TAN-062

Application Note

Processing HDLC Messages

XRT86VL3x DS-1/E1 Framer + LIU Combo

TABLE OF CONTENTS

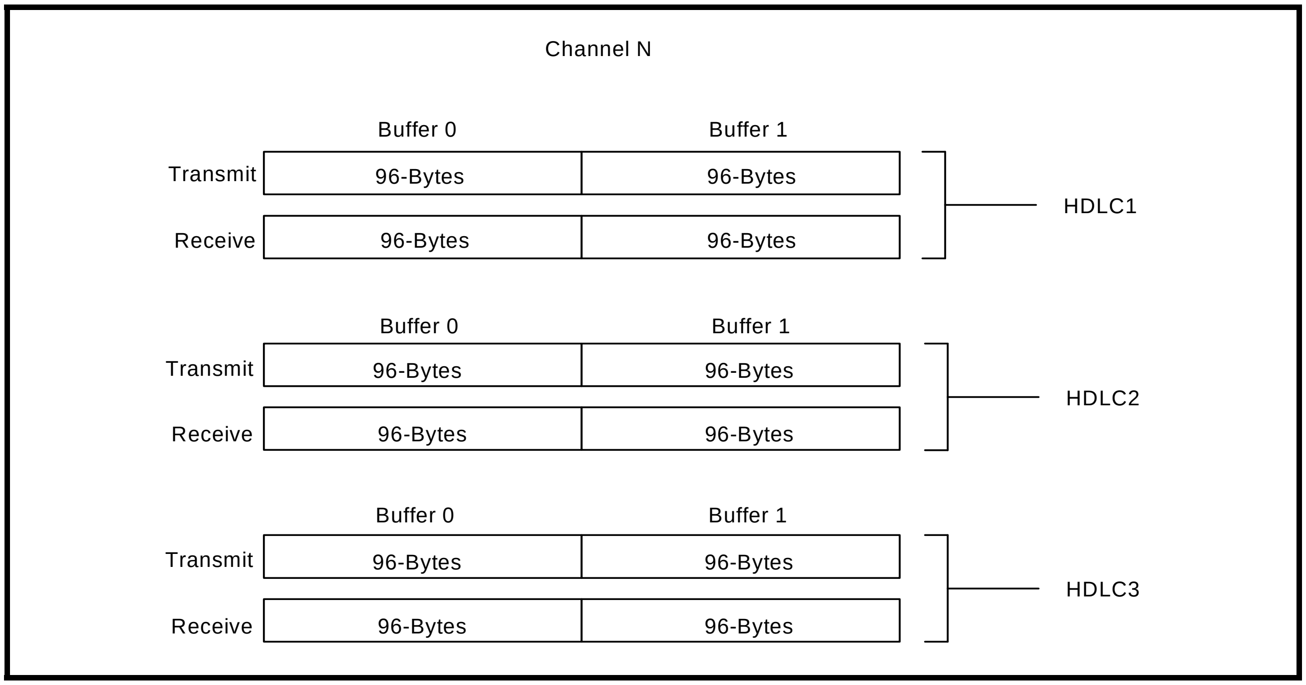
1.0 INTRODUCTION

This document describes the process for transmitting and receiving HDLC messages that can be used as a reference. The main three topics are Message Oriented Signaling (MOS), Bit Oriented Signaling (BOS), and SLC-96. It is important to note that when transmitting or receiving a message throught the FDL bits, HDLC Controller 1 must be used. Also, this document assumes HDLC Controller 1 for all examples. For HDLC Controller 2 or HDLC Controller 3, see the full datasheet for more information.

1.1 HDLC Controller

This document describes the process for transmitting and receiving HDLC messages that can be used as a reference. Figure 1. is a simplified block diagram of the HDLC Controller Block. For each channel, there are three independent HDLC Controllers. Each controller has two 96-byte buffers for the Transmit path and two 96-byte buffers for the Receive path. Therefore, there is total of twelve buffers per channel.

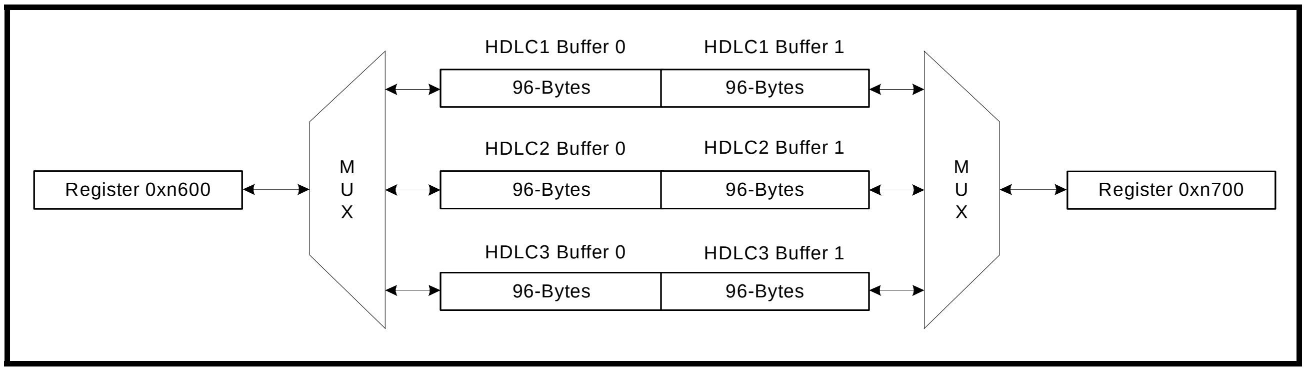
FIGURE 1. SIMPLIFIED BLOCK DIAGRAM OF THE HDLC CONTROLLER BLOCK



1.2 Multiplexed Access of the HDLC Buffers

Each buffer has its own dedicated internal holding register. To access the contents of these buffers or to store messages, the XRT86VL3x utilizes a multiplexed access as shown in Figure 2. The buffers are broken down into Buffer 0 or Buffer 1. Register 0xn600 is used to Read or Write to Buffer 0 of the selected HDLC controller, while register 0xn700 is used to Read or Write to Buffer 1 of the selected HDLC controller. Although, each HDLC controller has its own dedicated register set for control, the message contents of only one controller can be accessed at one time through the microprocessor or DMA interface.

FIGURE 2. MULTIPLEXED ACCESS OF THE HDLC MESSAGE CONTENTS



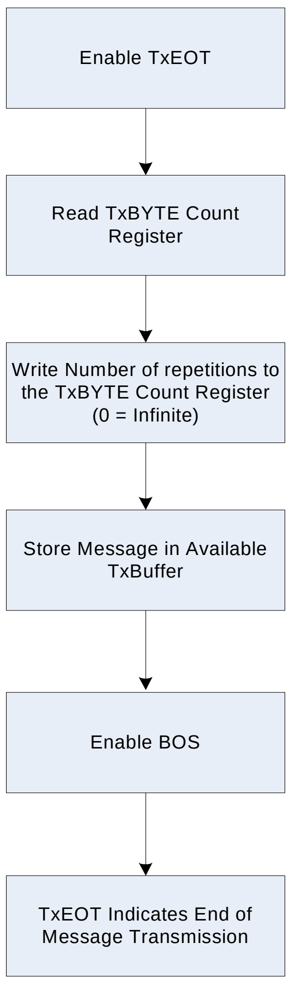
2.0 BIT ORIENTED SIGNALING

Although, the XRT86VL3x allows the user to send and receive BOS through designated D/E time slots, it’s most commonly transmitted through the Facility Data Link (FDL) bits. This section describes the process of transmitting and receiving BOS through FDL.

2.1 Transmitting BOS Messages

By default, the chip is configured to the FDL for message transmission. The flow chart below shows the process of transmitting a BOS message. The first step is to enable the TxEOT interrupt that will be used alert the system when transmission is complete. Next, read the TxBYTE count register to determine which buffer is available. Write the number of repetitions the BOS is to be transmitted into the TxBYTE count register. Store the message to be sent in the available buffer, and then enable the BOS message (write 0x00 to register 0xn113).

FIGURE 3. FLOW CHART FOR TRANSMITTING BOS MESSAGES


Enable TxEOT
Read TxBYTE Count Register
Write Number of repetitions to the TxBYTE Count Register (0 = Infinite)
Store Message in Available TxBuffer
Enable BOS
TxEOT Indicates End of Message Transmission

2.2 *Receiving BOS Messages*

By default, the chip is configured to the FDL for message transmission. The flow chart below shows the process of receiving a BOS message. The first step is to enable the RxEOT and the RxIDLE interrupts. While servicing the interrupts, read bit 7 in register 0xnB06. If this bit is set to "0", then it's a BOS type message. The BOS type is searching the FDL for 0xFF followed by one byte with its LSB and MSB set to "0". If BOS, read the RxBYTE count register to determine which buffer is available and the number of repetitions. Read the available buffer one time to retrieve the message.

NOTE: *To discard a message with the same value, use the AutoRx feature in register 0xn113.*

FIGURE 4. FLOW CHART FOR RECEIVING BOS MESSAGES

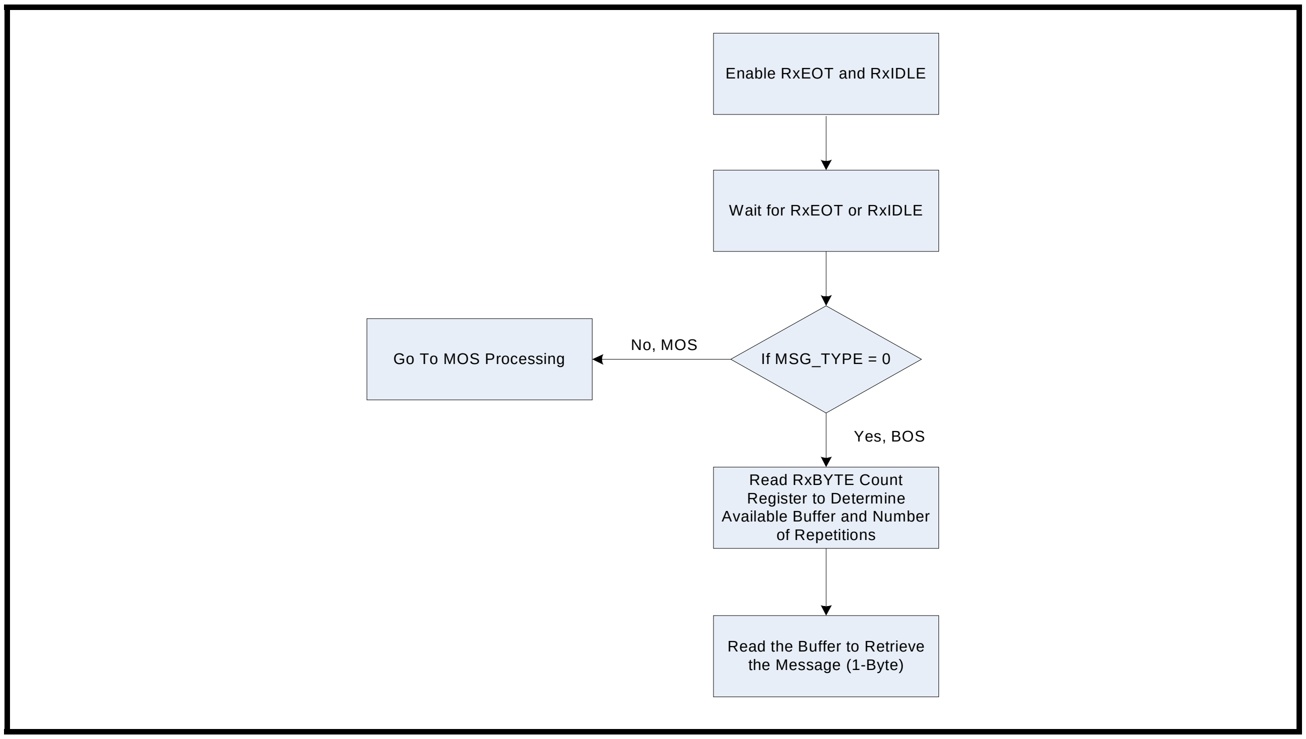
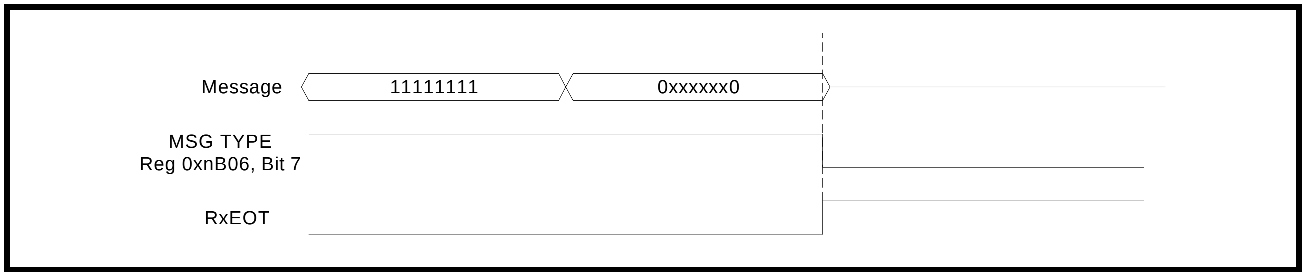

Enable RxEOT and RxIDLE
Wait for RxEOT or RxIDLE
If MSG_TYPE = 0
No, MOS
Go To MOS Processing
Yes, BOS
Read RxBYTE Count Register to Determine Available Buffer and Number of Repetitions
Read the Buffer to Retrieve the Message (1-Byte)

FIGURE 5. MESSAGE TYPE AND RXEOT DURING A BOS RECEPTION



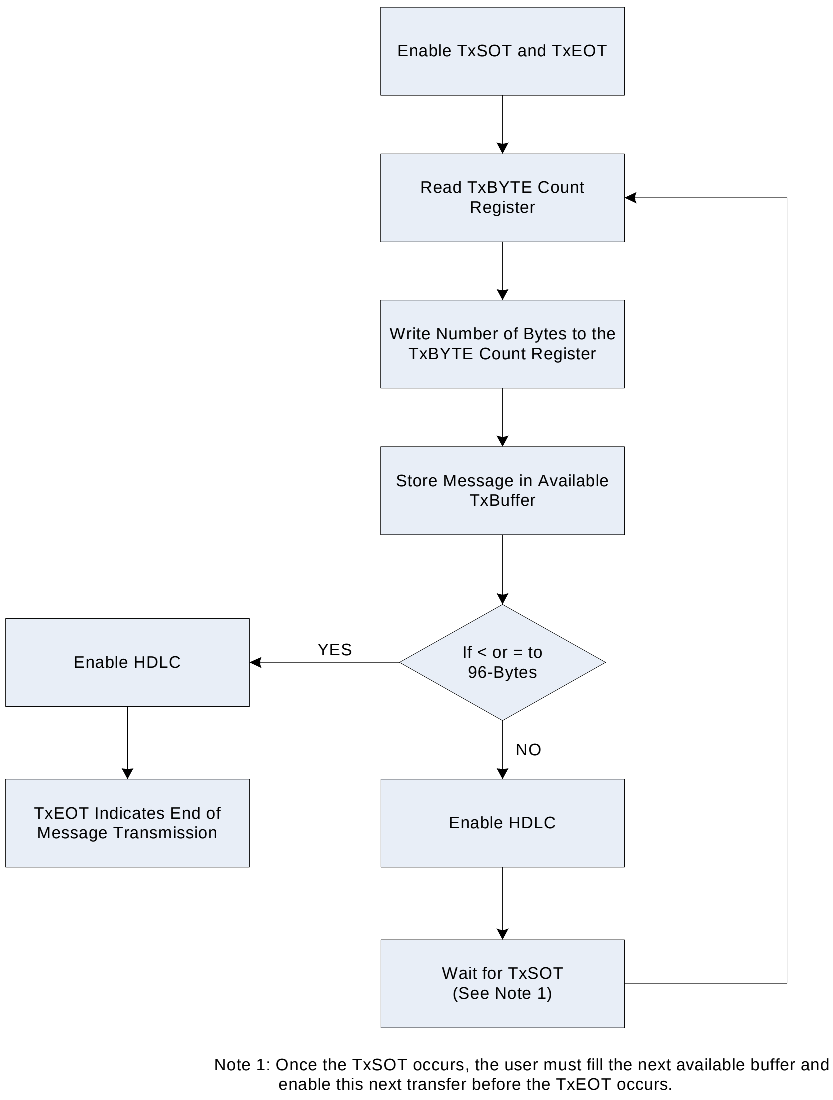
3.0 MESSAGE ORIENTED SIGNALING

Although, the XRT86VL3x allows the user to send and receive MOS through designated D/E time slots, it's most commonly transmitted through the Facility Data Link (FDL) bits. This section describes the process of transmitting and receiving MOS through FDL.

3.1 Transmitting MOS Messages

By default, the chip is configured to the FDL for message transmission. The flow chart below shows the process of transmitting a MOS message. The differentiating feature of MOS is its ability to send large messages. When sending a message larger than 96-bytes, it is important to write the next available block of the message before the TxEOT occurs.

FIGURE 6. FLOW CHART FOR TRANSMITTING MOS MESSAGES


Enable TxSOT and TxEOT
Read TxBYTE Count Register
Write Number of Bytes to the TxBYTE Count Register
Store Message in Available TxBuffer
If < or = to 96-Bytes
YES
Enable HDLC
TxEOT Indicates End of Message Transmission
NO
Enable HDLC
Wait for TxSOT (See Note 1)
Note 1: Once the TxSOT occurs, the user must fill the next available buffer and enable this next transfer before the TxEOT occurs.

3.2 *Receiving MOS Messages*

By default, the chip is configured to the FDL for message transmission. The flow chart below shows the process of receiving a MOS message. To differentiate between the end of a received message and when one buffer is merely full, use the RxIDLE interrupt. If the RxEOT occurs (with our RxIDLE), this means that the first buffer is full and needs to be serviced. If the RxEOT occurs with the RxIDLE flag, then it is the end of message. For RxABORT and event timing, see **Figure 8**.

***NOTE:** To discard a message with the same values, use the AutoRx feature in register 0xn113.*

FIGURE 7. FLOW CHART FOR RECEIVING MOS MESSAGES

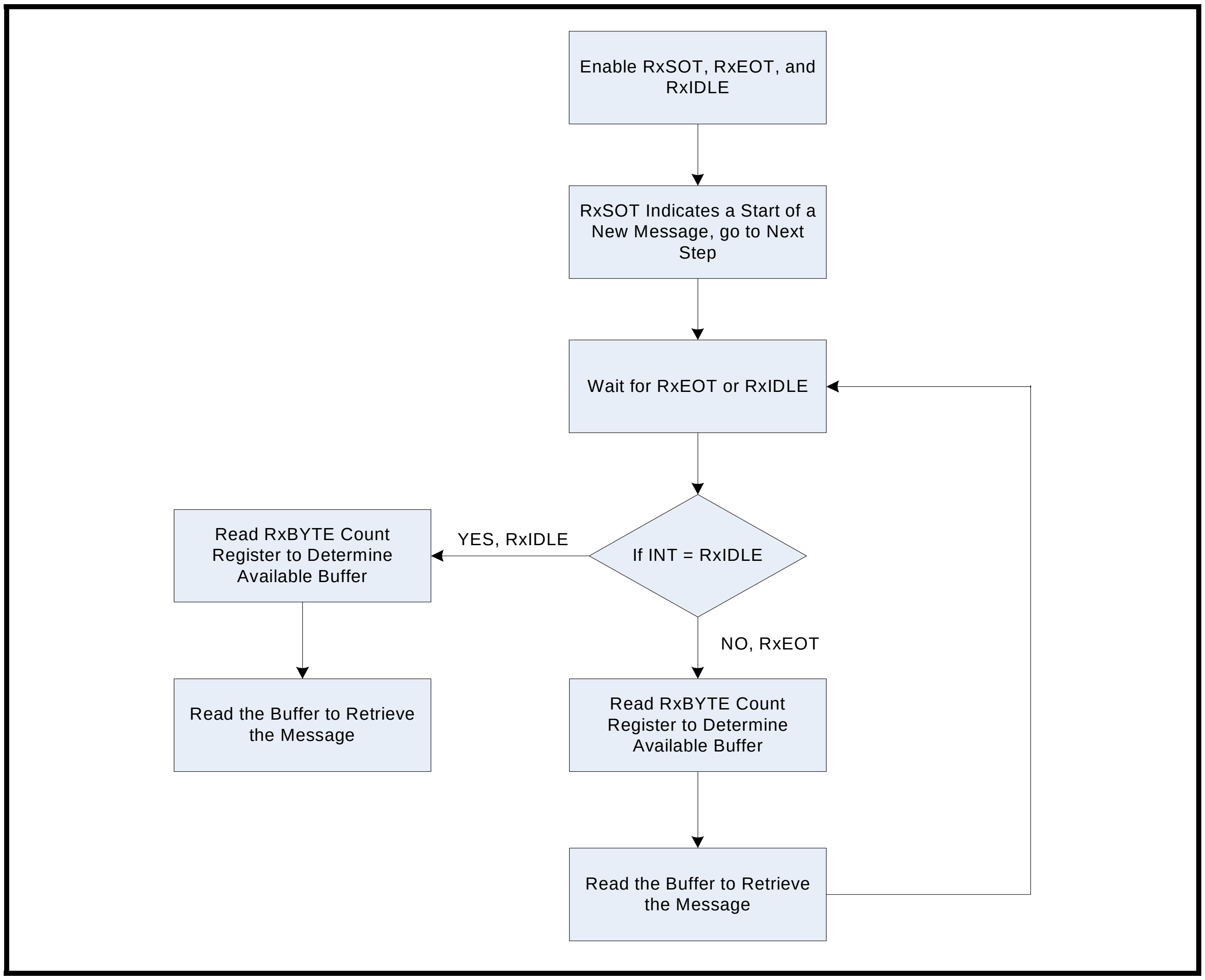

Enable RxSOT, RxEOT, and RxIDLE
RxSOT Indicates a Start of a New Message, go to Next Step
Wait for RxEOT or RxIDLE
If INT = RxIDLE
YES, RxIDLE
Read RxBYTE Count Register to Determine Available Buffer
Read the Buffer to Retrieve the Message
NO, RxEOT
Read RxBYTE Count Register to Determine Available Buffer
Read the Buffer to Retrieve the Message

FIGURE 8. HDLC EVENT TIMING FOR RXEOT, RXIDLE, AND RXABORT

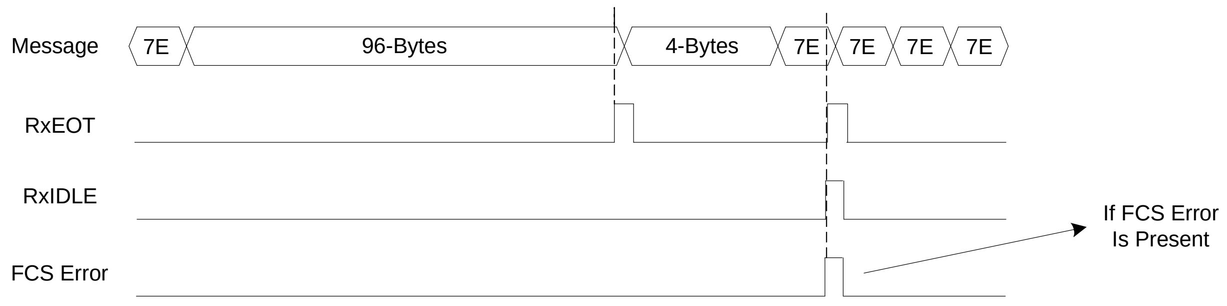


Figure a. Receiving a 100-Byte Message

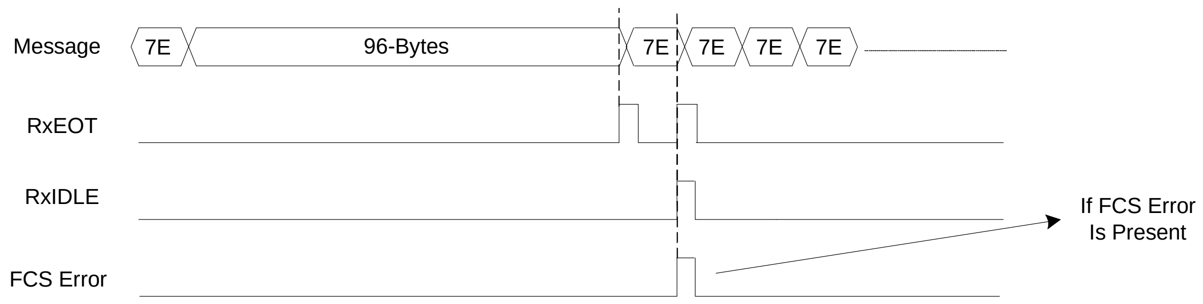


Figure b. Receiving a 96-Byte Message

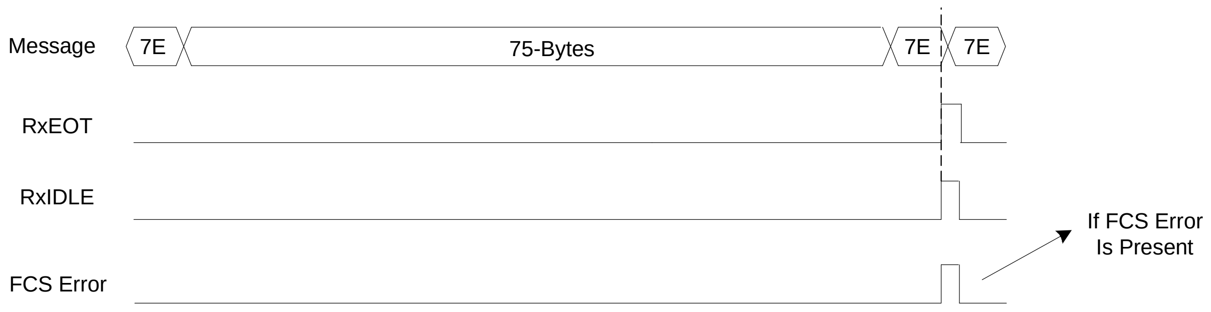


Figure c. Receiving a 75-Byte Message

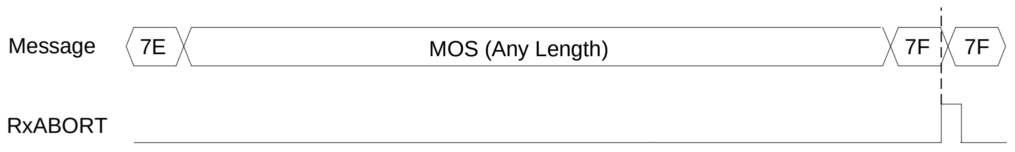


Figure d. Receiving an Abort Sequence (7 Consecutive 1's)

4.0 SLC-96 MESSAGE SIGNALING

Although, the XRT86VL3x allows the user to send and receive SLC-96 through designated D/E time slots, it's most commonly transmitted through the Facility Data Link (FDL) bits. This section describes the process of transmitting and receiving SLC-96 through FDL.

4.1 Transmitting SLC-96 Messages

By default, the chip is configured to the FDL for message transmission. The flow chart is the same as the process of transmitting a MOS message except for two distinguishing items. The first differentiating feature of SLC-96 is that it always contains 6-Bytes (with a pre-determined format). When sending a SLC-96 message, it is important to maintain the bytes order described below. This is NOT done automatically, the user must write the correct sequence inside the available buffer. The second differentiating item is that the user must specify SLC-96 while enabling the HDLC (register 0xn113, bit 7).

TABLE 1: SLC-96 MESSAGE 6-BYTES

BYTE	5	4	3	2	2	0
1	0	1	1	1	0	0
2	C1	1	1	1	0	0
3	C7	C6	C5	C4	C3	C2
4	1	0	C11	C10	C9	C8
5	A2	A1	M3	M2	M1	0
6	0	1	S4	S3	S2	S1

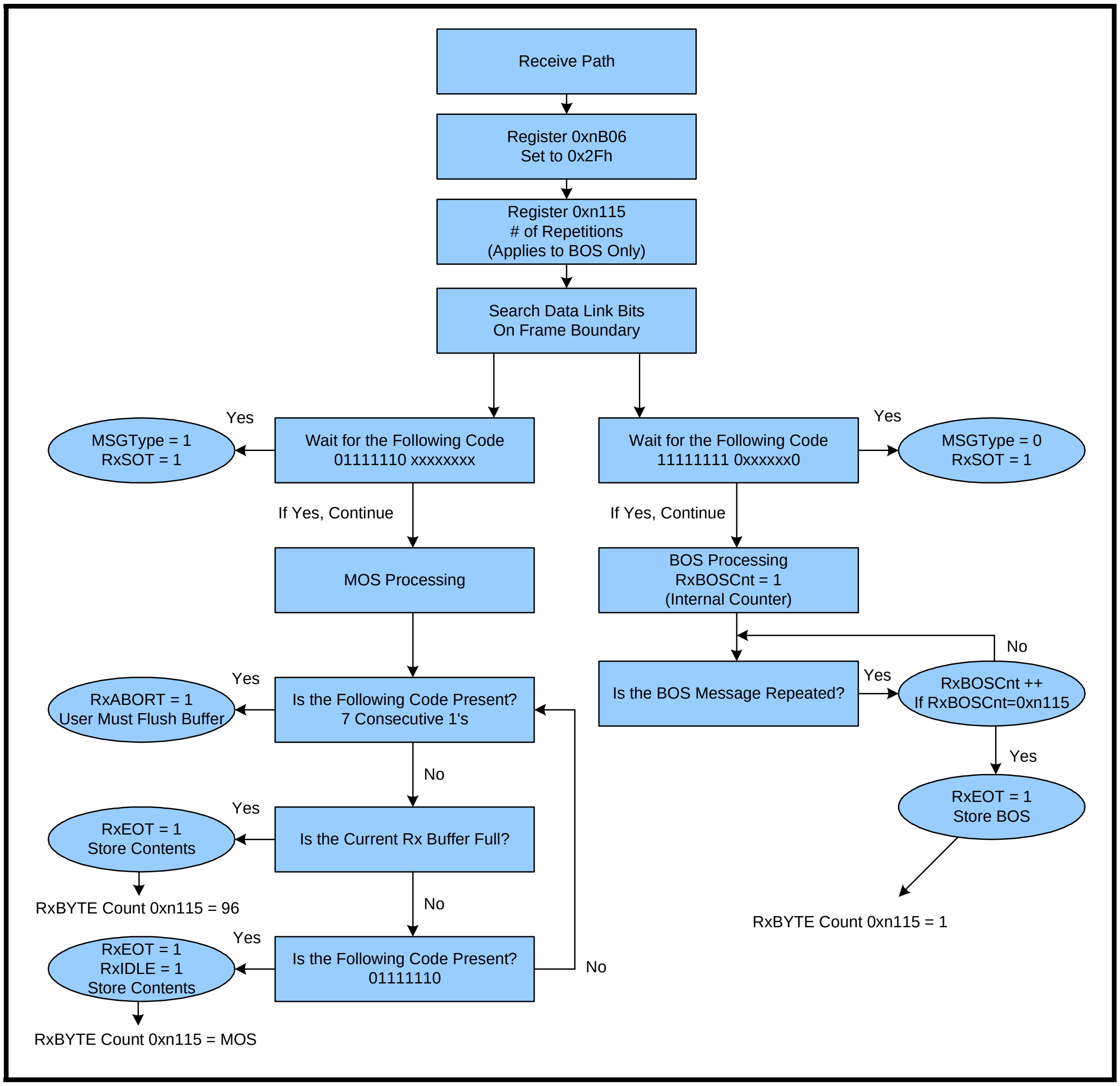
4.2 Receiving SLC-96 Messages

By default, the chip is configured to the FDL for message transmission. The flow chart is the same as the process of receiving a MOS message except that the message will only be 6-bytes long.

5.0 HOW THE XRT86VL3X DETECTS MOS OR BOS SIMULTANEOUSLY

The XRT86VL3x does not know which type of message is going to be received through the data link bits. It could be a MOS or BOS type. The following flow chart shows how the "Chip" searches for the incoming data link message.

FIGURE 9.


Receive Path
Register 0xnB06 Set to 0x2Fh
Register 0xn115 # of Repetitions (Applies to BOS Only)
Search Data Link Bits On Frame Boundary
Yes
MSGType = 1 RxSOT = 1
Wait for the Following Code 01111110 xxxxxxxx
Wait for the Following Code 11111111 0xxxxxx0
Yes
MSGType = 0 RxSOT = 1
If Yes, Continue
If Yes, Continue
MOS Processing
BOS Processing RxBOSCnt = 1 (Internal Counter)
No
Yes
RXABORT = 1 User Must Flush Buffer
Is the Following Code Present? 7 Consecutive 1's
Is the BOS Message Repeated?
Yes
RxBOSCnt ++ If RxBOSCnt=0xn115
No
Yes
Yes
RxEOT = 1 Store Contents
Is the Current Rx Buffer Full?
RxEOT = 1 Store BOS
RxBYTE Count 0xn115 = 96
No
RxBYTE Count 0xn115 = 1
Yes
RxEOT = 1 RxIDLE = 1 Store Contents
Is the Following Code Present? 01111110
No
RxBYTE Count 0xn115 = MOS