Implementation of a Packet Over SONET/SDH Transmission System using the SPECTRA-155.

Preliminary Issue 1

TABLE OF CONTENTS

1 OVERVIEW 1
1.1 INTRODUCTION 1
1.2 FRAME STRUCTURES FOR IMPLEMENTING POS 1
1.3 PACKET OVER SONET FRAME FORMAT 2
1.4 PAYLOAD SCRAMBLING 3
2 SYSTEM ARCHITECTURE FOR TRANSPORTING POS 4
2.1 POS MAPPED OVER A STS-3C OR STS-1 SONET/SDH SPE 4
3 SPECTRA-155 CONFIGURATION 7
4 IMPLEMENTATION DETAILS 10
4.1 SPECTRA-155 TO POS HDLC PROCESSOR INTERFACE 10
4.2 POS HDLC PROCESSOR TO LINK LAYER DEVICE INTERFACE 11
5 POS HDLC PROCESSOR DESIGN 12
5.1 FCS GENERATOR 13
5.2 BYTE STUFFING 14
5.3 BYTE DESTUFFING 25
5.4 FCS CHECK 25
6 FUNCTIONAL TIMING 33
7 DISCLAIMER 38
8 REFERENCES 39

LIST OF TABLES

TABLE 1 BYTE STUFFING CHARACTERS 14

TABLE 2 TRANSMIT FCSSEL CONFIGURATION 17

TABLE 3 FLAGSEL CONFIGURATION 17

TABLE 4 BYTE DESTUFFING CHARACTERS 25

TABLE 5 RECEIVE FCSSEL CONFIGURATION.. 27

LIST OF FIGURES

FIGURE 1: A SONET/SDH STS-1/STM-0 FRAME SHOWING THE TOH 1

FIGURE 2: A SONET/SDH STS-3C/STM-1 FRAME SHOWING THE TOH 2

FIGURE 3: POS HDLC FRAME FORMAT.. 2

FIGURE 4: POS OVER AN STS-3C/STM1 OR STS-1/STM-0 LINK 4

FIGURE 5: SPECTRA-155 TO POS HDLC PROCESSOR INTERFACE 10

FIGURE 6: POS HDLC PROCESSOR TO LINK LAYER DEVICE.................... 11

FIGURE 7: TRANSMIT SECTION.. 12

FIGURE 8: CRC-32 GENERATOR 14

FIGURE 9: TRANSMIT HDLC CONTROLLER 15

FIGURE 10: TRANSMIT HDLC CONTROLLER – DETAILED VIEW 18

FIGURE 11: TRANSMIT HDLC CONTROLLER STATE MACHINE 19

FIGURE 12: TRANSMIT FIFO 21

FIGURE 13: RECEIVE SECTION... 24

FIGURE 14: CRC-32 DECODER.. 26

FIGURE 15: RECEIVE HDLC CONTROLLER.. 26

FIGURE 16: RECEIVE HDLC CONTROLLER – DETAILED VIEW................... 28

FIGURE 17: RECEIVE HDLC CONTROLLER STATE MACHINE.................... 29

FIGURE 18: RECEIVE FIFO.. 30

FIGURE 19: TRANSMIT HDLC CONTROLLER – SPECTRA-155 TIMING...... 33

FIGURE 20: TRANSMIT HDLC CONTROLLER – FIFO INTERFACE TIMING. 34

FIGURE 21: TRANSMIT FIFO – LINK LAYER TIMING..................................... 35

FIGURE 22: RECEIVE HDLC CONTROLLER – SPECTRA-155 TIMING 35

FIGURE 23: RECEIVE HDLC CONTROLLER – FIFO TIMING 36

FIGURE 24: RECEIVE FIFO – LINK LAYER TIMING 37

1 OVERVIEW

1.1 Introduction

This application note addresses the need to transfer a point to point protocol (such as the Internet Protocol, IP, or any other packet based PPP) over a SONET/SDH payload envelope STS-3c/STM-1(AU3) or STS-1/STM-0 frame structures. This functionality is easily implemented, as described in this document, using PMC-Sierra's PM5342 SPECTRA-155 chip and a POS Processor.

Transportation of POS over the higher rate concatenated SONET/SDH payloads can be accomplished using the S/UNI-622. This is described in a separate application note described in PMC-Sierra document number PMC-960724.

1.2 Frame Structures for implementing POS

Figure 1 shows the STS-1/STM-0 mapping. Fixed stuff bytes exist at columns 30 and 59 and can optionally be programmed to carry POS data. The remaining payload is used entirely for the POS data.

Figure 1: A SONET/SDH STS-1/STM-0 Frame Showing the TOH

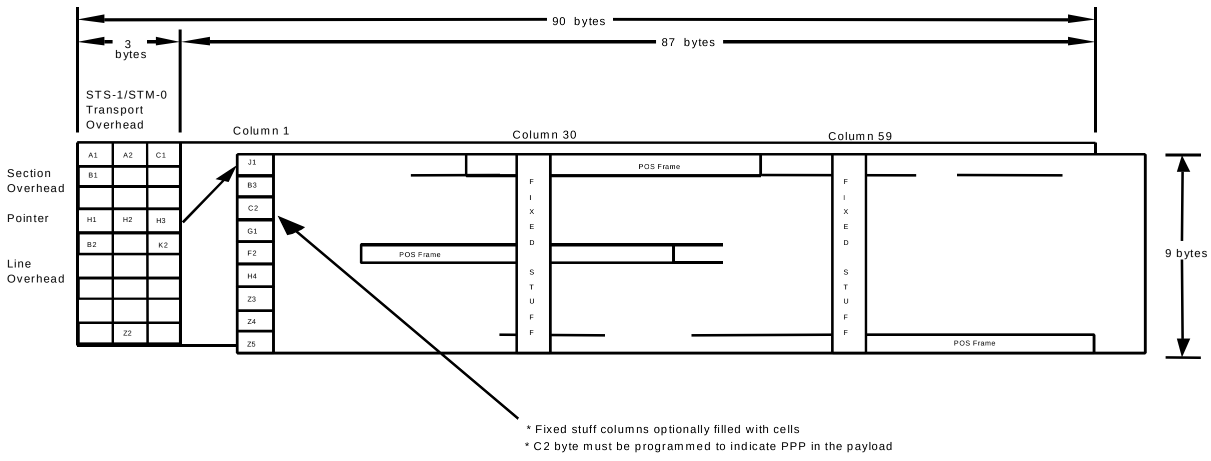
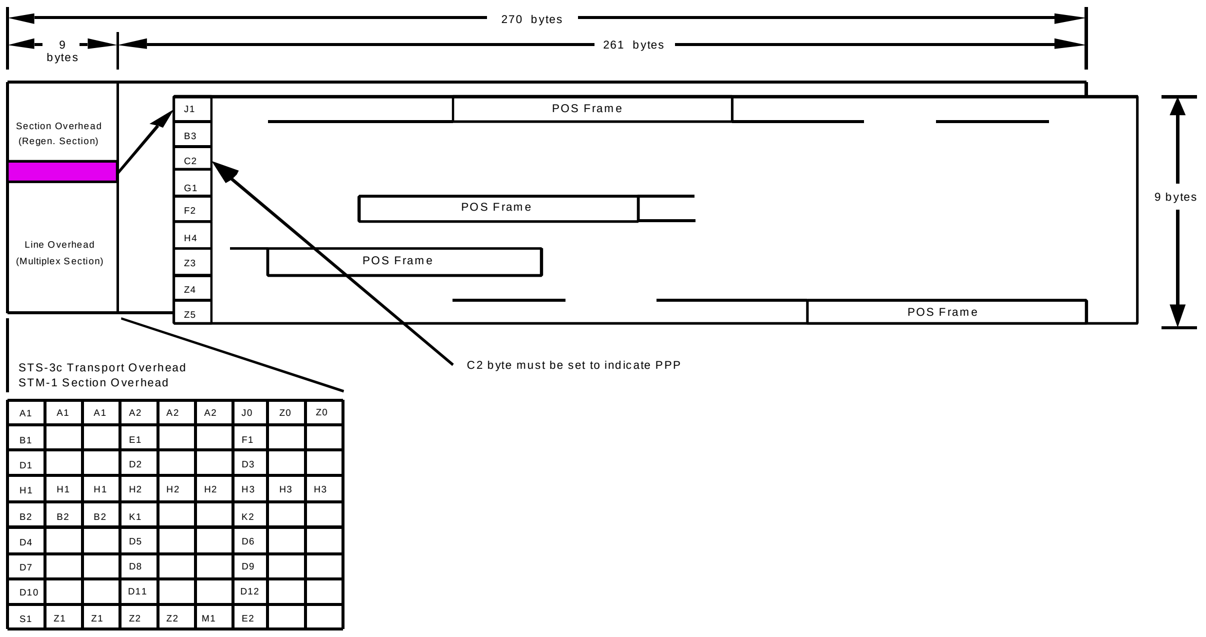


Figure 2 shows the STS-3c (STM-1) mapping. In this mapping, no stuff columns are included in the SPE. The entire SPE is used for PPP cell bytes.

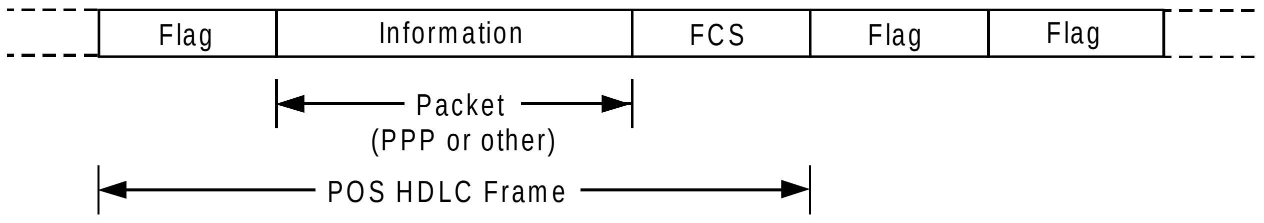
Figure 2: A SONET/SDH STS-3c/STM-1 Frame Showing the TOH



1.3 Packet over SONET frame format

The POS HDLC Processor provides for flexible implementation of Packet Over SONET, which allows to encapsulate various types of packets based protocols. The basic POS frame format is illustrated in Figure 3. POS frames are separated by a Flag Sequence, the 8 bit character 0x7E. This Flag Sequence is also used to fill inter-frame spacing when there is no data to be sent.. There must be a minimum of one Flag Sequence to delineate two POS frames. The POS frame closing Flag is normally preceded by either a 2 byte CRC-CCITT or 4 byte CRC-32 Frame Check Sequence (FCS). The FCS allows for error detection at the far end which can discard packets if errors are detected. The rest of the POS frame is the encapsulated data or packet. The POS HDLC processor does not process the packet data, other than byte stuffing, and provides a transparent transmission from end to end.

Figure 3: POS HDLC Frame Format



1.4 Payload Scrambling

Payload scrambling is extremely important in order to prevent a malicious user from causing problems on a network. In order to see how a malicious user could cause problems on a network it is important to understand the SONET scrambler in more detail. The SONET scrambler is a frame synchronous scrambler with the generating polynomial $1+X^6+X^7$. Since this is a frame synchronous scrambler, the scrambler is reset on each SONET frame. This means that the starting point of the scrambler is deterministic and not based on the previous history of the data stream. Also since the SONET scrambler is a 7th order scrambler, the pseudo random sequence repeats itself every 127 (2^7-1) bit periods. The sequence coming out of the scrambler is xored with the data to be transmitted. The sequence coming out of the scrambler is easily determinable and thus a malicious user can, by making a 1/127 assumption as to where his data lands in the SPE, control the output of the scrambled signal.

Applying a self-synchronous scrambler to the payload makes it virtually impossible for a malicious user to cause problems on a network. A self-synchronous scrambler is not aligned to any frame and thus the starting point is not deterministic and is in fact based on the previous history of the data stream. Thus a malicious user would have to know the entire history of the scrambler in order to create a data stream that would cause the output of the scrambler to be a sequence of constant identical digits.

Note that by applying a self-synchronous scrambler to the payload and then applying the SONET scrambler to the SONET frame (not including the framing bytes) means that a malicious user would have to be able to control the output of the $x^{43}+1$ self-synchronous scrambler in such a way as to produce a pattern (that would land in the correct position in the SPE) that would cause the SONET scrambler to generate a sequence of constant identical digits.

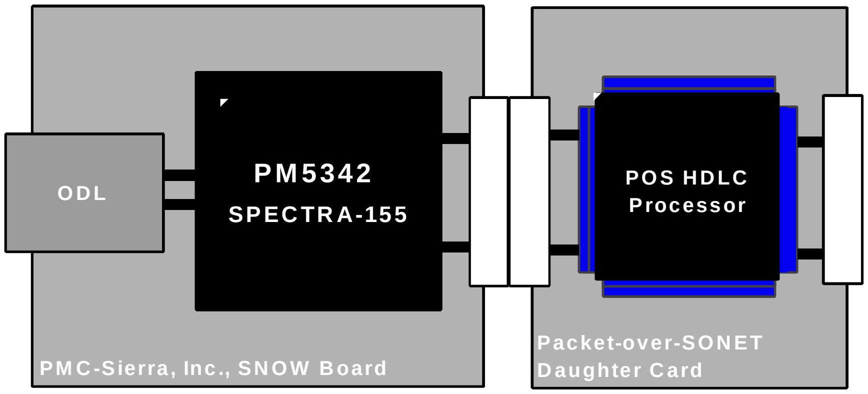
2 SYSTEM ARCHITECTURE FOR TRANSPORTING POS

The PM5342 SPECTRA-155 device can be used to transport data at either the STS-3c/STM-1 or the STS-1/STM-0 rate. The hardware architectures for both implementations are identical. The only difference between these two modes is that certain registers, described later on in this document in the SPECTRA-155 configuration section, must be set differently.

2.1 POS Mapped Over a STS-3c or STS-1 SONET/SDH SPE

Figure 4 below shows the system architecture required for implementing POS over a STS-3c/STM-1or STS-1/STM-0 rate using the SPECTRA-155.

Figure 4: POS over an STS-3c/STM1 or STS-1/STM-0 Link



The complete system is composed of two separate boards. The SONET/SDH Node Optical design with WAN filtering or SNOW board allows for the evaluation and demonstration of the PM5342 SPECTRA-155 device. This board has been specifically designed to mate with a system side application board to form an ATM, HDLC or SONET/SDH cross connect system. The SNOW board is comprised of two main blocks:

<u>ODL</u>

The optical data link (ODL) can be any optical transceiver designed to operate up to and including OC-3 (155 Mbits/s) rates.

SPECTRA-155

The SPECTRA-155 is a SONET/SDH Extractor/Aligner for use in STS-1 (STM-0/AU3), STS-3 (STM-1/AU3) or STS-3c (STM-1/AU4) interface applications. For a complete description of the SPECTRA-155 please refer to the PM5342 SPECTRA-155 Datasheet and the SNOW board reference design.

The SPECTRA-155 has been designed to allow for easy implementation of Packet over SONET (POS) applications. The system side interface of the PM5342 SPECTRA-155 can be configured to operate in a byte wide data bus mode. By using the SPECTRA-155 in this mode it is very easy to insert/extract POS data from the device. The SPECTRA-155 can also be configured to apply a self-synchronous X^43+1 scrambler/descrambler to the data stream.

The Packet over SONET daughter card is designed to mate to the SNOW board to implement a POS HDLC System. The POS daughter card is comprised of the POS HDLC Processor and an interface to a link layer device.

POS HDLC Processor

The POS HDLC Processor consists of a transmit HDLC Controller and FIFO and a receive HDLC Controller and FIFO.

In the transmit direction, the HDLC controller performs the following functions:

- Encapsulates PPP packets within a POS HDLC frame.
- Generates opening and closing flag sequence insertion.
- Can optionally insert a programmable number of flags between consecutive packets.
- Performs byte stuffing to provide transparency to control characters.
- Performs frame check sequence generation. The HDLC controller supports the generation of both CRC-CCITT and CRC-32 frame check sequences.
- Can optionally insert FCS errors for diagnostic purposes.
- Aborts packets under the direction of the host or when the FIFO underflows.
- Generates an interrupt if the FIFO underflows

- The HDLC controller does not perform self synchronous packet scrambling (1+X^{43} polynomial) since this function is performed by the SPECTRA-155.

In the receive direction, the HDLC Controller performs the following functions:

- Performs flag sequence detection and terminates the received HDLC frame.
- Performs frame check sequence (FCS) validation. The HDLC controller supports the validation of both CRC-CCITT and CRC-32 frame check sequences.
- Performs Control Escape de-stuffing.
- Checks for a packet abort sequence.
- The HDLC controller does not perform self synchronous PPP de-scrambling on the SPE payload since this function is already performed by the SPECTRA-155.
- Generates interrupts on FIFO overflows.

The TX and RX FIFO’s allow for rate decoupling between the line side and system side timing domains.

Link Layer Interface

In the transmit direction, the link layer interface passes data from memory to the HDLC Processor. In the receive direction, data is passed from the HDLC Processor to the link layer interface which sends the data to memory.

3 SPECTRA-155 CONFIGURATION

As mentioned previously, this implementation requires that the system side interface of the SPECTRA-155 be configured for Byte Data mode. This is accomplished by connecting the SMODE[2:0] pins in the following manner:

Pin	Configuration
SMODE[2]	O – GND
SMODE[1]	1 – VDD
SMODE[0]	1 - VDD

In Byte (and Nibble) data mode it is possible to configure the SPECTRA-155 to scramble/descramble the PPP data. The scrambler/descrambler uses a self synchronous x^43 + 1 polynomial.

Register	Name	Bit	Function	Set
0X87	SPECTRA-155 Data Mode Configuration	5	RDM_SCRMEN	1
		1	TDM_SCRMEN	1

As shown below, certain registers also need to be configured depending on whether a STS-3c or STS-1 line side interface is required.

STS-3c Configuration:

Register	Name	Bit	Function	Set
0X00	SPECTRA-155 Configuration	7	TMODE[1]	1
		6	TMODE[0]	1
		1	RMODE[1]	1
		0	RMODE[1]	1

STS-1 Configuration:

In STS-1 mode it is possible to configure the SPECTRA-155 to use the Fixed Stuff columns of the STS-1 SPE to carry data. The TDM_FSEN/RDM_FSEN bits must be set in order to configure the device to insert/extract data to/from the fixed stuff columns.

STS-1 mode with the Fixed Stuff columns configured to carry data:

Register	Name	Bit	Function	Set
0X00	SPECTRA-155 Configuration	7	TMODE[1]	0
		6	TMODE[0]	0
		1	RMODE[1]	0
		0	RMODE[1]	0
0X87	SPECTRA-155 Data Mode Configuration	4	RDM_FSEN	1
		0	TDM_FSEN	1

STS-1 mode with the Fixed Stuff columns configured to NOT carry data:

Register	Name	Bit	Function	Set
0X00	SPECTRA-155 Configuration	7	TMODE[1]	0
		6	TMODE[0]	0
		1	RMODE[1]	0
		0	RMODE[1]	0
0X87	SPECTRA-155 Data Mode Configuration	4	RDM_FSEN	0
		0	TDM_FSEN	0

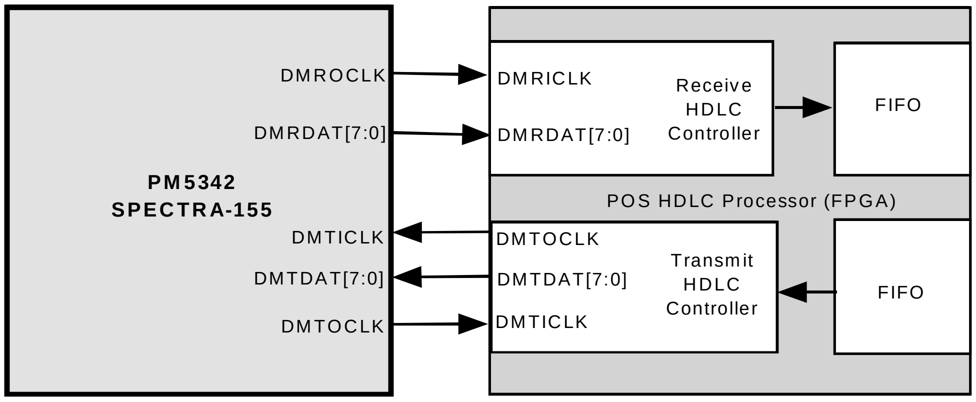
4 IMPLEMENTATION DETAILS

In order to implement the system described above, two interfaces need to be defined. Interface 1 is between the SPECTRA-155 and the HDLC Processor and interface 2 is between the HDLC Processor and an upstream link layer device. All signals are defined in detail in the POS HDLC Processor Design section.

4.1 SPECTRA-155 to POS HDLC Processor Interface

The system side interface of the SPECTRA-155 can be configured to operate in a byte wide data bus mode. In this mode of operation the SPECTRA-155 maps/de-maps the byte data stream into/from a STS-1 (STM-0/AU3) or STS-3c (STM-1/AU4) SPE. The DROP interface on the SPECTRA-155 consists simply of a receive output clock (which provides the timing for the receive HDLC controller) and an eight bit receive data bus. The ADD bus interface also has an eight bit transmit data bus but consists of two clock signals. The transmit output clock is used to provide timing for the upstream HDLC controller. The transmit input clock is used to clock data into the SPECTRA-155. below shows the interface between the SPECTRA-155 and the POS HDLC Processor.

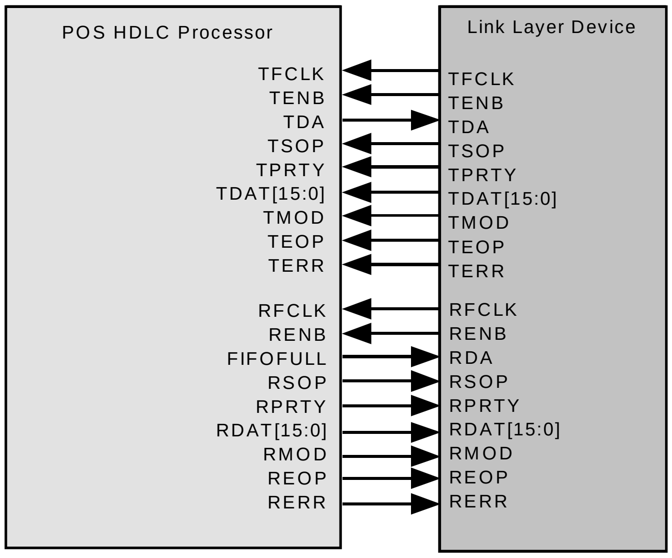
Figure 5: SPECTRA-155 to POS HDLC Processor Interface



4.2 POS HDLC Processor to Link Layer Device Interface

This interface is designed to directly connect to a POS link layer processor using a 256 byte synchronous FIFO interface over which packets are transferred. Figure 6 below shows the interface between the POS HDLC Processor and a PPP link layer processor.

Figure 6: POS HDLC Processor to Link Layer Device

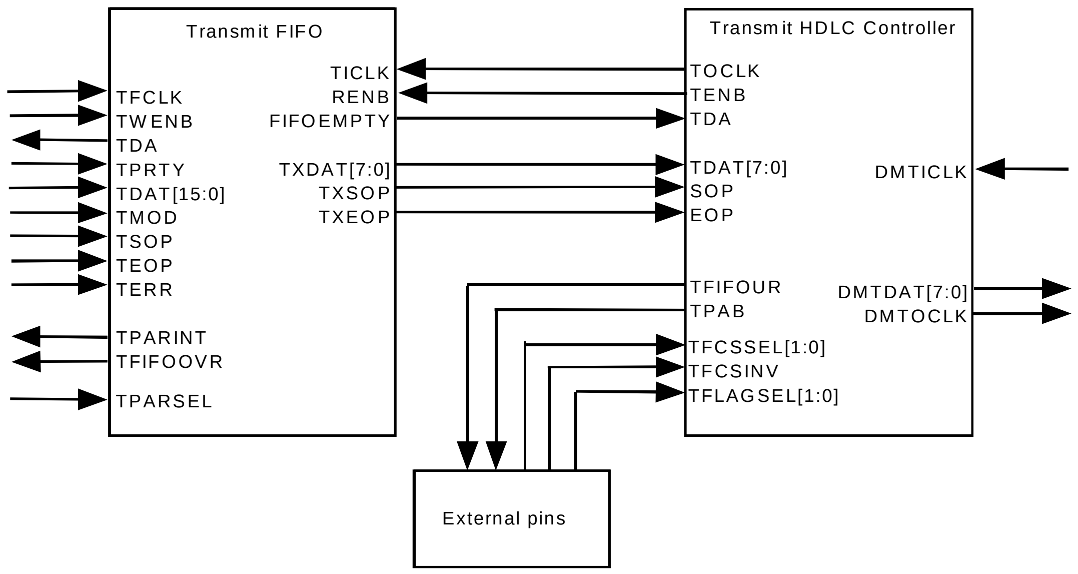


5 POS HDLC PROCESSOR DESIGN

The POS HDLC Processor is designed into a FPGA. As shown in Figure 5, the POS HDLC Processor is broken down into 2 main sections; transmit and receive.

The transmit section is also broken down into two separate blocks; a transmit HDLC controller and a transmit FIFO.

Figure 7: Transmit Section



The transmit HDLC controller performs the following functions:

- Encapsulates PPP packets within a POS HDLC frame.
- Generates opening and closing flag sequence insertion.
- Can optionally insert a programmable number of flags between consecutive packets.
- Performs byte stuffing for transparency processing.
- Performs frame check sequence generation. The HDLC controller supports the generation of both CRC-CCITT and CRC-32 frame check sequences.

- Can optionally insert FCS (by inverting the FCS) errors for diagnostic purposes.
- Aborts packets under the direction of the host or when the FIFO underflows.
- Generates an interrupt if the FIFO underflows

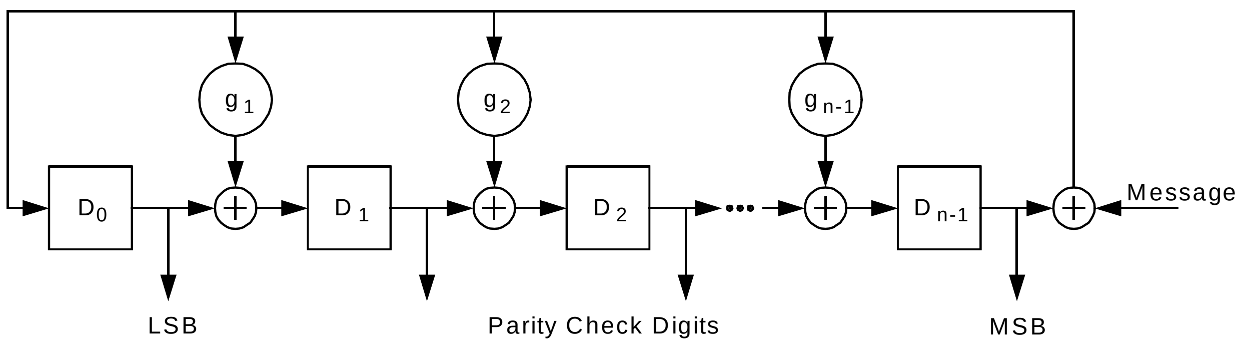
The transmit HDLC controller is clocked by DMTICLK, provided by the SPECTRA-155. Flags are inserted whenever the Transmit FIFO is empty and there is no packet to transmit. When there is a new packet in the transmit FIFO, the HDLC Controller operates normally; it removes the packets from the Transmit FIFO and transmits them to the SPECTRA-155 ADD interface. In addition, FCS generation and byte stuffing is performed on the data stream.

In the event of a FIFO underflow caused by the FIFO being empty while a packet is being transmitted, the packet is aborted by transmitting the abort sequence. The abort sequence consists of an escape control character (0x7D) followed by the flag sequence (0x7E). Bytes associated with this aborted frame are still read from the FIFO but are discarded and replaced with the flag sequence in the outgoing data stream. Transmission of data resumes when a start of packet is encountered in the FIFO data stream.

5.1 FCS Generator

The FCS Generator performs a CRC-CCITT or CRC-32 calculation on the whole POS frame, before byte stuffing and data scrambling. A parallel implementation of the CRC polynomial is used. The CRC algorithm for the frame checking sequence (FCS) field is either a CRC-CCITT or CRC-32 function. The CRC-CCITT is two bytes in size and has a generating polynomial $g(X) = 1 + X^5 + X^{12} + X^{16}$. The CRC-32 is four bytes in size and has a generating polynomial $g(X) = 1 + X + X^2 + X^4 + X^5 + X^7 + X^8 + X^{10} + X^{11} + X^{12} + X^{16} + X^{22} + X^{23} + X^{26} + X^{32}$. The first FCS bit transmitted is the coefficient of the highest term. When transmitting a packet from the Transmit FIFO, the FCS generator appends the result after the last data byte, before the closing flag.

Figure 8: CRC-32 Generator



An error insertion mechanism is provided for system diagnosis purposes. Error insertion is performed by inverting the resulting FCS value, before transmission. This should cause an FCS Error at the far end.

5.2 Byte Stuffing

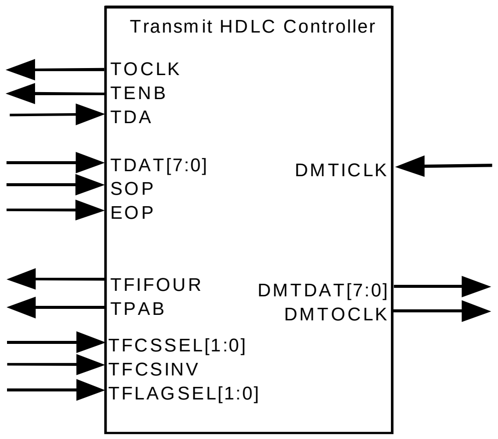
The transmit HDLC controller provides transparency by performing byte stuffing. Byte stuffing is required to escape real data that looks like special control characters. This operation is done after the FCS calculation. Two characters are being escaped (see Table 1 below), the Flag Sequence (0x7E) and the Escape Character itself (0x7D). When a character is being escaped, it is xored with 0x20 before transmission and preceded by the Control Escape (0x7D) character.

Table 1 Byte Stuffing Characters

Original PPP Data	**Escaped sequence**
7E (Flag Sequence)	7D-5E
7D (Control Escape)	7D-5D

The transmit HDLC controller is shown below in Figure 9.

Figure 9: Transmit HDLC Controller



Pin Name	Type	Function
TOCLK	Output	The Transmit Output Clock (TOCLK) is a buffered version of DMTICLK and is used to clock data out of the transmit FIFO into the transmit HDLC controller.
TENB	Output	The Transmit Enable signal is used to initiate reads from the transmit FIFO. When sampled low using the rising edge of TOCLK, a byte is transferred from the transmit FIFO to the transmit HDLC controller. This signal is updated on the rising edge of TOCLK.
TDA	Input	The Transmit Data Available signal indicates that there is data in the FIFO ready to be transmitted. The TDA signal will transition high when the minimum packet length (2 bytes) of data has been written into the FIFO. TDA will transition low when the FIFO is empty. TDA is sampled on the rising edge of TOCLK.
TDAT[7:0]	Input	The Transmit Data (TDAT) bus contains the raw POS data. TDAT is sampled on the rising edge of TOCLK.
SOP	Input	The Start of Packet (SOP) signal indicates that the byte being currently read in is the first byte of the packet. SOP is sampled on the rising edge of TOCLK.

EOP	Input	The End of Packet (EOP) signal indicates that the byte being currently read in is the last byte of the packet. EOP is sampled on the rising edge of TOCLK.
DMTICLK	Input	The Data Mode Transmit Input Clock (DMTICLK) is the timing reference for the HDLC controller. DMTICLK is tied to both the TOCLK, which is used to clock data into the HDLC controller from the transmit FIFO, and the DMTOCLK, which is used to clock data into the SPECTRA-155. DMTICLK is nominally a 18.72 MHz or 6.192 MHz clock generated by gapping a 19.44 MHz or 6.48 MHz clock respectively.
DMTDAT[7:0]	Output	The Data Mode Transmit Data (DMTDAT) bus contains the HDLC frame to be mapped into the SONET/SDH SPE. DMTDAT is updated on the rising edge of DMTICLK.
DMTOCLK	Output	The Data Mode Transmit Output Clock (DMTOCLK) provides timing to clock data into the SPECTRA-155.
TFCSSEL[1:0]	Input	The Transmit FCS Select (TFCSSEL) input pins are used to select between no CRC insertion, CRC-CCITT insertion, or CRC-32 insertion. See Table 2 for configuration specifications. The TFSCSEL pins are sampled on the rising edge of TOCLK.
TFCSINV	Input	The Transmit FCS Invert (TFCSINV) signal is used to invert the FCS which should cause an FCS error at the far end. TFCSINV is sampled on the rising edge of TOCLK.
TFLAGSEL[1:0]	Input	The Transmit Flag Select (TFLAGSEL) pins are used to program the number of flags to be transmitted between consecutive packets. See Table 3 below for configuration details. This signal is sampled on the rising edge of TOCLK.
TFIFOUR	Output	The Transmit FIFO Under Run (TFIFOUR) signal is used to indicate that a FIFO under run occurred, that is, the FIFO emptied before an End of Packet was detected. This signal is updated on the rising edge of TOCLK.
TPAB	Output	The Transmit Packet Abort (TPAB) signal is used to indicate that the packet currently transmitted was aborted. This signal is updated on the rising edge of TOCLK.

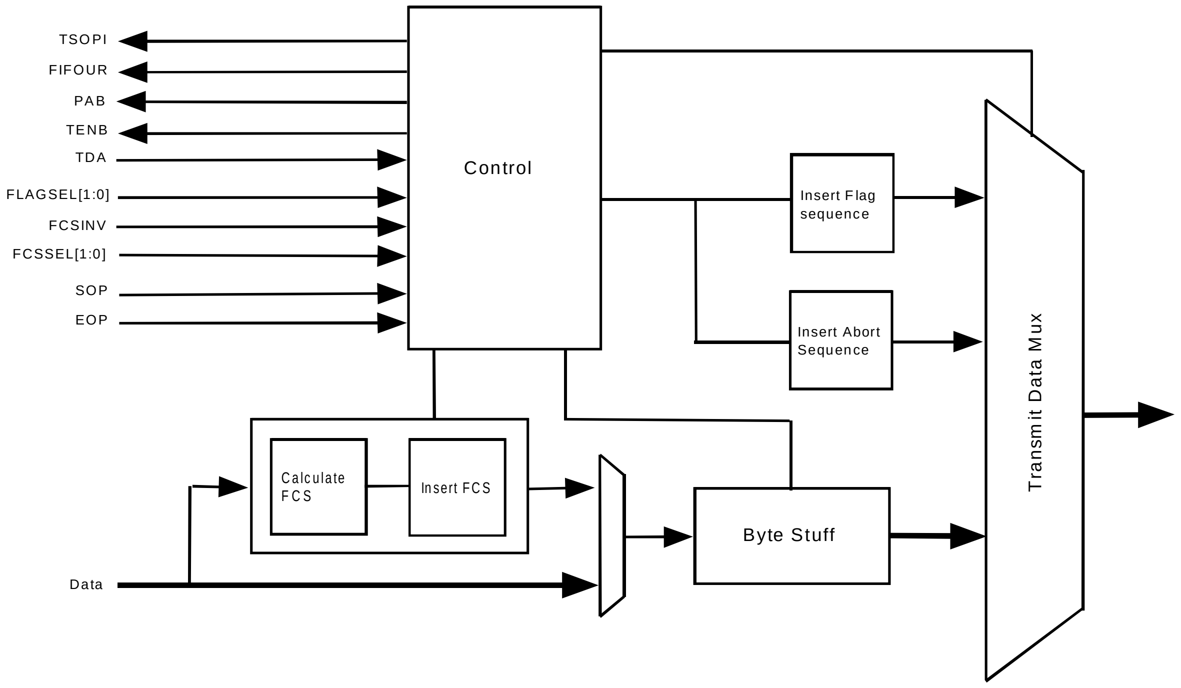
Table 2 Transmit FCSSEL Configuration

FCSSEL[1:0]	Description
00	No FCS insertion
01	2 byte CRC-CCITT insertion
10	4 byte CRC-32 insertion
11	Reserved

Table 3 FLAGSEL Configuration

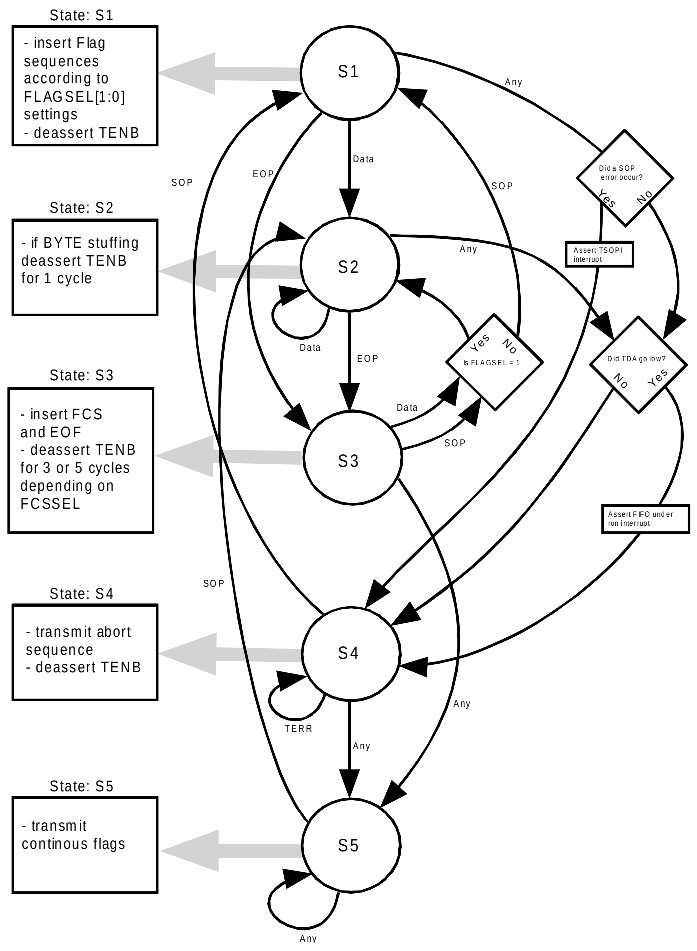
FLAGSEL[1:0]	Description
00	1 flag sequence is transmitted between consecutive packets. That is only one Flag sequence is used to indicate both the end of packet and the start of packet if the packet are immediately after each other, that is, there is no time delay between transmitting the end of packet flag for packet n and transmitting the first byte of packet n+1.
01	2 flag sequences is transmitted between consecutive packets. That is, each packet is started with a flag sequence and ended with a flag sequence.
10	8 flag sequences are transmitted between consecutive packets
11	16 flag sequences are transmitted between consecutive packets

Figure 10: Transmit HDLC Controller – detailed view



The flow of data in the transmit direction is shown above in Figure 10. As data is read out of the transmit FIFO it passes through the Byte Stuff block which searches for Control Escape characters and “stuffs” them to provide transparency in the transmit stream. At the same time, control signals from the transmit FIFO, such as SOP and EOP, tell the controller whether the byte is the first byte or the last byte of the packet. The controller appropriately insert flag sequences by controlling the input to the mux. The assertion of both the SOP and EOP signals simultaneously indicates that the current packet should be aborted and the controller consequentially inserts the abort sequence in the transmit data stream. When an EOP is detected, the controller inserts both the FCS and a flag sequence, in that order, into the transmit data stream.

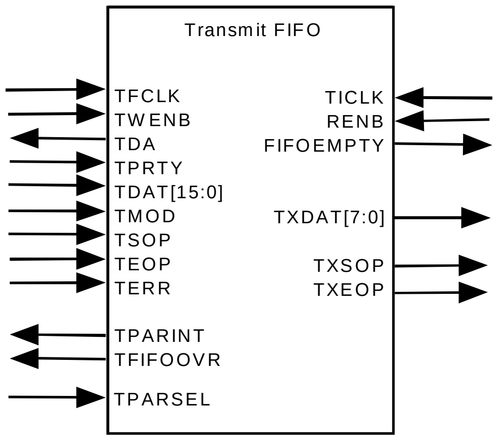
Figure 11: Transmit HDLC Controller State Machine



The Transmit HDLC (THDLC) Controller State Machine is comprised of 5 states. State S5 is the default state. In this state the THDLC Controller will continuously transmit flag sequences until a Start of Flag signal is detected, at which point the state machine will transition to state S2. Note that in this situation, a transition to state S1 does not occur because a Start of Flag sequence does not have to be inserted in the transmit stream since in state S5 the THDLC Controller was already continuously transmitting flag sequences. In state S2, the data is just transmitted directly unless an escape sequence is detected, in which case byte stuffing is performed. The state machine will remain in state S2 as long as the control signals indicate that data is being transmitted. The THDLC Controller will transition to state S4 if any signal other than the End of Packet signal is detected. If, for example, the FIFO empties (that is TDA goes Low) before an End of Packet signal is detected, the THDLC Controller will assert the FIFO underrun interrupt signal and transition into state S4. The state machine will transition from state S2 to state S3 if an End of Packet signal is detected. In state S3, the End of Packet flag sequence is inserted in the transmit data stream. The FCS can also optionally, depending on the FCSSEL[1:0] settings, be inserted into the transmit data stream. The FCS precedes the End of Packet flag sequence. A transition to the default state, S5, will occur if any signal other than the Data or Start of Packet signal is detected. If either the Data or Start of Packet signals are detected, a transition to either state S2 or state S1 will occur depending on the FLAGSEL[1:0] settings. If FLAGSEL[1:0] is set to 00, that is, only one flag sequence is transmitted between consecutive packets, than the state machine will transition from state S3 to state S2 otherwise the transition will be from state S3 to state S1. A transition into state S4 indicates that an error has occurred and that the current packet is to be aborted. Hence in state S4, the current packet is aborted by transmitting the abort sequence. The Packet Abort signal is also asserted. The state machine will transition from state S4 to S5 if any signal other than the Start of Packet signal, in which case a transition to state S1 will occur, is detected. In state S1 the Start of Flag sequence is inserted into the transmit data stream. If any other signal other than the Data or End of Packet signals are detected the state machine will transition from state S1 to S4. The state machine will transition from state S1 to S2 if the control signals indicate that the next byte is Data or will transition from state S1 to S3 if the next byte is also the end of packet.

The transmit FIFO, shown in Figure 12, is responsible for holding packets provided through the input interface until they are transmitted. The transmit FIFO can accommodate a maximum of 256 bytes. Octets are written in with a single 16 bit data bus clocked by TFCLK and read out with a single 8 bit data bus clocked by TICLK. Separate read and write clocks provide for rate decoupling between the line side and system side timing domains.

Figure 12: Transmit FIFO



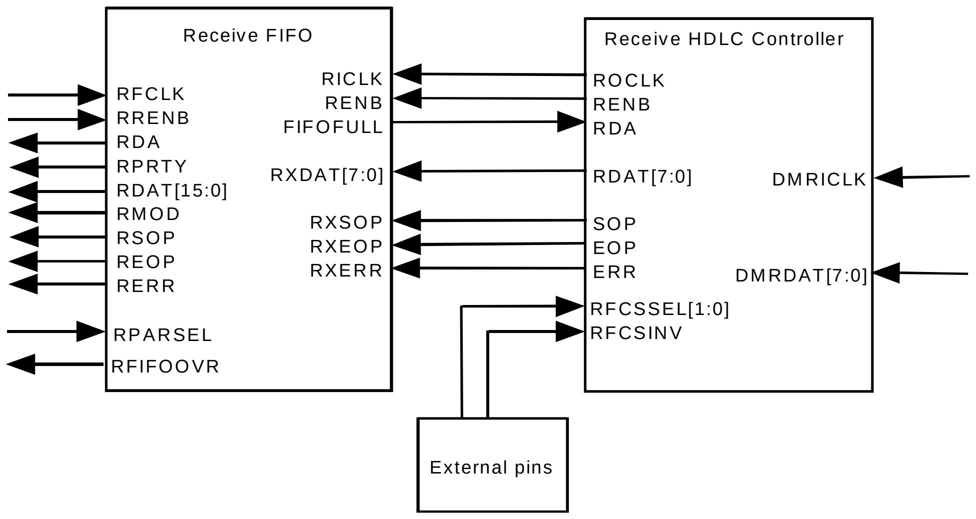
Pin Name	Type	Function
TFCLK	Input	The Transmit FIFO Clock (TFCLK) is used to write POS data into the 256 byte transmit FIFO. TFCLK cycles at a 50 MHz or lower instantaneous rate.
TWENB	Input	The Transmit Write Enable (TWENB) signal is an active low input which is used to initiate writes to the transmit FIFO. When sampled low using the rising edge of TFCLK, the word on the TDAT bus is written into the transmit FIFO. When sampled high using the rising edge of TFCLK, no write is performed.
TDA	Output	The TDA signal indicates that the FIFO is not full and thus can accept at least a full word. When TDA is low the FIFO is full. This signal is updated on the rising edge of TFCLK.
TSOP	Input	The Transmit Start Of Packet (TSOP) signal marks the start of a packet on the TDAT bus. When TSOP is high, the first word of the packet structure is present on the TDAT bus. TSOP is required to be present at the beginning of every packet. An interrupt is generated if TSOP is high during any word other than the first word of the packet structure. TSOP is sampled on the rising edge of TFCLK and is considered valid only when TENB is simultaneously asserted.

TPRTY	Input	The Transmit Parity (TPRTY) signal indicates the parity of the TDAT[15:0] bus. Odd or even parity selection is made using the PARSEL input.
TDAT[15:0]	Input	The Transmit Data Bus (TDAT[15:0]) carries the POS packet octets that are written to the transmit FIFO. TDAT[15:0] is sampled on the rising edge of TFCLK and is considered valid only when TENB is simultaneously asserted.
TMOD	Input	The Transmit Transfer size signal (TMOD) indicates the number of bytes transferred in the current word (modulo 2). TMOD is only used during the last cycle of a packet transfer, that is, when TEOP is asserted. This signal is sampled on the rising edge of TFCLK.
TERR	Input	The Transmit Error (TERR) signal is used to indicate the current packet should be aborted. This signal is sampled on the rising edge of TFCLK.
TEOP	Input	The Transmit End Of Packet (TEOP) signal is used to indicate the last word of a packet transfer. The receive size TMOD signal identifies how many bytes are valid during an end of packet cycle. This signal is sampled on the rising edge of TFCLK.
TPARSEL	Input	The Transmit Parity Select (TPARSEL) signal is used to select between odd or even parity. A logic level High on this input selects odd parity while a logic level Low selects even parity. This signal is sampled on the rising edge of TFCLK.
TPARINT	Output	The Transmit Parity Interrupt (TPARINT) signal is used to indicate that a parity error was detected on TDAT[15:0]. This signal is updated on the rising edge of TFCLK.
TFIFOOVR	Output	The Transmit FIFO Over Run (TFIFOOVR) signal is used to indicate that a FIFO over run has occurred. This signal is updated on the rising edge of TFCLK.

TXSOP	Output	The Transmit Start of Packet (TXSOP) signal is used to indicate, to the HDLC controller, that the byte currently being read out of the FIFO is the first byte of the packet. This signal is updated on the rising edge of TICLK.
TXEOP	Output	The Transmit End of Packet (TXEOP) signal is used to indicate, to the HDLC controller, that the byte currently being read out of the FIFO is the last byte of the packet. This signal is updated on the rising edge of TICLK.
TXDAT[7:0]	Output	The Transmit Data (TXDAT) bus contains the raw data. This data is encapsulated in an HDLC frame by the transmit HDLC controller. This signal is updated on the rising edge of TICLK.
FIFOEMPTY	Output	The FIFOEMPTY signal indicates that there is no more data in the FIFO. The FIFOEMPTY signal will transition low when the minimum packet length (2 bytes) of data has been written into the FIFO. FIFOEMPTY will transition high when the FIFO is empty.
RENB	Input	The Receive Enable (RENB) signal is used to initiate reads from the transmit FIFO. This signal is sampled on the rising edge of TICLK.
TICLK	Input	The Transmit Input Clock (TICLK) is used to clock data out of the transmit FIFO.

The receive section is also broken down into two separate blocks; a receive HDLC controller and a receive FIFO.

Figure 13: Receive Section



The receive HDLC controller performs the following functions:

- Performs flag sequence detection and terminates the received HDLC frame.
- Performs frame check sequence (FCS) validation. The HDLC controller supports the validation of both CRC-CCITT and CRC-32 frame check sequences.
- Performs Control Escape de-stuffing.
- Checks for a packet abort sequence.
- The HDLC controller does not perform self synchronous PPP de-scrambling on the SPE payload since this function is already performed by the SPECTRA-155.
- Generates interrupts on FIFO overflows.

The receive HDLC controller accepts data one byte at a time from the SPECTRA-155 and arranges it as POS framed octets. Frame boundaries are found by searching for the Flag Character (0x7E). Flags are also used to fill inter-packet spacing. This block removes the Flag Sequence and passes the data onto the Byte Destuffing block.

In the event of a FIFO overflow caused by the FIFO being full while a packet is being received, the packet is marked with an error, by asserting the ERR signal, so it can be discarded by the system. Following bytes associated with this now aborted packet are discarded by the Receive HDLC Controller. Reception of POS data resumes when a Start of Packet is encountered.

5.3 Byte Destuffing

The byte destuffing algorithm searches the Control Escape character (0x7D). These characters are added for transparency in the transmit direction, as shown in Table 1, and must be removed to recover the user data. When the Control Escape character is encountered, it is removed and the following data byte is XORed with 0x02. Only the Flag Sequence (0x7E) and the Control Escape character itself are expected to have been escaped in the transmit direction, but this implementation does not preclude escaping other values as well.

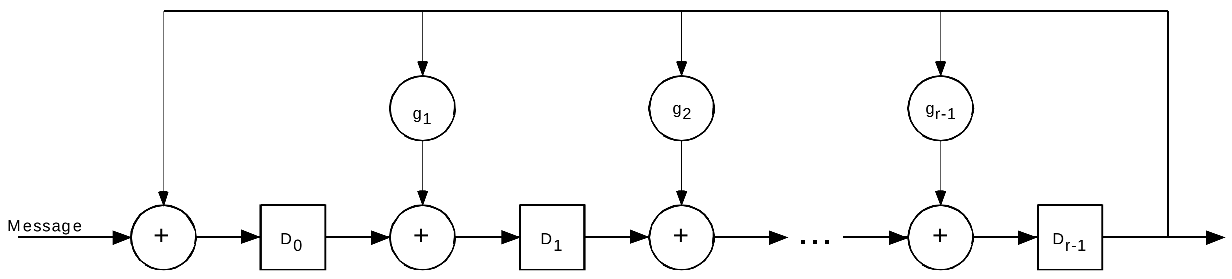
Table 4 Byte Destuffing Characters

Stuffed POS Data	De-stuffed Data
7D-5E	7E
7D-5D	7D

5.4 FCS Check

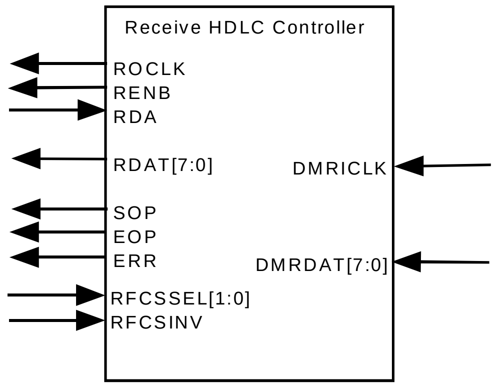
The FCS Generator performs a CRC-CCITT or CRC-32 calculation the whole POS frame, after byte destuffing and data descrambling scrambling. A parallel implementation of the CRC polynomial is used. The CRC algorithm for the frame checking sequence (FCS) field is either a CRC-CCITT or CRC-32 function. The CRC-CCITT is two bytes in size and has a generating polynomial $g(X) = 1 + X^5 + X^{12} + X^{16}$. The CRC-32 is four bytes in size and has a generating polynomial $g(X) = 1 + X + X^2 + X^4 + X^5 + X^7 + X^8 + X^{10} + X^{11} + X^{12} + X^{16} + X^{22} + X^{23} + X^{26} + X^{32}$. The first FCS bit transmitted is the coefficient of the highest term. The decoder is designed such that after computation over the whole packet, including the FCS field, the result should be all zeros. A different value indicates an error. Packets with FCS errors are marked as such and should be silently discarded by the system.

Figure 14: CRC-32 Decoder



The Receive HDLC controller is shown below in Figure 15.

Figure 15: Receive HDLC Controller



Pin Name	Type	Function
ROCLK	Output	The Receive Output Clock (ROCLK) is a buffered version of DMRICLK and is used to clock data into the receive FIFO.
RENB	Output	The Receive Read Enable (RENB) signal is used to initiate reads from the receive FIFO. RENB is updated on the rising edge ROCLK.
RDA	Input	The Receive Data Available (RDA) signal is used to indicate that the FIFO is not full and thus can accept at least one word of data.
RDAT[7:0]	Output	The Receive Data (RDAT) bus contains the extracted POS data. RDAT is updated on the rising edge of ROCLK.

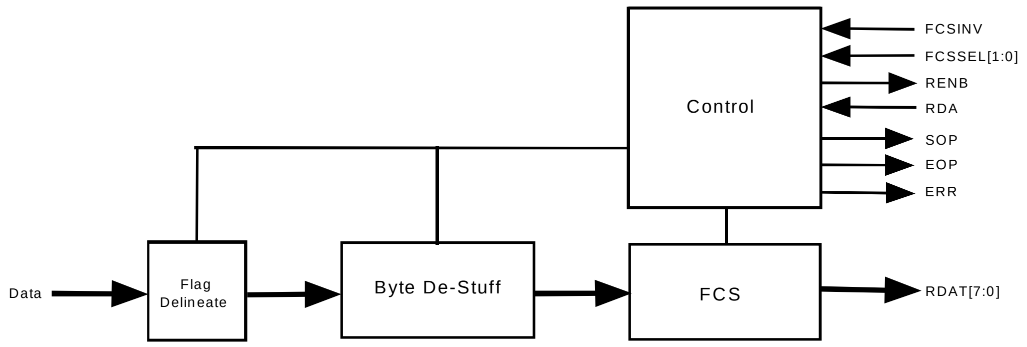
DMRICLK	Input	The Data Mode Receive Input Clock (DMRICLK) is the timing reference for the receive HDLC controller. DMRICLK is tied to ROCLK, which is used to clock data from the receive HDLC controller into the receive FIFO. DMRICLK is used to clock in DMRDAT.
DMRDAT[7:0]	Input	The Data Mode Receive Data (DMRDAT) bus contains the SONET/SDH SPE receive payload data. DMRDAT is sampled on the falling edge of DMRICLK.
RFCSSEL[1:0]	Input	The Receive FCS Select (RFCSSEL) pins are used to select between no CRC, CRC-CCITT or CRC-32 calculation. See Table 5 below for configuration details. RFCSSEL pins are sampled on the rising edge of ROCLK.
RFCSINV	Input	The Receive FCS Invert (RFCSINV) signal is used to invert the FCS calculation. When RFCSINV is High, the FCS calculation is inverted. When RFCSINV is Low, the FCS is calculated normally. RFCSINV is sampled on the rising edge of ROCLK.
SOP	Output	The Start of Packet (SOP) signal is used to indicate that the byte currently being written out of the receive HDLC controller is the first byte of the packet. This signal is updated on the rising edge of ROCLK.
EOP	Output	The End of Packet (EOP) signal is used to indicate that the byte currently being written out of the receive HDLC controller is the last byte of the packet. This signal is updated on the rising edge of ROCLK.
ERR	Output	The Error (ERR) signal is used to indicate that an error in the receive data stream was encountered, that is, and abort sequence could have been detected or an insufficient number of flags between packets, etc. This signal is updated on the rising edge of ROCLK.

Table 5 Receive FCSSEL Configuration

FCSSEL[1:0]	Description
00	No FCS calculation

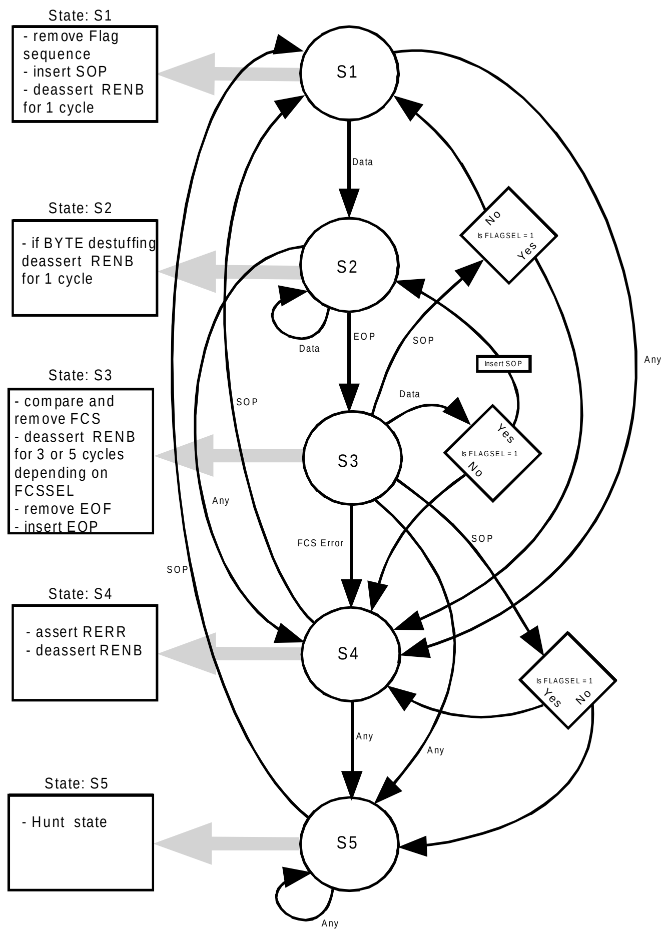
FCSSEL[1:0]	Description
01	CRC-CCITT calculation
10	CRC-32 calculation
11	Reserved

Figure 16: Receive HDLC Controller – detailed view



The flow of data through the receive HDLC Controller is shown above in Figure 16. As data is received, flag sequences are removed from the data path. The controller keeps track of whether or not the flag sequence is the first flag sequence, indicating a SOP (Start of Packet), or the second flag sequence, indicating an EOP (End of Packet) of the received packet. The data is then de-stuffed, that is, escape characters that were inserted in the transmit stream are removed (see Sections 5.2 and 5.3). When the controller detects an End of Packet, it removes the preceding two or four bytes, depending on the FCS selection (CRC-CCITT or CRC-32). A FCS is calculated on the de-stuffed data. The calculated FCS is then compared to the FCS removed from the data stream to see if they match. If they match, then the data can be assumed to have been received correctly, otherwise an error has occurred and the controller will assert the ERR signal.

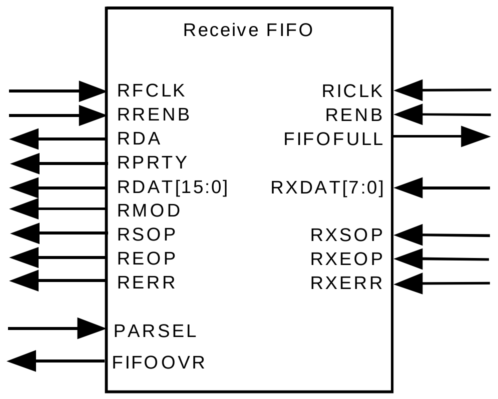
Figure 17: Receive HDLC Controller State Machine



The Receive HDLC (RHDLC) Controller State Machine is comprised of 5 states. State S5 is the default state. In this state, the RHDLC Controller is continuously searching for a flag sequence followed by a byte of data. If this flag sequence followed by a byte of data sequence is detected than the state machine will transition from state S5 to state S1, otherwise it will remain in state S5. In state S1, the flag sequence is removed from the data stream. The data byte, along with a Start of Packet indication is written into the FIFO. A transition from state S1 to state S2 will occur if the next byte is a Data byte, anything else will cause a transition to state S4. In state S2 the incoming data is written directly into the FIFO unless an escape sequence is detected, in which case the data is de-stuffed. The state machine will remain in state S2 as long as Data is being received. The reception of a flag sequence would indicate an end of packet and will cause a transition from state S2 to S3, anything else will cause a transition to state S4. In state S3 the FCS, which was calculated on the just received packet, is compared to the FCS transmitted for the current packet to check for errors. If errors are detected, a transition to state S4 occurs. If the RHDLC Controller is configured to detect only one flag sequence between consecutive packets and the byte following the just received flag sequence is data, and not another flag sequence, then a transition from state S3 to S2 will occur. The start of packet indication will also be asserted. If errors are detected, a transition to state S4 occurs. If the RHDLC Controller is not configured to detect only one flag sequence between consecutive packets and the byte following the just received flag sequence is data, and not another flag sequence, then a transition from state S3 to S4 will occur. If the RHDLC Controller is configured to detect more than one flag sequence between packets and the byte following the last received flag sequence is another flag sequence than a transition from state S3 to S5 will occur. If the RHDLC Controller is configured to detect more than one flag sequence between packets and the byte following the last received flag sequence is data and not another flag sequence than a transition from state S3 to S4 will occur. In state S4 the RERR signal is asserted to indicate to the link layer device that an error has occurred with the current packet and that it should be aborted.

The receive FIFO is used to provide rate decoupling between the line side and system side timing domains. The receive FIFO interface is shown below in Figure 18.

Figure 18: Receive FIFO



Pin Name	Type	Function
RFCLK	Input	The Receive FIFO Read Clock (RFCLK) is used to read the POS packets from the receive FIFO's. RFCLK must cycle at a 50 MHz or lower instantaneous rate, but at a high enough rate

		to avoid FIFO overflows.
RRENB	Input	The Receive Read Enable (RRENB) signal is used to initiate reads from the receive FIFO. When sampled low using the rising edge of RFCLK, a byte is read (if one is available) from the receive FIFO and output on the RDAT bus. RRENB must operate in conjunction with RFCLK to access the FIFO's at a high enough rate to prevent FIFO overflows. The Link layer device may de-assert RENB at anytime if it is unable to accept another byte. RRENB is sampled on the rising edge of RFCLK.
RDA	Output	The Receive Data Available (RDA) signal indicates that there is data in the FIFO available to be read out. RDA is updated on the rising edge of RFCLK.
RSOP	Output	The Receive Start of Packet (RSOP) signal marks the start of a packet on the RDAT bus. RSOP is updated on the rising edge of RFCLK.
RPRTY	Output	The Receive Parity (RPRTY) signal indicates the parity of the RDAT bus. Odd or even parity selection is made using the PARSEL pin. RPRTY is updated on the rising edge of RFCLK.
RDAT[15:0]	Output	The Receive Packet Data Bus (RDAT[15:0]) carries the POS packet octets that are read from the receive FIFO. RDAT is updated on the rising edge of RFCLK.
RMOD	Output	The Receive transfer size (RMOD) signal indicates the number of bytes transferred in the current word (modulo 2). RMOD is only used during the last cycle of a packet transfer, that is, when REOP is asserted. RMOD is updated on the rising edge of RFCLK.
REOP	Output	The Receive End Of Packet (REOP) signal indicates the last word of a packet transfer. The receive size RMOD signal identifies how many bytes are valid during an end of packet cycle. REOP is updated on the rising edge of RFCLK.
RERR	Output	The Receive Error (RERR) indicator is used to indicate that the current packet is aborted and should be discarded. RERR is updated on the rising edge of RFCLK.
RICLK	Input	The Receive Input Clock is a buffered version of DMROCLK

		and is used to clock data in to the receive FIFO from the receive HDLC controller.
RENB	Input	The Receive Enable (RENB) signal is used initiate writes into the receive FIFO from the receive HDLC controller. RENB is sampled on the rising edge of RICLK.
FIFOFULL	Output	The FIFOFULL signal is used to indicate to the HDLC controller that the FIFO is full and can accept no more writes. FIFOFULL is updated on the rising edge of RICLK.
RXDAT[7:0]	Input	The Receive Data (RXDAT) bus contains the POS data. RXDAT[7:0] are sampled on the rising edge of RICLK.
RXSOP	Input	The Receive Start of Packet (RXSOP) signal is used to indicate that the byte currently being read into the FIFO is the first byte of the packet. This signal is sampled on the rising edge of RICLK.
RXEOP	Input	The Receive Endo of Packet (RXEOP) signal is used to indicate that the byte currently being read into the FIFO is the last byte of the packet. RXEOP is sampled on the rising edge of RICLK.
RXERR	Input	The Receive Error (RXERR) signal is used to indicate that the packet currently being read out of the FIFO has been aborted. RXERR is sampled on the rising edge of RICLK.

6 FUNCTIONAL TIMING

Figure 19: Transmit HDLC Controller – SPECTRA-155 Timing

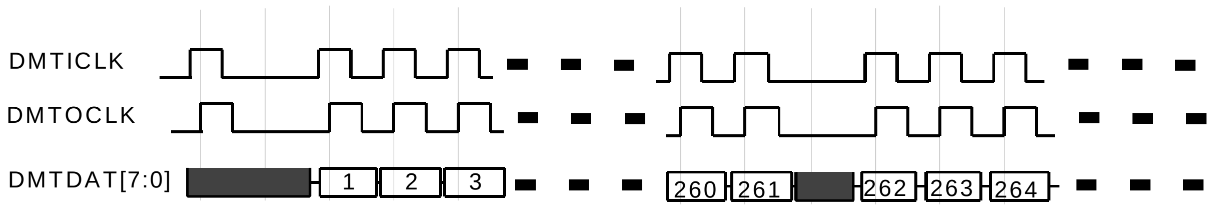


Figure 19 above shows the basic transmit bus timing between the SPECTRA-155 and the transmit HDLC Controller. DMTOCLK is a gapped clock used to clock data to the SPECTRA-155. Data bytes on the DMTDAT[7:0] bus are updated on the rising edge of DMTICLK. In STS-1 (STM-0/AU3) data mode, the gapped clock is generated using a 6.48 MHz clock. The nominal frequency of DMTICLK is 6.048 MHz if each transmitted synchronous payload envelope (SPE) has two fixed stuff columns. Otherwise, the nominal frequency of DMTICLK is 6.192 MHz. In STS-3c (STM-1/AU4) data mode, the gapped clock is generated using a 19.44 MHz clock. The nominal frequency of DMTICLK is 18.72 MHz.

Figure 20: Transmit HDLC Controller – FIFO Interface timing

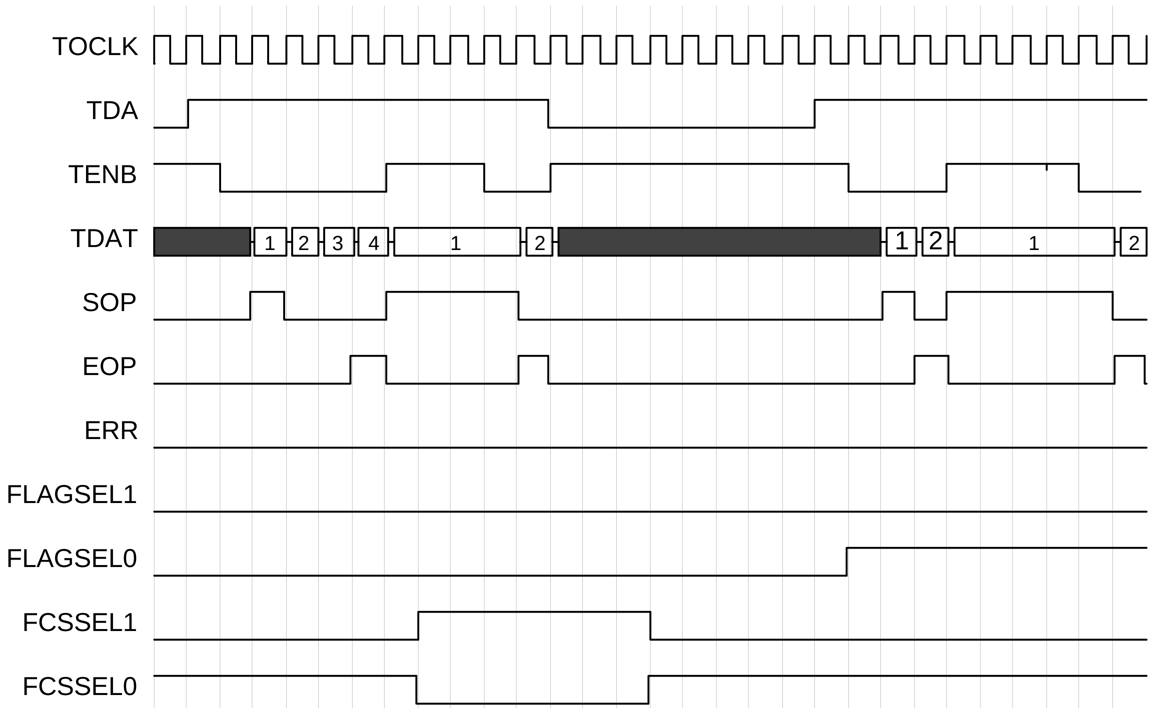
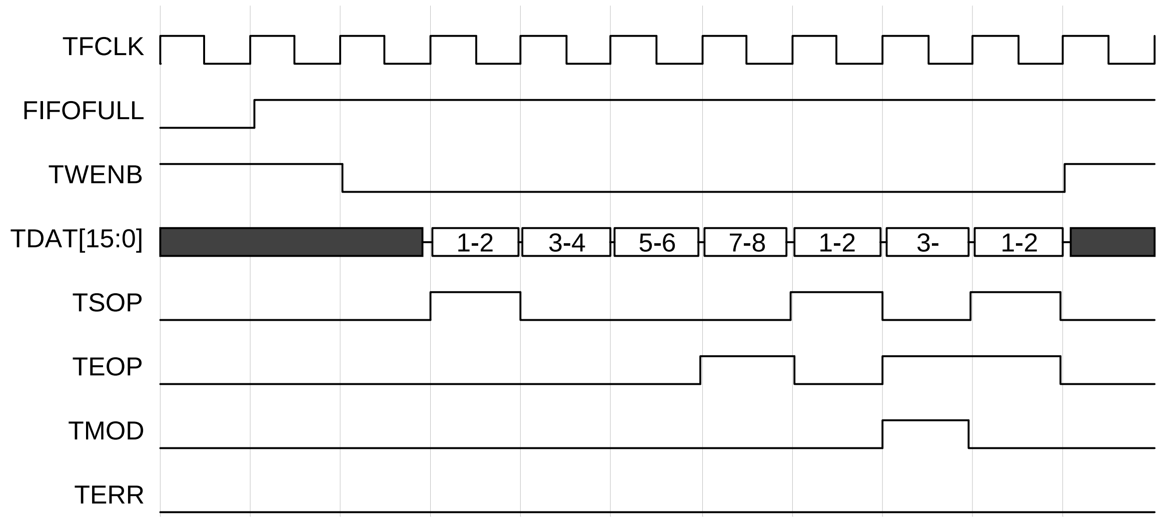


Figure 20 above shows the basic functional timing between the Transmit HDLC Controller and the transmit FIFO.

Figure 21: Transmit FIFO – Link Layer Timing



The transmit FIFO timing diagram (Figure 21) illustrates the basic operation of the drop side transmit FIFO timing. Assertion of the FIFOFULL signal indicates that the FIFO is full and can accept no more writes. TSOP must be high during the first word of the packet and must be present (reasserted) for each packet. TMOD is only used on the last word of the packet and is used to determine if the last word of the packet is composed of one or two bytes. If TSOP is asserted and the previous word transfer was not marked with a TEOP, the input interface realigns itself to the new timing, and the previous packet is marked to be aborted.

Figure 22: Receive HDLC Controller – SPECTRA-155 Timing

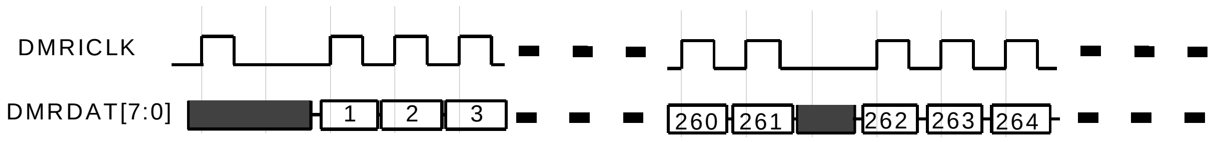


Figure 22 above shows the basic receive bus timing between the SPECTRA-155 and the receive HDLC Controller. DMRICLK is a gapped clock. Data bytes on the DMRDAT[7:0] bus are updated on the falling edge of DMRICLK. In STS-1 (STM-0/AU3) data mode, the gapped clock is generated using a 6.48 MHz clock. The nominal frequency of DMRICLK is 6.048 MHz if each received synchronous payload envelope (SPE) has two fixed stuff columns. Otherwise, the nominal frequency of DMRICLK is 6.192 MHz. In STS-3c (STM-1/AU4) data mode, the gapped clock is generated using a 19.44 MHz clock. The nominal frequency of DMRICLK is 18.72 MHz.

Figure 23: Receive HDLC Controller – FIFO Timing

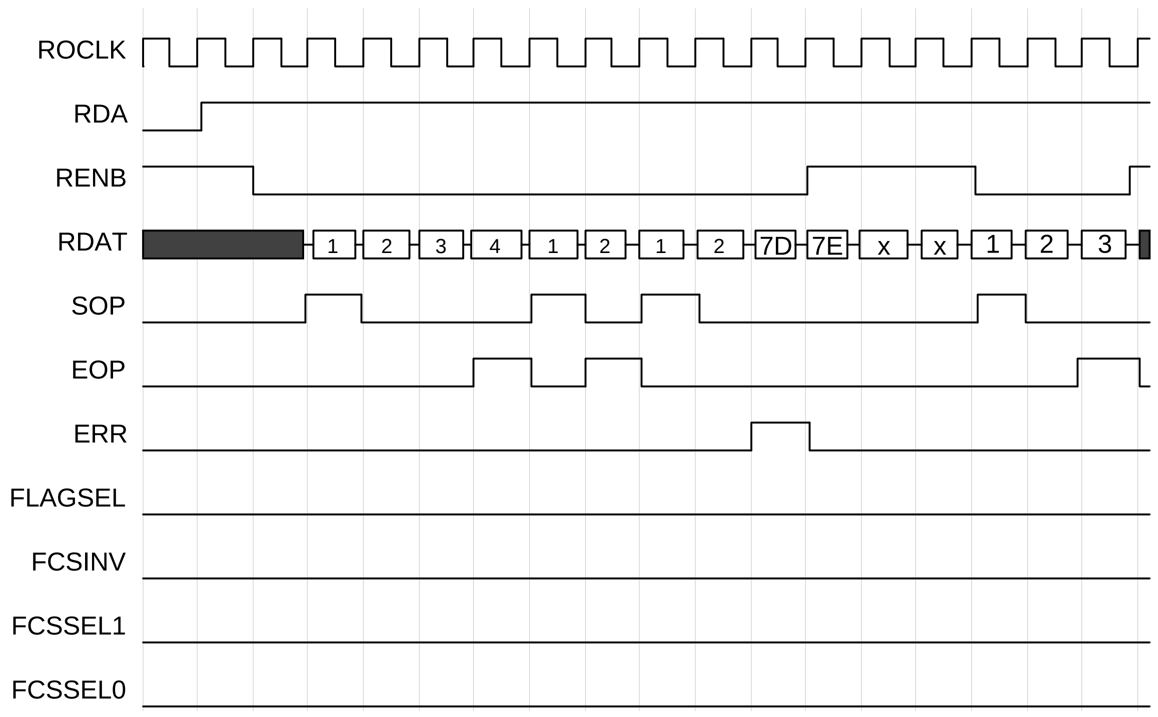
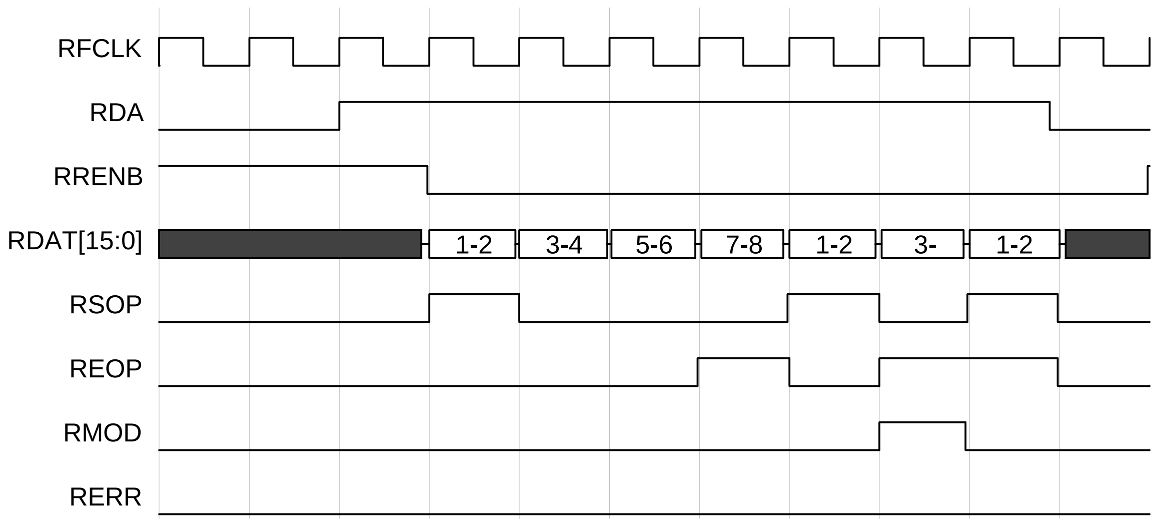


Figure 23 above shows the basic timing between the Receive HDLC Controller and the Receive FIFO.

Figure 24: Receive FIFO – Link Layer Timing



The Receive FIFO timing diagram (Figure 24) illustrates the basic timing of the drop side receive FIFO. The receive FIFO indicate that there is data available by asserting the RDA signal. RDA remains high until the receive FIFO is empty. At anytime, the downstream device can throttle back reception of words by deasserting RRENB.

7 DISCLAIMER

The circuits presented in this application note have not been built or simulated. These circuits are therefore preliminary in this release of this document.

8 REFERENCES

1) PMC-Sierra, Inc., PM5342 SPECTRA-155 Data Sheet, Issue 2, Dec, 1996.
2) Network Working Group Request For Comments: RFC 1619 , May 1994.

NOTES

PM-971133 (R4) ref PMC-xxxxxx (Rx) Issue date: February 1998

CONTACTING PMC-SIERRA, INC.

PMC-Sierra, Inc.
105-8555 Baxter Place Burnaby, BC
Canada V5A 4V7

Tel: (604) 415-6000

Fax: (604) 415-6200

Document Information:	document@pmc-sierra.com
Corporate Information:	info@pmc-sierra.com
Application Information:	apps@pmc-sierra.com
Web Site:	http://www.pmc-sierra.com

PM-971133 (R4) ref PMC-xxxxxx (Rx) Issue date: February 1998