Simplex Control and Message/ACK Exchange
ECE4305 Final Project Module 6

by

Renato Iida, Le Wang, Rebecca Copper

A Project Report
Submitted to Prof. Wyglinski
in partial fulfillment of the requirements of
ECE 4305
in
Electrical and Computer Engineering

Abstract

Put your abstract here.

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This is the introcution to my thesis or dissertation

Chapter 2

Background

Here is some background you'll need to know about my research.

2.1 Background subsection

Just a small section for part of the background of my research. [**?**]

Bibliography

[1] The Author, *Awesome book*, WPI Press, 2008.

Listing 2.1: FrameObj Source Code

```
classdef FrameObj
    %FRAMEOBJ has 2 input configuraions; 4 inputs mean you are using the
    %frame requirements to create a FrameObj, 1 input means you are using
    %the bits
    %   In the first configuration (4 inputs) the first 3 inputs must be
    %   numbers and it is recommended that the constant properties of
    %   FrameObj are used to ensure accuracy. The last input depends on the
    %   frameType.
    %    -DATAFRAME: the last input must be a string and only the first 234
    %       characters will be included.
    %    -ACKFRAME: the last input does not matter but must exist.
    %    -POLLFRAME
    %    -REQFRAME
    %    -TABLEFRAME
    %   In the second configuration (1 input) the input must be a binary
    %   nx1 array with a supported frame type in the first byte. To ensure
    %   that there is no indexing outside the dimensions of the input array
    %   it is best if all inputs have a size of 1920x1


    properties (Constant)
        %for sndID and rcvID
```

```
        IDBS1 = 100;
        IDUE1 = 101;
        IDUE2 = 102;
        IDUE3 = 203;
        IDBS2 = 200;
        IDALLUE = 000;

        % for frameTypes    %these numbers are chosen to resist flips and
        % shifts. It is the most important that frameType is correct as
        % very wrong frames can be dropped based on frameType
        DATAFRAME = 240;    %1111 0000
        ACKFRAME  = 255;    %1111 1111
        POLLFRAME = 202;    %1100 1010
        REQFRAME  =  83;    %0101 0011
        TABLEFRAME = 15;    %0000 1111
        INVALID = 0;        %0000 0000

        % for classUse
        ENCODE = 1;
        DECODE = 2;

        CHUE1BS1 = 1;
        CHUE2BS1 = 2;
        CHBS1BS2 = 3;
        CHUE3BS2 = 4;

        CRCOK = 1;
        CRCFAIL = 2;

        ACKRECEIVED = 3;
        TIMEOUT = 4;

        MAXDATA = 234; % Probably the only constant that has meaning
        MAXBYTES = 240;
    end

    properties
```

```
    classUse    %Identifies which configuration we are in
    frameType   %Identifies the type of frame that is being used
    rcvID       %The identification number of the destination receiver
    sndID       %The identification number of the sender.
    data        %The data field (cuts off after more than 234 bytes)

end
properties (Dependent)
    dataSize    %Indicates the length of the payload in bytes.
    header      %The array of the frame header with hCRC8
    hCRC8       %CRC-8 code verfication of the header field
    dCRC8       %CRC-8 code verfication of the data field
    frameArray  %The frame as an n*1 array
end

methods
    %This function sets the inputs to the property functions below.
    %Those functions define the actual properties, this is where we
    %call those functions.
    function obj = FrameObj(inputType,inputrcvID,inputsndID,inputData)

        % create a FrameObj with 4 inputs from the frame requirements
        if nargin == 4
            obj.classUse = inputType;
            %test if the frame type is valid
            obj.frameType = inputType;
            obj.rcvID = inputrcvID;
            obj.sndID = inputsndID;
            obj.data = inputData;

        %create a FrameObj with 1 array of bits
        elseif nargin == 1
            %Not sure if we need to do this. It might just be a
            %reminder that inputType is not the data that is being used
            %by the class properties in this case.
            bitwise = inputType;
```

```
        %size check
        [size_in, ¬] = size(bitwise);
        if (size_in ≥ 40)
            % hCRC check

            % needed for the crc calculation
            hDetect = comm.CRCDetector([8 7 6 4 2 0]);
            % detects if there is an error in the CRC of the header
            [¬, err] = step(hDetect, bitwise(1:5*8,1));

            if (err ==0)
                obj.classUse = bitwise;

                %We are actually converting the array of bits here
                %and then passing the pretty decimal numbers we get
                %to the propertyfunctions.
                obj.frameType = bi2de(bitwise(1:8,1)','left-msb');
                obj.rcvID = bi2de(bitwise(1+8:2*8,1)','left-msb');
                obj.sndID =bi2de(bitwise(1+2*8:3*8,1)','left-msb');

                %Whether there is data or not depends on frameType.
                %The location of data in the frame is dependent on
                %dataSize so we pass the unaltered FrameObj input
                %to obj.data
                obj.data = bitwise;
            else
                % the crc does not match and the header is junk
                obj.frameType = FrameObj.INVALID;
            end
        else
            % the data we recdived is not long enough to check the
            % header crc
            obj.frameType = FrameObj.INVALID ;
        end
    else %incorrect number of inputs
        error('That is not a valid number of inputs')
    end
```

```
        end

%classUse
        %This property enables us to distingush between the two uses of the
        %class FrameObj though the length of the first (or only) input.
        %This is only used by obj.data which takes both string type inputs
        %and arrays of bits and needs to be able to distinguish them
        function obj = set.classUse(obj,inputframeType)
            [array_or_vector,¬] = size(inputframeType);
            if array_or_vector == 1
                obj.classUse = FrameObj.ENCODE;
            else
                obj.classUse = FrameObj.DECODE;
            end
        end

%frameType
        function obj = set.frameType(obj,inputframeType)
            %Using the switch statement in this way ensures that a
            %supported data type is used
            switch inputframeType
                case FrameObj.DATAFRAME %DATA
                    obj.frameType = uint8(inputframeType);
                case FrameObj.ACKFRAME  %ACK
                    obj.frameType = uint8(inputframeType);
                case FrameObj.POLLFRAME %POLL
                    obj.frameType = uint8(inputframeType);
                case FrameObj.REQFRAME  %REQ
                    obj.frameType = uint8(inputframeType);
                case FrameObj.INVALID   %INVALID
                    obj.frameType = uint8(inputframeType);
                otherwise % also INVALID
                    obj.frameType = uint8(FrameObj.INVALID);
            end
        end

%rcvID
```

```
        function obj = set.rcvID(obj,inputrcvID)
            obj.rcvID = uint8(inputrcvID);
        end

%sndID
        function obj = set.sndID(obj,inputsndID)
            obj.sndID = uint8(inputsndID);
        end

%data
        %frameType dependent
        %classUse dependent
        %Data actually refers to the data and the CRC8 number in an n*1
        %binary array
        function obj = set.data(obj,datainput)
            %These variables mean we can vary the size of MAXBYTES or the
            %header without and data will still be functional.
            header_bits = (FrameObj.MAXBYTES-(FrameObj.MAXDATA+1))*8;
            max_data_bits = FrameObj.MAXDATA*8;
            switch obj.frameType
                case FrameObj.DATAFRAME %DATA
                    if obj.classUse == FrameObj.ENCODE;
                        %This converts the datainput into an array of bits
                        temp_bin = reshape(dec2bin(datainput,8)',1,[]);

                        %Define the length of temp_data for speed
                        % the length of temp_data is limited by MAXBYTES
                        if size(temp_bin,2)≥max_data_bits
                            temp_data = zeros(1,max_data_bits);
                        else
                            temp_data = zeros(1,size(temp_bin,2));
                        end

                        for j=1:size(temp_data,2)
                            temp_data(1,j) = str2num(temp_bin(1,j));
                        end
```

```
            crcGen = comm.CRCGenerator([8 7 6 4 2 0]);

            %Calculates the CRC and adds it to the end of data
            obj.data =  step(crcGen, temp_data');

        elseif obj.classUse == FrameObj.DECODE;
            data_bits = size(datainput, 1)-header_bits -8;
            %This seperates the data from the rest of the array
            %using dataSize.

            %First seperate dataSize then convert to decimal
            %Cast to double and convert from bytes to bits
            Temp =  bi2de(datainput(1+3*8:4*8,1)','left-msb');
            ds = double(Temp*8);

            %This allows FrameObj to not exceed the dimensions
            %of inputdata in case the dataSize was corrupted to
            %be or larger than the length of the input array
            %and passed the hCRC.
            if ds ≥ data_bits
                ds = data_bits
            end
            % or larger than MAXDATA
            if ds ≥ max_data_bits
                ds = max_data_bits
            end

            %Seperate data using the start of the data and the
            %length+crc
            bits = datainput(header_bits+1:header_bits+ds+8,1);
            %cast to double
            obj.data = double(bits);
        end
    case FrameObj.ACKFRAME %ACK
        obj.data = ''; %could be anything
        %if there is no data you should not try to access tha
        %ACK data but data is always assesed regardless of the
```

```
                %frameType
            case FrameObj.POLLFRAME %POLL
                obj.data = bi2de(datainput);%work on this
                    %is there anything we don't want converted here?
                    %unsure
            case FrameObj.REQFRAME %REQ
                obj.data = bi2de(); %work on this.
            %Do we need this?  Currently no data to be passed besides
            %recID and sendID.
            case FrameObj.INVALID   %INVALID
                obj.data = ''; %could be anything
                %if there is no valid frameType you should not try to
                %access tha data but data is always assesed regardless
                %of the frameType
            otherwise
                error('Not a supported frame type for data')
                % If this error occurs while using a legitimate frame
                % type please add an addiional case statement for that
                % frame type.

                % if there is no data for this frame type copy ACKFRAME
                % if there is string data copy DATAFRAME
                % A diferent type of data may require a different case
        end
    end

%dataSize
    % frameType dependent
    % returns 0 if ACK
    function value = get.dataSize(obj)
        switch obj.frameType
            case FrameObj.DATAFRAME %DATA
                %Convert from bits to bytes and subtract 1 to account
                %for the CRC
                value = (length(obj.data)/8)-1;
            case FrameObj.ACKFRAME  %ACK
                value = 0;
```

```
                    %the ACKFRAME has no data but it must have a dataSize
                otherwise
                    error('Not a supported frame type for dataSize')
                    % If this error occurs while using a legitimate frame
                    % type please add an addiional case statement for that
                    % frame type.

                    % If there is no data for this frame type copy ACKFRAME
                    % If there is data copy DATAFRAME
                    % A diferent type of data may require a different case
            end
        end

%dCRC8
        %frameType dependent
        function value = get.dCRC8(obj)
            switch obj.frameType
                case FrameObj.DATAFRAME %DATA
                    %The last byte of obj.data is the CRC. It is seperated
                    %from the data here
                    [m, ¬] = size(obj.data);
                    value =zeros(8,1);      %Define value for speed
                    for j=1:8
                        value(j,1) = obj.data(m-8+j,1);
                    end
                case FrameObj.ACKFRAME %ACK
                    error('This is an ACK, it has no data therefor no CRC')
                    %If there is no data there should be no check
                otherwise
                    error('Not a supported frame type for CRC8')
                    % If this error occurs while using a legitimate frame
                    % type please add an addiional case statement for that
                    % frame type.

                    % If there is no data for this frame type copy ACKFRAME
                    % If there is data copy DATAFRAME
            end
```

```
        end

%header
        %all frametypes have the same length and components
        %frametype, sndID, rcvID, dataSize, hCRC8
        function value = get.header(obj)
            % Each part of the header is converted into binary arrays
            type_array  = de2bi(obj.frameType,8,'left-msb');
            rcvid_array = de2bi(obj.rcvID,8,'left-msb');
            sndid_array = de2bi(obj.sndID,8,'left-msb');
            size_array  = de2bi(obj.dataSize,8,'left-msb');

            temp_header = [type_array rcvid_array sndid_array size_array];

            crcGen = comm.CRCGenerator([8 7 6 4 2 0]);
            %Calculates the CRC and adds it to the end of header
            value =  step(crcGen, logical(temp_header'));
        end

%hCRC8
        function value = get.hCRC8(obj)
            % The last byte of obj.header is the hCRC
            [m, ¬] = size(obj.header);
            value =zeros(8,1);      %Define value for speed
            for j=1:8
                value(j,1) = obj.header(m-8+j,1);
            end
        end

%frameArray
        %frameType dependent
        %ACKFRAME --> frametype, sndID, rcvID, dataSize, hCRC8
        %DATAFRAME--> frametype, sndID, rcvID, dataSize, hCRC8, data, dCRC8
        % If we want to fill the end with zeros that could easily be done
        function value = get.frameArray(obj)
            % for data we combine the data and header in a n*1 binary array
            % the ack is only a header.
```

```
            % both header and data have a crc8 included

            switch obj.frameType
                case obj.DATAFRAME
                    value = [obj.header; obj.data];
                case obj.ACKFRAME
                    value = [obj.header];
                otherwise
                    error('Not a supported frame type for frameArray')
                    % If this error occurs while using a legitimate frame
                    % type please add an addiional case statement for that
                    % frame type.

                    % If there is no data for this frame type copy ACKFRAME
                    % If there is data copy DATAFRAME
                    %A diferent type of frame may require a different case.
            end
        end
    end
end
```