm Change directive received
Comm Change
Comm Change Y1
Symbol Inlock Status = false
No Frames Pending
Comm Change Y2
Symbol Inlock Status = false
no
Output buffer empty (Comm Change directive sent)
yes
Max number of Comm Change attempts reached?
Persistence Wait Time timeout
Comm Change Y3
Symbol Inlock Status = false
Comm Change Z1
Comm Change Lifetime Two Duration timeout
Valid Frame Received
Comm Change Y4
Output buffer empty (previous frames sent)
Comm Change Y5
Tail Idle Duration timeout
Idle
Comm Change Failed
Radiate Carrier Only
Comm Change Succeeded
Radiate Carrier Only
Comm Change Succeeded

Figure A.2: Comm Change states

A.3 Remote Comm Change Directive and Related State Transitions

The state diagram in Figure A.2 shows the states related to a remote Comm Change directive. The states are described as parameters in the Proximity-1 standard [4], but like the local Comm Change states they are separate states of the MAC state machine in our implementation. Again, Waiting states have been added where needed in order to support the handshaking communication between blocks. The Comm Change is initialized by the MAC receiving a Comm Change directive (i. Set Transmitter Parameters and Set Receiver Parameters directive). The MAC takes the received receiver and transmitter configurations and places them into the registers holding the current receiver and transmitter configurations. It then sends a Comm Change response (PLCW) using the new communication configurations and goes to the Radiate Carrier Only state.

A.4 Registers

The tables in this section list all the registers that can be accessed through the register interface of the Proximity-1 module. The configuration registers table contains a column which indicates whether the register is required (mandatory) when setting up a communication session. Some of the configuration registers need to be static during a communication session (indicated with (static) in the table). This means that these configuration registers must only be changed when the module is in the Idle state, before a communication session is started. The dynamic configuration registers can be changed during a communication session (indicated with (dynamic) in the table). Since most of the registers in this section are parameters defined in the Proximity-1 standard, more information can be found in [4].

Configuration Registers

Register	**Session requirement**	**Explanation**
Acquisition Idle Duration	Mandatory (static)	Time that the initial idle sequence is radiated
Carrier Loss Timer Duration	Mandatory (static)	Time that session continues if the carrier is lost
Carrier Only Duration	Mandatory (static)	Time that an unmodulated carrier is radiated
Comm Change Lifetime	Mandatory (static)	Number of times the initiator of the Comm Change should repeat the first phase of the Comm Change activity
Comm Change Waiting Period	Mandatory (static)	Time the initiator of the Comm Change should wait in the second phase of Comm Change
Hail Lifetime	Mandatory (static)	Number of times the caller should repeat the hailing activity
Hail Wait Duration	Mandatory (static)	Time that the caller should wait for a hail respons
Interval Clock	Mandatory (static)	Number used to divide the Bus Clock to generate a clock for timers
Local Spacecraft ID	Mandatory (static)	Local Spacecraft Identifier
Local PCID	Mandatory (static)	Local receivers physical channel ID
Local Resynchronization	Mandatory (static)	Indicates if the OBC or the Proximity-1 module is responsible for Resync activities
Source Destination	Mandatory (static)	Source SCID or Destination SCID

Test Source	Mandatory (static)	Indicates if a test should be performed on the received frames SCID
Acknowledge several frames	Mandatory (static)	Indicates if one PLCW is sent per received Sequence Controlled frame or if one PLCW is sent every Window Size-1 frames.
Window Size	Mandatory (static)	Maximum COP-P transmission window size
PLCW Repeat Interval	Mandatory (static)	Maximum transmission time between successive PLCWs
Drop Carrier Duration	Mandatory (static)	Time that the caller drops the carrier in the Reconnect state
Resync Lifetime	Mandatory (static)	Number of times that the resynchronisation activity is repeated
Resync Waiting Period	Mandatory (static)	Time spent waiting for Resync respons
Sync Timeout	Mandatory (static)	Value that the Sync Timer is initialized to
Tail Idle Duration	Mandatory (static)	Time to radiate the idle sequence at the end of a session
Hail Transmitter Parameters	Mandatory (static)	Transmitter configurations used during hail
Hail Receiver Parameters	Mandatory (static)	Receiver configurations used during hail
Maximum Frame Length	Mandatory (dynamic)	Maximum frame length
Remote PCID	Mandatory (dynamic)	Identifies remote physical channel
Remote Spacecraft ID	Mandatory (dynamic)	Remote Spacecraft Identifier
Dynamic Time Persistence MIB Parameter	Mandatory (dynamic)	Time that the initiator of the Comm Change should wait for that the responder of the Comm Change drops the carrier

Receiving SCID Buffer	Optional (dynamic)	Set value of receiving SCID Buffer
Set Transmitter Parameters	Mandatory (dynamic)	Transmitter configurations sent during hail and Comm Change
Set Receiver Parameters	Mandatory (dynamic)	Receiver configurations sent during hail and Comm Change
Set PL Extensions Transmitter	Mandatory (dynamic)	Transmitter PL Extension configurations sent during hail and Comm Change
Set PL Extensions Receiver	Mandatory (dynamic)	Receiver PL Extension configurations sent during hail and Comm Change
Port Id to Route Address Map 0-3	Mandatory (dynamic)	Router Bytes
Port Id to Route Address Map 4-7	Mandatory (dynamic)	Router Bytes

Directive Registers

Register	**Function**
Set Mode	Initiate session/Terminate Session
Local Comm Change	Change transceiver configurations during an active communication session.
Local No More Data	Tells the Proximity-1 module that the OBC has no more data to send in the communication session.
Local Time Sampling	Initiate Time Sampling (unused register)
Set Duplex	Set Duplex (unused register)

Status Registers

Register	Status
MAC state	The current operational state of the Proximity-1 module
Current Transmitter Parameters	Current configuration of the transmitter
Current Receiver Parameters	Current configuration of the receiver
Current PL Extensions Transmitter	Current additional configurations of the transmitter
Current PL Extensions Receiver	Current additional configurations of the receiver
Local Directive Busy	Indicates whether the Proximity-1 module is ready to receive a new directive or if it is busy handling a previously written directive.
Time Received Value	Unused
Time Transmitted Value	Unused
Block Status	Contains the value of several internal Proximity-1 signals and registers

Interrupt Registers

Register	**Interrupt explanation**
Hail Success	Hail succeeded
Hail Failed	Hail failed
Session Terminated	Session Terminated
CommChange Success	CommChange activity completed successfully
CommChange Failed	CommChange activity failed
Resync Success	Resynchronisation activity succeeded
Resync Failed	Resynchronisation activity failed
COP-P Lost Sync	COP-P synchronisation lost (Sync Timer expires)
Timings Services Available	Timing Services time tags are available
Carrier Lost	Carrier loss timer expires
Invalid Frame Source	Invalid frame source received
Carrier Only	Only carrier received during hailing
Register Write Ignored	Register write ignored, other register write pending
Sequence Controlled Packet Acknowledged	A Sequence Controlled packet was acked by the remote transceiver.
Expedited Packet Radiated	An Expedited packet was transmitted.
Time Sample Receive	Time sample for received frame is available (unused interrupt)
Time Sample Transmit	Time sample for transmitted frame is available (unused register)