

# **CoreSDLC v3.0**

*Handbook*

## **Actel Corporation, Mountain View, CA 94043**

© 2010 Actel Corporation. All rights reserved.

Printed in the United States of America

Part Number: 50200269-0

Release: July 2010

No part of this document may be copied or reproduced in any form or by any means without prior written consent of Actel.

Actel makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability or fitness for a particular purpose. Information in this document is subject to change without notice. Actel assumes no responsibility for any errors that may appear in this document.

This document contains confidential proprietary information that is not to be disclosed to any unauthorized person without prior written consent of Actel Corporation.

### **Trademarks**

Actel, Actel Fusion, IGLOO, Libero, Pigeon Point, ProASIC, SmartFusion and the associated logos are trademarks or registered trademarks of Actel Corporation. All other trademarks and service marks are the property of their respective owners.

---

# Table of Contents

---

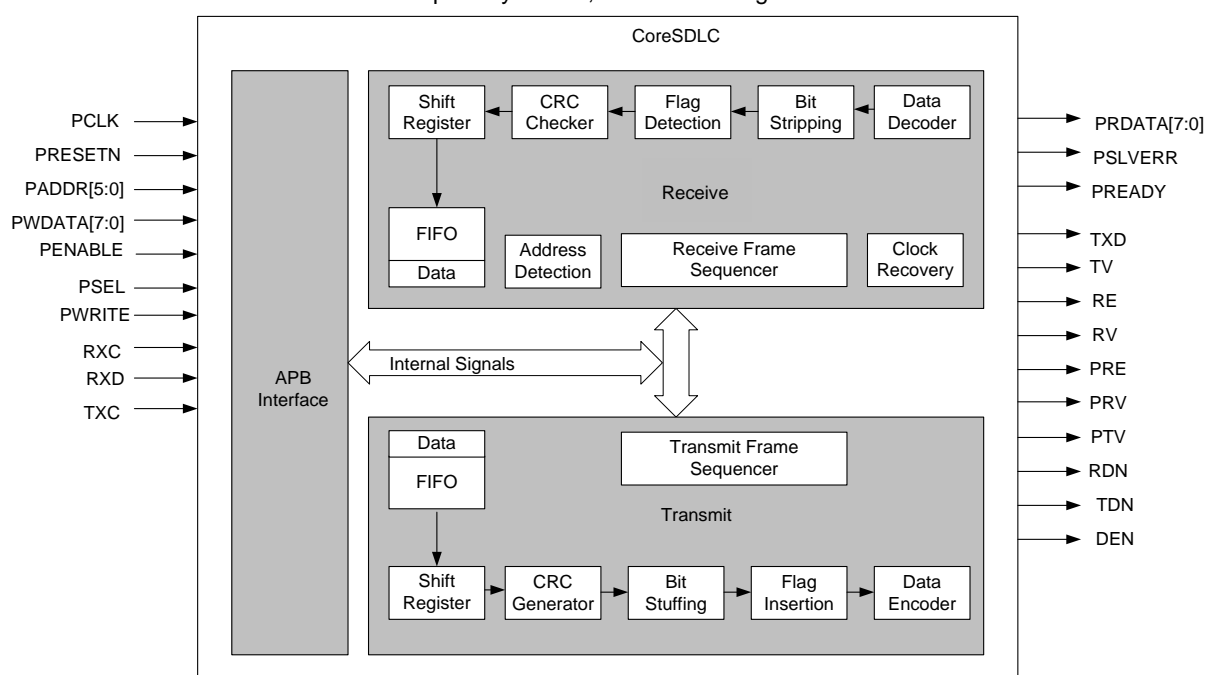
<b>Introduction</b> . . . . .	<b>5</b>
Key Features . . . . .	5
Supported Actel FPGA Families . . . . .	6
Core Version . . . . .	6
Supported Interfaces . . . . .	6
Utilization and Performance . . . . .	6
<b>Design Description</b> . . . . .	<b>9</b>
Verilog/VHDL Parameters . . . . .	9
I/O Signals . . . . .	9
<b>Register Map and Descriptions</b> . . . . .	<b>11</b>
<b>Design Details</b> . . . . .	<b>17</b>
Interface Definitions . . . . .	17
SDLC Protocol Overview . . . . .	17
Modes of Operation . . . . .	21
General Description of the Transmitter . . . . .	23
General Description of Receiver . . . . .	24
<b>Tool Flows</b> . . . . .	<b>29</b>
Licensing . . . . .	29
SmartDesign . . . . .	29
Simulation Flows . . . . .	30
Synthesis in Libero IDE . . . . .	31
Place-and-Route in Libero IDE . . . . .	31
<b>Ordering Information</b> . . . . .	<b>33</b>
Ordering Codes . . . . .	33
<b>Product Support</b> . . . . .	<b>35</b>
Customer Service . . . . .	35
Actel Technical Support . . . . .	35



# Introduction

The CoreSDLC macro provides a high-speed synchronous serial communication controller that utilizes the synchronous data link control (SDLC) protocol. Operation of the controller is similar to that used in the Intel 8XC152 global serial channel (GSC) device working in SDLC mode under CPU control. Communication with a CPU is realized through the Advanced Peripheral Bus (APB) interface and three interrupt sources. This enables interfacing CoreSDLC easily with any APB master.

CoreSDLC consists of three primary blocks, as shown in Figure 1.



**Figure 1** CoreSDLC Block Diagram

## Key Features

- Based on Intel's 8XC152 global serial channel working in SDLC mode
- Single and double-byte address recognition
- Address filtering enables multicast and broadcast addresses
- 16-bit (CRC-16) and 32-bit (CRC-32) frame check sequence
- NRZ and NRZI data encoding
- Automatic bit stuffing/stripping
- 3-byte deep internal receive and transmit FIFOs
- Full or half-duplex operation
- Variable BAUD rate
- External or internal transmit and receive clocks

- Optional preamble generation
- Programmable interframe space
- Raw transmit and receive testing modes
- All major Actel device families supported

## Supported Actel FPGA Families

- IGLOO<sup>®</sup>
- IGLOOe
- IGLOO nano
- IGLOO PLUS
- ProASIC3E
- ProASIC3
- ProASIC<sup>®</sup>3L
- SmartFusion
- Fusion
- ProASIC<sup>PLUS</sup><sup>®</sup>
- Axcelerator<sup>®</sup>
- RTAX-S
- SX-A
- RTSX-SU

## Core Version

This handbook supports CoreSDLC version 3.0.

## Supported Interfaces

CoreSDLC is available with an AMBA 3 APB slave interface.

## Utilization and Performance

CoreSDLC has been implemented in several Actel devices. A summary of CoreSDLC utilization and performance for various devices is listed in Table 1.

**Table 1** CoreSDLC Device Utilization and Performance

Family	Cells or Tiles			Utilization		Performance
	Sequential	Combinatorial	Total	Device	Total	
Fusion	392	786	1178	AFS250	19%	128 MHz
SmartFusion	392	795	1187	A2F200M3F	26%	120 MHz
ProASIC3	392	788	1180	A3P250	19%	131 MHz
ProASIC3E	392	788	1180	A3PE600	9%	124 MHz
ProASIC3L	392	775	1167	A3P250L	19%	97 MHz

IGLOO	392	792	1184	AGL125V2	39%	51 MHz
IGLOOe	392	792	1184	AGLE600V2	9%	54 MHz
IGLOO PLUS	392	810	1202	AGLP125V2	39%	48 MHz
ProASICPLUS	376	1112	1488	APA150	24%	80 MHz
Axcelerator	410	601	1011	AX125	50%	150 MHz
RTAX-S	410	600	1010	RTAX250S	24%	112 MHz
SX-A	405	556	961	A54SX32A	34%	100 MHz
RTSX-SU	405	556	961	RT54SX32S	34%	60 MHz

*Note: Data in this table were achieved using typical synthesis and layout settings.*



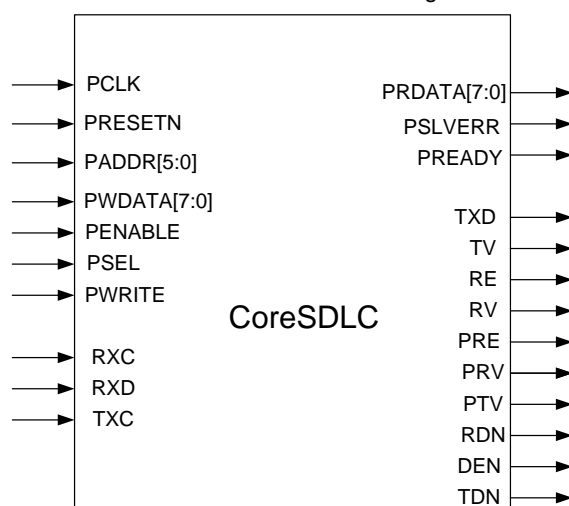
# Design Description

## Verilog/VHDL Parameters

CoreSDLC has no parameters (Verilog) or generics (VHDL).

## I/O Signals

The port signals for the CoreSDLC macro are shown in Figure 2 and defined in Table 2.



**Figure 2** CoreSDLC I/O Signal Diagram

**Table 2** CoreSDLC I/O Signals

Port Name	Type	Description
PCLK	In	APB System clock – Reference clock for all internal logic.
PRESETN	In	APB Active low asynchronous reset.
PWDATA[7:0]	In	APB Write data.
PRDATA[7:0]	Out	APB Read data.
PADDR[5:0]	In	APB Address bus. This port is used to address internal CoreSDLC registers.
PENABLE	In	APB Strobe – indicates the second cycle of an APB transfer.
PSEL	In	APB Slave select – selects CoreSDLC for reads or writes on APB bus.
PWRITE	In	APB Write/Read select signal – If high (logic 1), a write occurs when an APB transfer to CoreSDLC takes place. If low (logic 0), a read from CoreSDLC occurs.
PREADY	Out	APB Ready signal - <i>tied high</i> .
PSLVERR	Out	APB Transfer error signal - <i>tied low</i> .
RV	Out	Receive valid interrupt.

Port Name	Type	Description
RE	Out	Receive error interrupt.
TV	Out	Transmit valid interrupt.
PRV	Out	Receive valid interrupt priority.
PRE	Out	Receive error interrupt priority.
PTV	Out	Transmit valid interrupt priority.
RXD	In	Receive input.
TXD	Out	Transmit output.
RXC	In	Receive clock.
TXC	In	Transmit clock.
DEN	Out	Active-low external driver enable.
RDN	Out	Receive done interrupt.
TDN	Out	Transmit done interrupt.

*Note: All signals are active High (logic 1) unless otherwise noted.*

# Register Map and Descriptions

These sections and Table 4 through Table 25 detail the APB-accessible registers within CoreSDLC.

## Register Summary

Values shown in the tables below are in hexadecimal format; type designations: R = Read only; W = Write only; R/W = Read/Write.

**Table 3** CoreSDLC Internal Register Address Map

Address	Register Name	Type	Width	Reset Value	Description
0x00	GMOD	R/W	8	0x00	GSC Mode
0x04	TFIFO	W	8	0xFF	Transmit FIFO
0x08	PCON	R/W	8	0x00	Power Control
0x0C	BAUD	R/W	8	0x00	Baud Rate
0x10	ADR0	R/W	8	0x00	Address Match 0
0x14	IFS	R/W	8	0x00	Interframe Space
0x18	ADR1	R/W	8	0x00	Address Match 1
0x1C	ADR2	R/W	8	0x00	Address Match 2
0x20	ADR3	R/W	8	0x00	Address Match 3
0x24	IEN1	R/W	8	0xC0	Interrupt Enable
0x28	AMSK0	R/W	8	0x00	Address Mask 0
0x2C	TSTAT	R/W	8	0x04	Transmit Status
0x30	AMSK1	R/W	8	0x00	Address Mask 1
0x34	RSTAT	R/W	8	0x00	Receive Status
0x38	RFIFO	R	8	0xFF	Receive FIFO
0x3C	IPN1	R/W	8	0xC0	Interrupt Priority

## GSC Mode Register (GMOD)

**Table 4** GSC Mode Register

PADDR[6:0]	Register Name	Type	Width	Reset Value	Description
0x00	GMOD	R/W	8	0x00	GSC Mode

**Table 5** GSC Mode Register Bit Functions

Bits	Name	Type	Description
7	xtclk	R/W	External transmit clock 1 – External clock and NRZ encoding used by transmitter 0 – Internal clock generator and NRZI encoding used by transmitter

Bits	Name	Type	Description
6:5	m1 m0	R/W	Mode select 00 – Normal 01 – Raw transmit 10 – Raw receive 11 – Not allowed
4	a1	R/W	Address length 1 – 16-bit addressing is used 0 – 8-bit addressing is used
3	ct	R/W	The CRC type 1 – 32-bit CRC is used 0 – 16-bit CRC (CRC-CCITT) is used
2:1	p1 p0	R/W	Preamble length 00 – 0-bit 01 – 8-bits 10 – 32-bits 11 – 64-bits
0	–	–	Not used

## Transmit FIFO Register (TFIFO)

**Table 6** Transmit FIFO Register

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x04	TFIFO	W	8	0x00	Transmit FIFO

The TFIFO register represents the 3-byte deep transmit FIFO. Writing a byte to this register loads data into the transmit FIFO and automatically updates the FIFO pointers. Setting the 10-bit in the TSTAT register clears the transmit FIFO. The TFIFO is a write-only register from the perspective of the CPU.

## Power Control Register (PCON)

**Table 7** Power Control Register

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x08	PCON	R/W	8	0x00	Power Control

**Table 8** Power Control Register

Bits	Name	Type	Description
7:5	–	–	Not used
4	garen	R/W	Auxiliary receive enable 1 – The reception of back-to-back frames is enabled. The receiver is not disabled after receiving the EOF flag when this bit is set. 0 – Prevents reception of back-to-back frames. The receiver is disabled after receiving the EOF flag.

3	xrcclk	R/W	External receive clock 1 – External clock and NRZ encoding scheme used by receiver 0 – Internal clock generator and NRZI encoding used by receiver
2	gfien	R/W	Flag idle enable 1 – Idle flags (01111110) are generated between transmitted frames representing the sequence 01111110 01111110... 0 – No idle flag generation
1:0	–	–	Not used

*Note:* This register has unimplemented bits (–). Unless otherwise specified, if these bits are read they return 0. Writing to these bits has no effect.

## Baud Rate Register (BAUD)

**Table 9** Band Rate Register

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x0C	BAUD	R/W	8	0x00	BAUD rate

The BAUD register is used to set the value of an internal programmable baud rate generator. The baud rate generator operates by down-counting the BAUD register. When BAUD decrements to an all '0' value, it is reloaded. Writing a value into BAUD stores the value in the reload register. Reading it gives the current count value. The baud rate can only be programmed in multiples of 1/8th of the PCLK input frequency. This is accomplished by entering the appropriate value into the BAUD register, as shown in the following formula:

$$\text{Baud Rate} = \text{PCLK} / ((\text{BAUD register value} + 1) \times 8)$$

For example, if the PCLK input frequency is 20MHz, and the BAUD register is set to 0x01; the baud rate is set to 1.25 Mbps.

## Address Match Registers (ADR0, ADR1, ADR2, ADR3)

**Table 10** Address Match Register (ADR0)

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x10	ADR0	R/W	8	0x00	Address Match 0

**Table 11** Address Match Register (ADR1)

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x18	ADR1	R/W	8	0x00	Address Match 1

**Table 12** Address Match Register (ADR2)

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x18	ADR2	R/W	8	0x00	Address Match 1

**Table 13** Address Match Register (ADR3)

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x20	ADR3	R/W	8	0x00	Address Match 3

The address match registers contain values that are compared with the address in received frames. In an 8-bit addressing mode, an address match occurs when one of these four address values match. In a 16-bit addressing mode, address registers are combined into two 16-bit registers: ADR1:ADR0 and ADR3:ADR2.

An address match occurs when one of these two 16-bit registers trigger a match. Address registers are used only in the receive operation. When CoreSDLC transmits, the frame address is treated as normal data. Therefore, the user's software is responsible for loading the address bytes into the transmit FIFO before other data.

## Interframe Space Register (IFS)

**Table 14** Interframe Space Register

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x14	IFS	R/W	8	0x00	Interframe space

The IFS register determines the minimum number of bit times that must elapse between two consecutive transmitted frames. CoreSDLC only takes the seven most significant bits of the written value (only even numbers can be used) and computes the interframe space by counting this 7-bit number down to an all '0' value twice. When read by the user's software, the seven most significant bits of the IFS register show the current count value, while the least significant bit is a '1' for first counting and a '0' for the second.

## Interrupt Enable Register (IEN1)

**Table 15** Interframe Space Register

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x24	IEN1	R/W	8	0xC0	Interrupt enable

**Table 16** Interframe Space Register Bit Functions

Bits	Register Name	Type	Description
7:4	-	-	Not used (fixed at 1100)
3	egstv	R/W	Transmit valid interrupt enable
2	-	-	Not used
1	egsre	R/W	Receive error interrupt enable
0	egsrv	R/W	Receive valid interrupt enable

*Note:* This register has unimplemented bits (-). Unless otherwise specified, if these bits are read they return '0'. Writing to these bits has no effect.

## Address Mask Registers (AMSK0, AMSK1)

**Table 17** Address Mask Register (AMSK0)

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x28	AMSK0	R/W	8	0x00	Address Mask 0

**Table 18** Address Mask Register (AMSK1)

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x30	AMSK1	R/W	8	0x00	Address Mask 1

Bits in the address mask registers, AMSK1 and AMSK0, correspond to bits in the registers ADR1 and ADR0, respectively. Setting a mask register bit to '1' causes the corresponding bit in the address register to be omitted during the address matching process.

## Transmit Status Register (TSTAT)

**Table 19** Transmit Status Register (TSTAT)

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x2C	TSTAT	R/W	8	0x04	Transmit status

**Table 20** Transmit Status Register Bit Functions

Bits	Name	Type	Description
7	lni	R	Line idle 1 – Receive line is idle (15 consecutive '1' values are received on rxd) 0 – Receive line is not idle
6:4	-	-	Not used
3	tdn	R	Transmit done This bit is set after successful completion of a frame transmission and cleared after setting the ten bit.
2	tfnf	R	Transmit FIFO not full When set indicates that new data may be written into TFIFO.
1	ten	W	Transmit enable Setting this flag clears tdn and TFIFO and enables transmission. This bit is automatically cleared after the end of transmission. If this bit is cleared to a '0' before the end of transmission, that transmission is aborted.
0	-	-	Not used

*Note:* This register has unimplemented bits (-). Unless otherwise noted, if these bits are read they return '0'. Writing to these bits has no effect.

## Receive Status Register (RSTAT)

**Table 21** Receive Status Register (RSTAT)

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x34	RSTAT	R/W	8	0x00	Receive status

**Table 22** Receive Status Register Bit Functions

Bits	Name	Type	Description
7	ovr	R	Overflow error If set, ovr indicates that receive FIFO was full when attempting to store new data. The user can clear the ovr bit by setting the gren bit.
6	rcabt	R	Abort detect If set, rcabt indicates that seven consecutive '1' values were received before the EOF flag but after data had been loaded into RFIFO. The user can clear the rcabt bit by setting the gren bit.
5	ae	R	Alignment error If set, ae indicates that a non byte-aligned flag was received after data had been loaded into RFIFO. The user can clear the ae bit by setting the gren bit.

4	crce	R	CRC error If set, crce indicates that a frame was received with a mismatched CRC. The user can clear the crce bit by setting the gren bit.
3	rdn	R	Receive done When set, rdn indicates successful completion of a frame receive operation. The user can clear the rdn bit by setting the gren bit.
2	rfne	R	Receive FIFO not empty When set, rfne indicates that new data can be read from RFIFO.
1	gren	W	Receive enable Setting this flag enables the receiver and clears the ovr, rcabt, ae, crce and rdn bits. This bit is automatically cleared after the end of the receive operation.
0	–	–	Not used

*Note:* This register has unimplemented bits (-). Unless otherwise specified, if these bits are read they return '0'. Writing to these bits has no effect. All bits of the RSTAT register are read only, except for bit 1, which is read/write.

## Receive FIFO Registers (RFIFO)

**Table 23** Receive FIFO Register

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x38	RFIFO	R	8	0x00	Receive FIFO

The RFIFO register represents the 3-byte deep receive FIFO. Reading a byte from RFIFO loads a value from the receive FIFO and automatically updates the FIFO pointers. Setting the gren bit in the RSTAT register clears the receive FIFO. RFIFO is a read-only register from the perspective of the CPU.

## Interrupt Priority Register (IPN1)

**Table 24** Interrupt Priority Register (IPN1)

PADDR[6:0]	Register Name	Type	Width	Reset Value	Brief Description
0x3C	IPN1	R/W	8	0xC0	Interrupt priority

**Table 25** Interrupt Priority Register Bit Functions

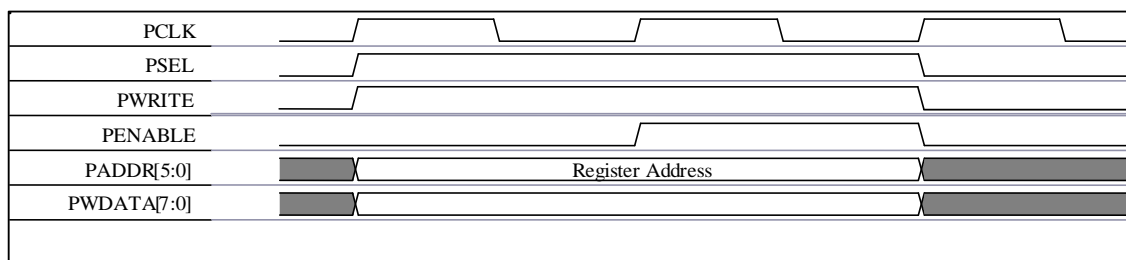
Bits	Register Name	Type	Description
7:4	–	–	Not used (fixed at 1100)
3	pgstv	R/W	Transmit valid interrupt priority
2	–	–	Not used
1	pgsrv	R/W	Receive valid interrupt priority
0	pgsre	R/W	Receive error interrupt priority

*Note:* This register has unimplemented bits (-). Unless otherwise specified, if these bits are read they return '0'. Writing to these bits has no effect.

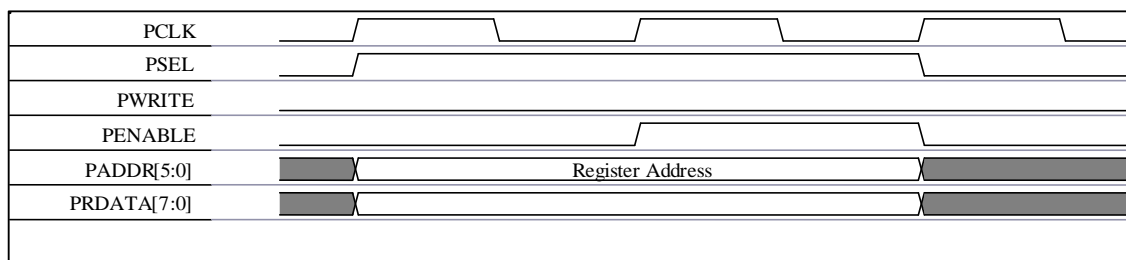
# Design Details

## Interface Definitions

The CoreSDLC APB slave interface conforms to the standard AMBA 3 APB specifications. Figure 3 and Figure 4 depict typical write cycle and read cycle timing relationships relative to the system clock.



**Figure 3** APB Data Write Cycle



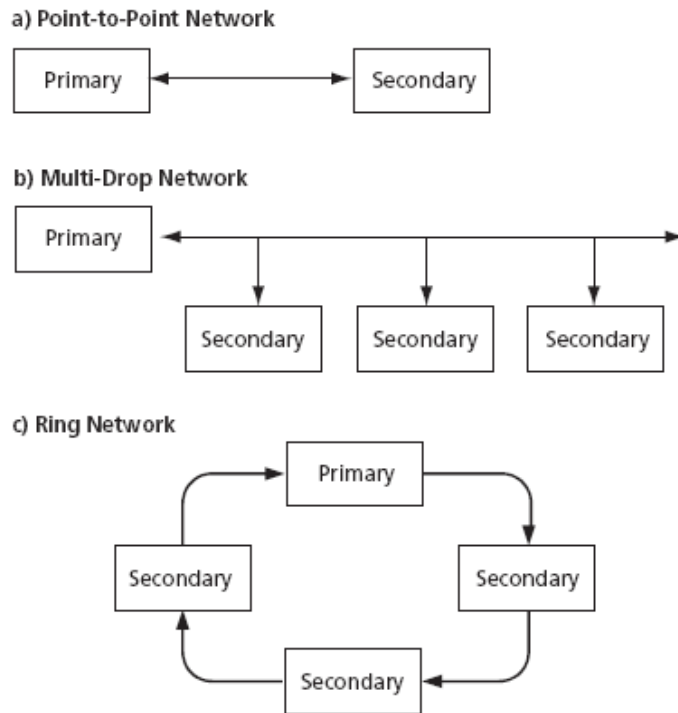
**Figure 4** APB Data Read Cycle

## SDLC Protocol Overview

The SDLC protocol has two types of network nodes: Primary and Secondary. There is always one primary node in the network, but there may be one or more secondary nodes. The primary node controls operation of the secondary nodes and manages the network. Secondary nodes can send information only if the primary node has given them permission. This is accomplished by the primary node polling the secondary nodes in a predetermined order to see if they need to send information.

As shown in Figure 5, SDLC nodes are connected in one of the three following configurations:

- Point-to-point, when there is one primary and only one secondary node.
- Multi-drop, when there is one primary and multiple secondary nodes.
- Ring, when all nodes are connected in a loop and the output channel of one node is connected to the input channel of the next node.


**Figure 5** SDLC Network Configurations

## SDLC Frames

The SDLC frame consists of six fields. Table 26 shows the order of the fields in the SDLC frame.

**Table 26** SDLC Frame

BOF	ADDRESS	CONTROL	INFO	CRC	EOF
-----	---------	---------	------	-----	-----

### BOF (Beginning of Frame)

The BOF flag, which indicates the beginning of a frame, is assigned the value 01111110. The controller's hardware properly distinguishes normal data from the BOF flag because of a process called bit stuffing, which is described in 'Bit Stuffing'. Bit stuffing, performed by the transmit logic, is the process of inserting a '0' after each five consecutive '1' values. The receiver logic utilizes a process called bit stripping. Each time a sequence of five '1' values followed by a '0' is received, the controller automatically removes this '0' from the incoming bit stream. BOF is one of two possible bit combinations that consist of more than five consecutive '1' values. The BOF marks the beginning of a frame and also assures receive clock synchronization.

### ADDRESS

In standard SDLC, an 8-bit field in the frame is used to identify the target controller for which the frame is intended. In CoreSDLC, this field may also be 16 bits in length, extending the addressing capability. The address length can be further extended by the user's software; however, the hardware addresses checking only works up to 16-bit addresses. There is also one special address defined in SDLC called "broadcast address," consisting of all '1' values. All stations connected to the network receive the frame containing the broadcast address. CoreSDLC transmits the address field's least significant bit (LSB) first.

## CONTROL

This field is used for initializing the system and managing tasks, such as data acknowledge, identifying frame sequence numbers, and indicating the end of the message. CoreSDLC does not provide any functions for managing the CONTROL field, so the user's software is responsible for insertion and interpretation. There are three types of control fields, depending on the type of SDLC frame used:

- Information frame (Table 27)
- Supervisory frame (Table 28)
- Non-sequenced (or unnumbered) frame (Table 29)

**Table 27** Control Field - Information Format

Bit Position	7	6	5	4	3	2	1	0
Function	Reception Sequence			Poll / Final	Sending sequence			1

**Table 28** Control Field – Supervisory Format

Bit Position	7	6	5	4	3	2	1	0
Function	Reception Sequence			Poll / Final	Mode		0	1

**Table 29** Control Field – Non-sequenced Format

Bit Position	7	6	5	4	3	2	1	0
Function	Command / Response			Poll / Final	Command / Response		0	1

The CONTROL field of the information frame contains a 3-bit sending sequence number (the number of the current frame) and a 3-bit reception sequence number (the expected number of the next frame). The same reception sequence number is also part of the control field in the supervisory frame. In both cases, it is used for frame acknowledgement. If the receiving station has no data to send, it acknowledges the received frames by sending a supervisory frame in response. However, if the receiving station wants to send data, the response may be part of the information frame (piggybacking). This allows for full-duplex operation in which two continuous data streams are transmitted in both directions without supervisory frame insertion. Up to seven information frames can be sent without acknowledgement. Due to this capability, continuous transmission (continuous ARQ) is possible, which means that the CoreSDLC transmitter does not need to wait for an acknowledgement. The poll/final bit in each control field is used for polling secondary nodes by a primary node (poll) and for indicating the end of the message (final). The supervisory frame contains an additional two mode bits, which affect the retransmission scheme. Although it is possible for four modes to exist, three of them are used in CoreSDLC:

- Receiver Ready (RR) – Indicates that the receive line of this station is ready to accept frames.
- Receiver Not Ready (RNR) – Indicates that the receiver is not ready to accept frames (possible FIFO overflow).
- Reject (REJ) – Indicates that the previously received frame was rejected.

The unnumbered (non-sequential) frame contains 5 bits that indicate commands or responses used for initializing the network and eliminating errors. These commands include:

- Unnumbered information (UI)
- Set initialization mode (SIM)
- Disconnect (DISC)
- Response optional (UP)
- Function descriptor in information field (CFGR)

- Identification in information field (XID)
- Test pattern in information field (TEST)
- Request for initialization (RIM)
- Frame reject (FRMR)
- Unnumbered acknowledgment (UA)
- Signal loss of input (BCN)
- Station wants to disconnect (RD)
- Station in disconnected mode (DM)

### INFO

This field contains data that is transmitted by one device to the other. It can be an arbitrary length, although it must be byte-aligned (its length must be a multiple of 8 bits). It is possible that some frames may not contain an INFO field. The INFO field is transmitted LSB first.

### CRC (Cyclic Redundancy Check)

This is the error checking sequence. It is a widely used method for detecting errors in messages transmitted over noisy channels. CoreSDLC offers two types of CRC: 16-bit CRC (often referred to as CRC-CCITT) and 32-bit CRC. CRC is generated in the transmitter over the ADDRESS, CONTROL, and INFO fields before the bit-stuffing process. First, the CRC shift register is preset with an all '1' value. For each incoming data bit, the most significant bit (MSB) of the current CRC remainder is XORed with the data bit, and the remainder is shifted left with the LSB set to '0'. If the result of the XOR is '1', the remainder is XORed with the generator polynomial. For the 16-bit CRC, the polynomial is:

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

For the 32-bit CRC, the polynomial is:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

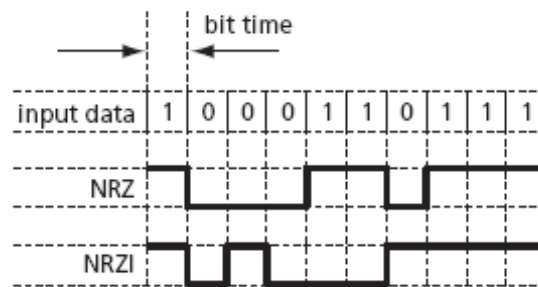
After the last data bit has passed through the CRC generator, the current CRC generator value is inverted and sent to the receiver, MSB first. The receiver operates exactly the same way as the transmitter by generating the CRC remainder over the incoming ADDRESS, CONTROL, INFO, and CRC fields. After all data has passed through, the current CRC remainder is checked. If no errors have occurred, this remainder is equal to the polynomial residual. For a 16-bit CRC, this residual is 00011101 00001111 (0x1D0F). For a 32-bit CRC, it is 11000111 00000100 11011101 01111011 (0xC704DD7B). The CRC field is transmitted MSB first.

### EOF (End of Frame)

The EOF flag consists of the same bit pattern as the BOF flag (01111110). EOF indicates when the transmission is completed and also can serve as the BOF for the next frame. Frames that share EOF and BOF flags are called back-to-back frames.

## Data Encoding

CoreSDLC employs NRZI (non-return-to-zero inverted) data encoding when transmitting frames. In NRZI, a '1' is represented by no change in the output signal level, and a '0' is represented by a change in the level. A data stream consisting of all '0' values causes the NRZI output to toggle each bit time, while a stream of all '1' causes no transitions. Although NRZI is typical for SDLC networks, CoreSDLC also performs NRZ (non-return-to-zero) encoding. In this encoding method, a '1' is represented by a high level and a '0' by a low level. Figure 6 shows an example of NRZ and NRZI encoding.

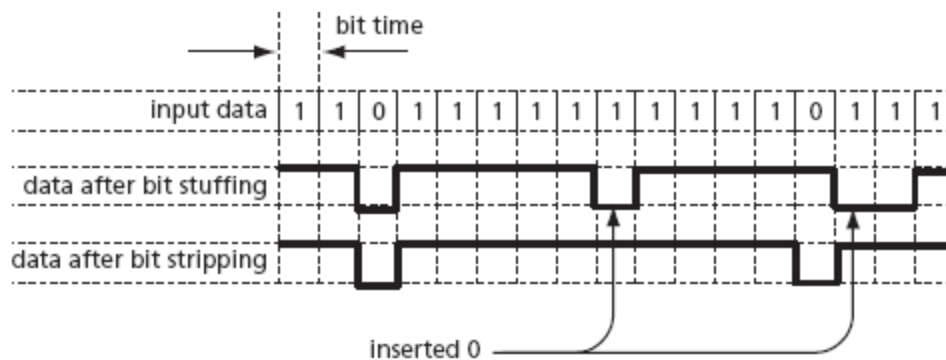


**Figure 6** NRZI and NRZ Data Encoding

## Bit Stuffing

CoreSDLC employs bit stuffing to ensure the minimum number of signal transitions that are necessary for “recovering” the receiver clock. Bit stuffing is the process of inserting a ‘0’ after every five consecutive ‘1’ values in a data stream to force a transition in a NRZI output data stream. This guarantees that there is at least one transition every 6 bit times while transmitting data. The receiver must recognize the inserted bits and remove them from the data stream.

This process is called bit stripping. Bit stuffing and stripping are automatically performed by CoreSDLC and are completely transparent to the user. Figure 7 shows an example of bit stuffing and bit stripping.



**Figure 7** Bit Stuffing / Stripping

## Modes of Operation

CoreSDLC provides three modes of operation: normal mode, raw transmit mode, and raw receive mode. The first is normally used in standard communications within SDLC networks, while the others may be used for testing the controller’s operation or transmit user data, (not necessarily SDLC- formatted). Information about all operational modes, as well as a summary of options available for each mode, are described in detail in the following text and Table 30. Each of the three modes of operation is selected by the CPU setting the m1 and m0 bits in the GMOD register, as listed in Table 5.

### Normal

In normal mode, data is transmitted in standard SDLC format. After transmission is enabled by the CPU setting the ten bit in the TSTAT register and loading the transmit FIFO with data, CoreSDLC tests if the interframe space (time from previous transmission) has expired. If this condition is met, the den (external driver enable) output is forced low. One bit time later; CoreSDLC begins transmission by sending the

appropriate number of preamble bits and the BOF flag. Immediately after BOF is sent, a byte from TFIFO is loaded into the shift register. As bits are shifted out of this register, they also pass through the CRC generator, updating the current CRC value. This process is performed as long as the transmit FIFO contains data. If the FIFO is empty when the transmitter is about to load the next byte, CoreSDLC assumes "end of data." Transmission ends with sending the current CRC generator value followed by the EOF flag. All transmitted bits are encoded with NRZI (if the internal clock generator is selected) or NRZ (if external clock input is selected). The receiver working in normal mode searches the input for the BOF flag. Immediately after BOF is detected, the frame's address field is checked if it matches the address assigned with address registers addr3-0. When the address does not match, that frame is ignored. If the address matches, the receiver loads incoming bits including ADDRESS, CONTROL, and INFO fields into the shift register and then into the receive FIFO. The CRC is not loaded into the receive FIFO.

## Raw Transmit Mode

In raw transmit mode, the transmit output is internally connected to the receive input. All data written to the transmit FIFO are transmitted without preamble, BOF and EOF flags, address, and CRC. Additionally, bit stuffing is disabled. The receiver operates as normal in this mode. Raw transmit mode can be used for receiver testing or for transmitting user data (not necessarily SDLC formatted).

## Raw Receive Mode

In raw receive mode, the transmitter operates as in normal mode. The receiver also operates as normal except that all bytes between the BOF and EOF flags are loaded into the receive FIFO, including the CRC field. The receiver does not check the CRC and no CRC error is set. In addition, address matching is not performed, and therefore, all frames are received. To use raw receive as a test mode, the transmit output should be externally connected to the receive input. This allows most of the transmitter functions, as well as the external transceiver, to be checked.

**Table 30** Functions Available in Individual Modes

Mode	Normal (m1 = 0, m0 = 0)		Raw Transmit (m1 = 0, m0 = 1)		Raw Receive (m1 = 1, m0 = 0)	
	Transmit	Receive	Transmit	Receive	Transmit	Receive
Preamble	O	NA	N	NA	O	NA
BOF and EOF	Y	Y	N	Y	Y	Y
Address matching	NA	Y	NA	Y	NA	N
CRC check	Y	Y	N	Y	Y	N
Bit Stiffing and Stripping	Y	Y	N	Y	Y	Y
NRZ/NRZI	Y	Y	Y	Y	Y	Y

*Notes:*

1. m1, m0 – GMOD register mode bits in Table 5.
2. Y – Used
3. N – Not used
4. O – Optional
5. NA – Not applicable

## General Description of the Transmitter

### Interframe Spacing

Interframe space is a period of time that must elapse between two consecutive transmissions. It is measured in bit times. Interframe space can be set by writing an appropriate value (number of bit times) into the IFS register. Note that only even numbers can be used (the LSB must always be set to '0'), because only the seven most significant bits are loaded into the IFS register. This means that interframe space can be set from two bit times to 256 bit times. A value of 0x02 written into IFS corresponds to two bit times; 0xFE corresponds to 254 bit times, while 0x00 corresponds to 256 bit times.

### Preamble

The preamble is a series of toggling '1' and '0' values. The length of preamble can be set to 0, 8, 32 or 64 bits by writing appropriate values into pl1 and pl0 bits in the GMOD register. The preamble is not defined in the standard SDLC protocol and thus it is not considered part of the SDLC frame. The purpose of the preamble is only for synchronization between stations in the network. Note that if idle flags are used in conjunction with a preamble, the addresses 0x00 and 0x55 should not be assigned to the controller. Otherwise, a preamble following the idle flags is interpreted as a matching address.

### Sending an Abort Flag

An abort flag is the sequence of seven or more '1' values. If the receiver detects an abort flag between EOF and BOF, it immediately ends reception. There are three ways to generate an abort flag using CoreSDLC:

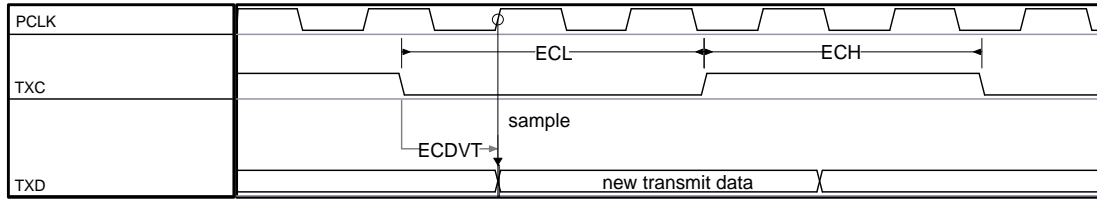
- One method is to clear the ten bit in the TSTAT register and wait at least seven bit times. In this case, the delay necessary to transmit seven bits must be measured by the user's software.
- The second method is based on programmable interframe space. The first step is to write into the IFS register, a value greater than or equal to eight. Then the user's software must clear the ten bit, which disables transmission and forces the output to a high level. With this method, the ten bit can be immediately re-enabled. The output remains at a high level until the interframe space expires, which is accomplished automatically by CoreSDLC.
- The third method is to use raw transmit mode. Writing a value of 0xFF into TFIFO generates a high output for eight bit times. This is possible because the transmitter does not use bit stuffing in raw transmit mode.

### External Driver Interface

CoreSDLC uses the DEN output for an external driver interface. This output is activated one bit time before transmission begins and remains active until the last bit of the EOF is transmitted, or until the ten bit is cleared by the user's software.

### Transmission with an External Clock

An external clock is selected by setting the xtclk bit in the GMOD register. When this bit is set, NRZI decoding is also disabled. Data is transmitted with the NRZ decoding scheme on the falling edge of the TXC clock. Due to the TXC input synchronization with the global clock, PCLK, there can be a delay of up to two clock periods until the data begins to transmit, as shown in Figure 8 and Table 31.


**Figure 8** External Transmit Clock Timing

**Table 31** External Transmit Clock Timing

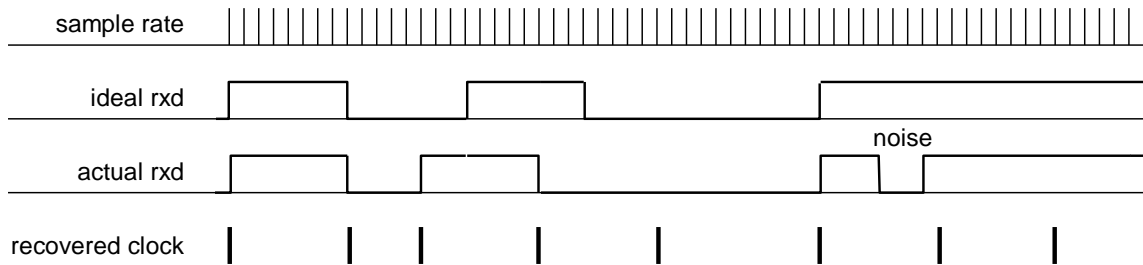
Symbol	Parameter	Minimum Value
ECL	External clock low	$2 \times (\text{period of PCLK})$
ECH	External clock high	$2 \times (\text{period of PCLK})$
ECDVT	External clock to data valid transmit	$1 - 2 \times (\text{period of PCLK})$

## General Description of Receiver

### Receive Clock Recovery

In synchronous serial protocols like SDLC, data and clock are both transmitted over the same medium. Receiver clock recovery is the process of separating the clock signal from the incoming data bit stream (Figure 9). The receiver performs this action. In CoreSDLC, the receiver input is always monitored at eight times the baud rate frequency and searched for the level transitions. Every transition causes the receiver to correct its own clock for proper synchronization with the transmitter. Additionally, CoreSDLC performs digital filtering by ignoring input pulses shorter than four baud rate periods.

The only exception to this rule is when the `xrclk` bit in the `PCON` register is set. In that case, an external receive clock and NRZ decoding are used. Receive data input is then sampled on the rising edge. No clock recovery and no digital filtering are performed in that case.


**Figure 9** Receive Clock Recovery

## Receive Error Conditions

CoreSDLC detects four kinds of receive errors represented by bits in the RSTAT register (Table 32):

- crce – CRC error
- ae – alignment error
- rcabt – receive abort
- ovr – overrun in receive FIFO

The user's software can read these bits, but only CoreSDLC can write them in response to the various error conditions that they represent. When an error occurs, CoreSDLC sets one or more of these bits and also clears the gren bit in the RSTAT register. When the user's software sets the gren bit re-enabling the receiver, all error bits are cleared. This is the only method for clearing error bits.

It is possible that multiple error bits get set in response to certain errors:

- rcabt and ae can be set when receiving misaligned abort flag
- ovr and crce can be set when an overrun error is forced
- ae and crce can be set when an alignment error occurs

In order to determine the correct cause of the receiver error, the user's software should poll error bits in the following sequence:

1. rcabt
2. ovr
3. crce
4. ae

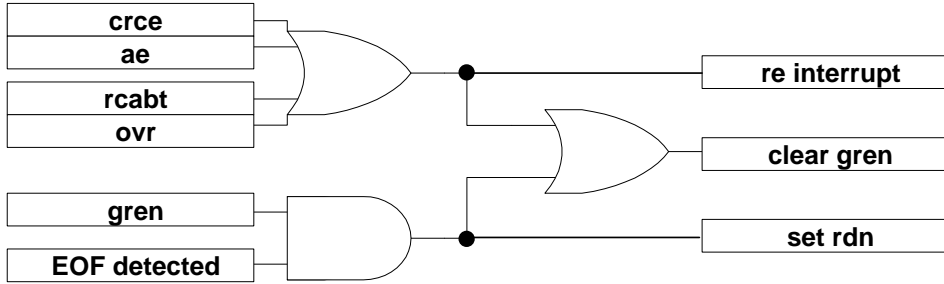
**Table 32** Receive Error Conditions

Error bit	Condition
crce	This bit is set if the CRC remainder after passing all ADDRESS, CONTROL, INFO and CRC fields through the CRC generator is not equal to the polynomial residual. For a 16-bit CRC, this residual is 00011101 00001111, and for a 32-bit CRC it is 11000111 00000100 11011010 01111011. This bit is also set when alignment or overrun errors occur.
ae	This bit is set if the number of bits received between BOF and EOF flags are not a multiple of eight (INFO field is not byte-aligned). This bit is also set when the abort flag is detected.
rcabt	This bit is set if the receiver detects an abort sequence (7 or more consecutive '1' values) in an incoming frame between the BOF and EOF flags but after the first received data has already passed to the receive FIFO. If the abort flag is detected before loading the first byte into the FIFO, the incoming frame is ignored and no error bits are set.
ovr	This flag is set if the receiver gets new data but receive fifo is already full.

## Receive Enable Bits

There are two receive enable bits: gren (receive enable, in the RSTAT register) and garen (auxiliary receive enable, in the PCON register). In order to enable the receiver, at least one of these bits should be set. Although setting only the garen bit enables the receiver, the rdn (receive done, in the RSTAT register) bit, which indicates the end of a valid reception, is set only if the gren bit is set (Figure 10). When the frame reception is in process, clearing both gren and garen causes the receiver to end reception. There is a 0 to 1-bit time delay between clearing the receive enable bits and end of reception.

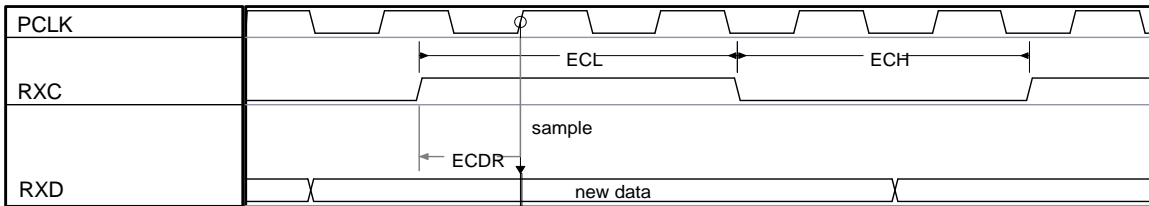
**Note:** crce, ae, rcabt, ovr and gren are RSTAT register bits. The re interrupt can be monitored via the RE output port. The "clear gren" and "set rdn" actions are both taken automatically by CoreSDLC.



**Figure 10** Receive Error Logic

### Receive with External Clock

The external clock is selected by setting the xrclk bit in the PCON register. When this bit is set, NRZI encoding is also disabled. Data is received with NRZ decoding on the RXC clock rising edge. Due to RXC input synchronization with the global clock, PCLK, as shown in Figure 11 and listed in Table 33, there is a maximum of one clock period delay from RXC rising edge to receive data (ECDR).



**Figure 11** External Receive Clock Timing

**Table 33** External Receive Clock Timing

Error bit	Condition	Minimum Value
ECL	External clock low	2*(period of PCLK)
ECH	External clock high	2*(period of PCLK)
ECDR	External clock to receive data sample	0 to 1*(period of PCLK)

### Interrupt Structure

There are three interrupt sources in CoreSDLC - transmit valid, receive valid, and receive error interrupt (Table 34). The transmit valid interrupt flag is set when tfnf (transmit FIFO not full) is set. End of frame transmission is not indicated by an interrupt, so the user must poll the tdn (transmit done) bit in order to know if transmission has ended. The receive valid interrupt flag is set when rfne (receive FIFO not empty) is set. End of frame reception is not indicated by an interrupt, so the user must poll the rdn (receive done) bit in order to know if reception has ended. The receive error interrupt is set in response to a receive error indicated by any of the error bits in the RSTAT register. These error bits are crce, ae, rcabt, and ovr.

**Table 34** Interrupt Summary

Name	Output	Enable bit	Priority bit	Condition
Transmit Valid	TV	egstv	pgstv	This flag is set when tfnf is set (transmit FIFO is not full). Setting the ten bit by the user's software clears this flag.
Receive Valid	RV	egsvr	egstv	This flag is set when rfne is set (receive fifo is not empty). Setting gren bit by user software clears this flag.
Receive Error	RE	egsre	pgstv	This flag is set when at least one of error bits (crce, ae, rcabt, ovr) is set. Setting gren bit by user software clears this flag.
Receive Done	RDN	-	-	This flag is set after end of frame reception. Setting gren bit by user software clears this flag.

## Clock and Reset Control

CoreSDLC is fully synchronous with respect to the global clock PCLK. In other words, there is only one clock domain in the core. All internal registers operate synchronous to the rising edge of PCLK. All input signals (except the reset signal PRESETN) including TXC and RXC, are sampled with the rising edge of PCLK. The PRESETN input signal is asynchronous with respect to the global clock PCLK. For proper operation, PRESETN should be active for at least one global clock period. When PRESETN is active, all registers return to their default states.



# Tool Flows

---

## Licensing

CoreSDLC is licensed in two ways. Depending on your license tool flow, functionality may be limited.

### Obfuscated

Complete RTL code is provided for the core, allowing the core to be instantiated with SmartDesign. Simulation, Synthesis, and Layout can be performed within Libero<sup>®</sup> Integrated Design Environment (IDE). The RTL code for the core is obfuscated<sup>1</sup> and some of the testbench source files are not provided; they are precompiled into the compiled simulation library instead.

### RTL

Complete RTL source code is provided for the core and testbenches.

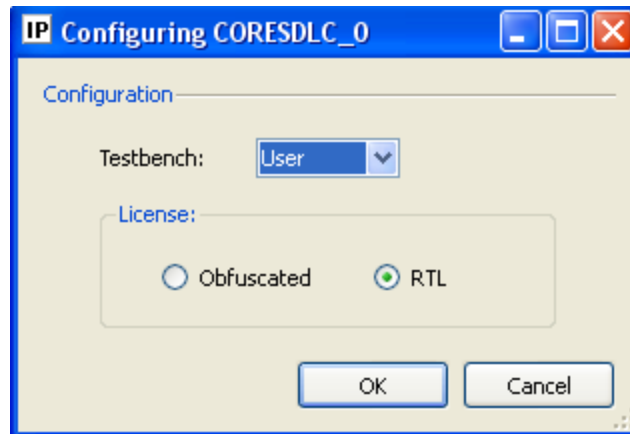
## SmartDesign

CoreSDLC is preinstalled in the SmartDesign IP deployment design environment. The core can be configured using the configuration GUI within SmartDesign, as shown in Figure 12.

For more information on using SmartDesign to instantiate and generate cores, refer to the [Using DirectCore in Libero<sup>®</sup> IDE User's Guide](#).

---

<sup>1</sup> Obfuscated means the RTL source files have had formatting and comments removed, and all instance and net names have been replaced with random character sequences.



**Figure 12** SmartDesign CoreSDLC Configuration Window

## Simulation Flows

The user testbench for CoreSDLC is included in all releases.

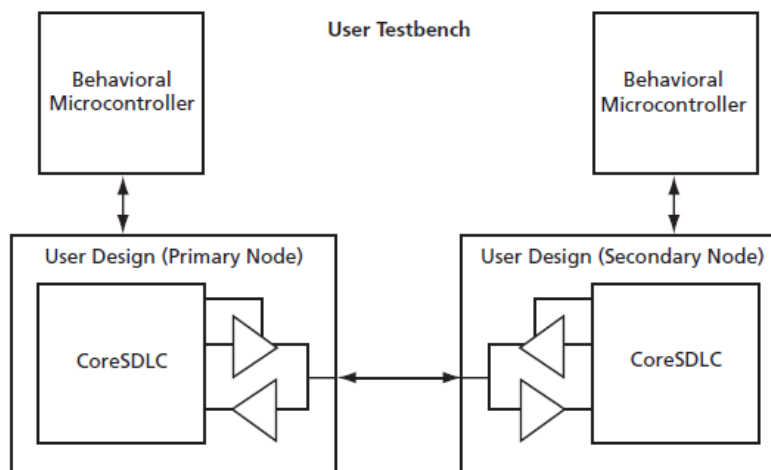
To run simulations, select the user testbench flow within SmartDesign and click Save & Generate on the Generate pane. The user testbench is selected through the Core Testbench Configuration GUI.

When SmartDesign generates the Libero IDE project, it installs the user testbench files.

To run the user testbench, set the design root to the CoreSDLC instantiation in the Libero IDE Design Hierarchy pane and click the Simulation icon in the Libero IDE Design Flow window. This invokes ModelSim® and automatically runs the simulation.

## User Testbench

An example user testbench is included with the obfuscated and RTL releases of CoreSDLC. The user testbench is provided in precompiled ModelSim format for the evaluation release. The obfuscated and RTL releases provide the precompiled ModelSim format, as well as the source code for the user testbench to ease the process of integrating and verifying the CoreSDLC macro into a design. A block diagram of the example user design and testbench is shown in Figure 13.



**Figure 13** Example User Design and User Testbench

The user testbench includes a simple example design that serves as a reference for users who want to implement their own designs. RTL source code for the example design and user testbench (Figure 13) is included in the source directory for the obfuscated and RTL releases of CoreSDLC. As shown in Figure 13, two instantiations of the CoreSDLC macro are connected together in a primary/secondary node connection. Transmits and receive frames pass between the two CoreSDLC nodes, as demonstrated by the user testbench, so that you can gain a basic understanding of how to use this core. The source code for the user testbench contains example support routines to aid in testing an embedded system containing the CoreSDLC macro.

## Synthesis in Libero IDE

Click the **Synthesis** icon in Libero IDE. The Synthesis window appears, displaying the Synplicity® project. Set Synplicity to use the Verilog 2001 standard if Verilog is being used. To run Synthesis, select the **Run** icon.

## Place-and-Route in Libero IDE

Click the **Layout** icon in Libero IDE to invoke Designer. CoreSDLC requires no special place-and-route settings.



# Ordering Information

---

## Ordering Codes

CoreSDLC can be ordered through your local Actel Sales Representative. It should be ordered using the following number scheme: CoreSDLC-XX, where XX is listed in Table 35.

**Table 35** Ordering Codes

Ordering Code	Description
OM	RTL for Obfuscated RTL—multiple use license
RM	RTL for RTL source — multiple-use license



---

# Product Support

---

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

## Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**

From Southeast and Southwest U.S.A., call **650. 318.4480**

From South Central U.S.A., call **650.318.4434**

From Northwest U.S.A., call **650.318.4434**

From Canada, call **650.318.4480**

From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**

From Japan, call **650.318.4743**

From the rest of the world, call **650.318.4743**

Fax, from anywhere in the world **650. 318.8044**

## Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

## Actel Technical Support

Visit the [Actel Customer Support website \(http://www.actel.com/support/search/default.aspx\)](http://www.actel.com/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

## Website

You can browse a variety of technical and non-technical information on Actel's [home page](http://www.actel.com/), at <http://www.actel.com/>.

## Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

### Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure

to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is [tech@actel.com](mailto:tech@actel.com).

### Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

**650.318.4460**

**800.262.1060**

Customers needing assistance outside the US time zones can either contact technical support via email ([tech@actel.com](mailto:tech@actel.com)) or contact a local sales office. [Sales office listings](#) can be found at [www.actel.com/company/contact/default.aspx](http://www.actel.com/company/contact/default.aspx).





***Actel is the leader in low power FPGAs and mixed signal FPGAs and offers the most comprehensive portfolio of system and power management solutions. Power Matters. Learn more at <http://www.actel.com>.***

**Actel Corporation** • 2061 Stierlin Court • Mountain View, CA 94043 • USA

Phone 650.318.4200 • Fax 650.318.4600 • Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

**Actel Europe Ltd.** • River Court, Meadows Business Park • Station Approach, Blackwater • Camberley Surrey GU17 9AB • United Kingdom  
Phone +44 (0) 1276 609 300 • Fax +44 (0) 1276 607 540

**Actel Japan** • EXOS Ebisu Building 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan

Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • <http://jp.actel.com>

**Actel Hong Kong** • Room 2107, China Resources Building • 26 Harbour Road • Wanchai • Hong Kong

Phone +852 2185 6460 • Fax +852 2185 6488 • [www.actel.com.cn](http://www.actel.com.cn)