

# **BDLC Block Guide V01.03**

**Original Release Date: 19 JAN 2001  
Revised: July 19, 2001**

**Motorola, Inc**

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.

## Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
V01.00	04/20/2001	01/19/2001		Original Release
V01.01	06/12/2001	06/13/2001		Corrected formal issues w/paragraph formats, cross references and master pages.
V01.02	06/13/2001	06/14/2001		Removed references to internal signals Moved initialization chap. in 'RESETS' to functional descrip Removed redundant references in the Interrupts section Updated the Interrupts table Changed some explanations to bullets
V01.03	07/19/2001			Document names have been added Names and Variable defenitions have been hidden

# Table of Contents

## Section 1 Introduction

1.1	Overview . . . . .	11
1.2	Features . . . . .	11
1.3	Modes of Operation . . . . .	11
1.4	Block Diagram . . . . .	16

## Section 2 Signal Description

2.1	Overview . . . . .	19
2.2	Detailed Signal Descriptions . . . . .	19
2.2.1	TXB - BDLC Transmit Pin . . . . .	19
2.2.2	RXB - BDLC Receive Pin . . . . .	19

## Section 3 Memory Map and Registers

3.1	Overview . . . . .	21
3.2	Module Memory Map . . . . .	21
3.3	Register Descriptions . . . . .	21
3.3.1	BDLC Control Register 1 (DLCBCR1) . . . . .	21
3.3.2	BDLC State Vector Register (DLCBSVR) . . . . .	23
3.3.3	BDLC Control Register 2 (DLCBCR2) . . . . .	25
3.3.4	BDLC Data Register (DLCBDR) . . . . .	31
3.3.5	BDLC Analog Round Trip Delay Register (DLCBARD) . . . . .	32
3.3.6	BDLC Rate Select Register (DLCBRSR) . . . . .	33
3.3.7	BDLC Control Register (DLCSCR) . . . . .	35
3.3.8	BDLC Status Register (DLCBSTAT) . . . . .	35

## Section 4 Functional Description

4.1	General . . . . .	37
4.1.1	J1850 Frame Format . . . . .	37
4.1.2	J1850 VPW Symbols . . . . .	39
4.1.3	J1850 VPW Valid/Invalid Bits & Symbols . . . . .	41
4.1.4	J1850 Bus Errors . . . . .	50
4.2	Mux Interface . . . . .	53
4.2.1	Mux Interface - Rx Digital Filter . . . . .	53

4.3	Protocol Handler . . . . .	55
4.3.1	Protocol Architecture . . . . .	55
4.4	Transmitting A Message . . . . .	58
4.4.1	BDLC Transmission Control Bits . . . . .	58
4.4.2	Transmitting Exceptions . . . . .	60
4.4.3	Aborting a Transmission . . . . .	61
4.5	Receiving A Message . . . . .	62
4.5.1	BDLC Reception Control Bits . . . . .	63
4.5.2	Receiving a Message with the BDLC module . . . . .	63
4.5.3	Filtering Received Messages . . . . .	64
4.5.4	Receiving Exceptions . . . . .	64
4.6	Transmitting An In-Frame Response (IFR) . . . . .	67
4.6.1	IFR Types Supported by the BDLC module . . . . .	67
4.6.2	BDLC IFR Transmit Control Bits . . . . .	68
4.6.3	Transmit Single Byte IFR . . . . .	69
4.6.4	Transmit Multi-Byte IFR 1 . . . . .	69
4.6.5	Transmit Multi-Byte IFR 0 . . . . .	70
4.6.6	Transmitting An IFR with the BDLC module . . . . .	70
4.6.7	Transmitting IFR Exceptions . . . . .	76
4.7	Receiving An In-Frame Response (IFR) . . . . .	78
4.7.1	Receiving an IFR with the BDLC module . . . . .	78
4.7.2	Receiving IFR Exceptions . . . . .	79
4.8	Special BDLC Module Operations . . . . .	80
4.8.1	Transmitting Or Receiving A Block Mode Message . . . . .	80
4.8.2	Receiving A Message In 4X Mode . . . . .	80
4.9	BDLC Module Initialization . . . . .	81
4.9.1	Initialization Sequence . . . . .	82
4.9.2	Initializing the Configuration Bits . . . . .	82
4.9.3	Exiting Loopback Mode and Enabling the BDLC module . . . . .	83
4.9.4	Enabling BDLC Interrupts . . . . .	83

## Section 5 Resets

5.1	General . . . . .	87
-----	-------------------	----

## Section 6 Interrupts

6.1	General . . . . .	89
-----	-------------------	----

## Appendix A Electrical Specifications



## List of Tables

Table 3-1	Module Memory Map . . . . .	21
Table 3-2	Interrupt Summary . . . . .	23
Table 3-3	Transmit In-Frame Response Control Bit Priority Encoding . . . . .	27
Table 3-4	BARD Values vs. Transceiver Delay and Transmitter Timing Adjustment . . . . .	33
Table 3-5	BDLC Rate Selection for Binary Frequencies [CLKS = 1] . . . . .	34
Table 3-6	BDLC Rate Selection for Integer Frequencies [CLKS = 0] . . . . .	35
Table 4-1	BDLC Transmitter VPW Symbol Timing for Integer Frequencies . . . . .	42
Table 4-2	BDLC Transmitter VPW Symbol Timing for Binary Frequencies . . . . .	43
Table 4-3	BDLC Receiver VPW Symbol Timing for Integer Frequencies . . . . .	43
Table 4-4	BDLC Receiver VPW Symbol Timing for Binary Frequencies . . . . .	44
Table 4-5	BDLC Receiver VPW 4X Symbol Timing for Integer Frequencies . . . . .	44
Table 4-6	BDLC Receiver VPW 4X Symbol Timing for Binary Frequencies . . . . .	44
Table 4-7	BDLC module J1850 Error Summary . . . . .	52
Table 4-8	IFR Control Bit Priority Encoding . . . . .	69
Table 6-1	Interrupt Summary . . . . .	89





## List of Figures

Figure 1-1	BDLC Operating Modes State Diagram. . . . .	12
Figure 1-2	BDLC Block Diagram. . . . .	16
Figure 3-1	BDLC Control Register 1 . . . . .	21
Figure 3-2	BDLC State Vector Register . . . . .	23
Figure 3-3	BDLC Control Register 2 . . . . .	25
Figure 3-4	Types of In-Frame Response . . . . .	28
Figure 3-5	BDLC Data Register . . . . .	31
Figure 3-6	BDLC Analog Round Trip Delay Register . . . . .	32
Figure 3-7	BDLC Rate Select Register . . . . .	34
Figure 3-8	BDLC Control Register . . . . .	35
Figure 3-9	BDLC Status Register . . . . .	36
Figure 4-1	J1850 Bus Message Format (VPW) . . . . .	37
Figure 4-2	J1850 VPW Symbols. . . . .	40
Figure 4-3	J1850 VPW Passive Symbols . . . . .	46
Figure 4-4	J1850 VPW EOF and IFS Symbols. . . . .	47
Figure 4-5	J1850 VPW Active Symbols . . . . .	48
Figure 4-6	J1850 VPW BREAK Symbol . . . . .	49
Figure 4-7	J1850 VPW Bitwise Arbitrations . . . . .	50
Figure 4-8	BDLC Module Rx Digital Filter Block Diagram. . . . .	54
Figure 4-9	BDLC Protocol Handler Outline. . . . .	56
Figure 4-10	Basic BDLC Transmit Flowchart . . . . .	62
Figure 4-11	Basic BDLC Receive Flowchart. . . . .	66
Figure 4-12	Transmitting A Type 1 IFR. . . . .	72
Figure 4-13	Transmitting A Type 2 IFR. . . . .	74
Figure 4-14	Transmitting A Type 3 IFR. . . . .	77
Figure 4-15	Receiving An IFR With the BDLC module . . . . .	79
Figure 4-16	Basic BDLC Module Transmit Flowchart. . . . .	81
Figure 4-17	Basic BDLC Module Initialization Flowchart . . . . .	85



## Section 1 Introduction

### 1.1 Overview

The BDLC module is a serial communication module which allows the user to send and receive messages across a Society of Automotive Engineers (SAE) J1850 serial communication network. The user's software handles each transmitted or received message on a byte-by-byte basis, while the BDLC performs all of the network access, arbitration, message framing and error detection duties.

It is recommended that the reader be familiar with the operation and requirements of the SAE J1850 protocol as described in the document "SAE Standard J1850 Class B Data Communications Network Interface" prior to proceeding with this specification.

The BDLC module is designed in a modular structure for use as an IP block. A general working knowledge of the IP Bus signals and bus control is assumed in the writing of this document. For details, refer to the SRS IP Bus specifications.

### 1.2 Features

Features of the BDLC module include the following:

- SAE J1850 Class B Data Communications Network Interface Compatible and ISO Compatible for Low-Speed ( $\leq 125$  Kbps) Serial Data Communications in Automotive Applications
- 10.4 Kbps Variable Pulse Width (VPW) Bit Format
- Digital Noise Filter
- Digital Loopback Mode
- 4X Receive Mode, 41.6 Kbps, Supported
- Block Mode Receive and Transmit Supported
- Collision Detection
- Hardware Cyclical Redundancy Check (CRC) Generation and Checking
- Dedicated Register for Symbol Timing Adjustments
- IP Bus Interface
- In-Frame Response (IFR) Types 0, 1, 2, and 3 Supported
- Power-Saving Stop and Wait Modes with Automatic Wakeup on Network Activity
- Polling and CPU Interrupt Generation with Vector Lookup Available

### 1.3 Modes of Operation

- The BDLC module has 6 main modes of operation which interact with the power supplies, pins, and the rest of the MCU as shown below.

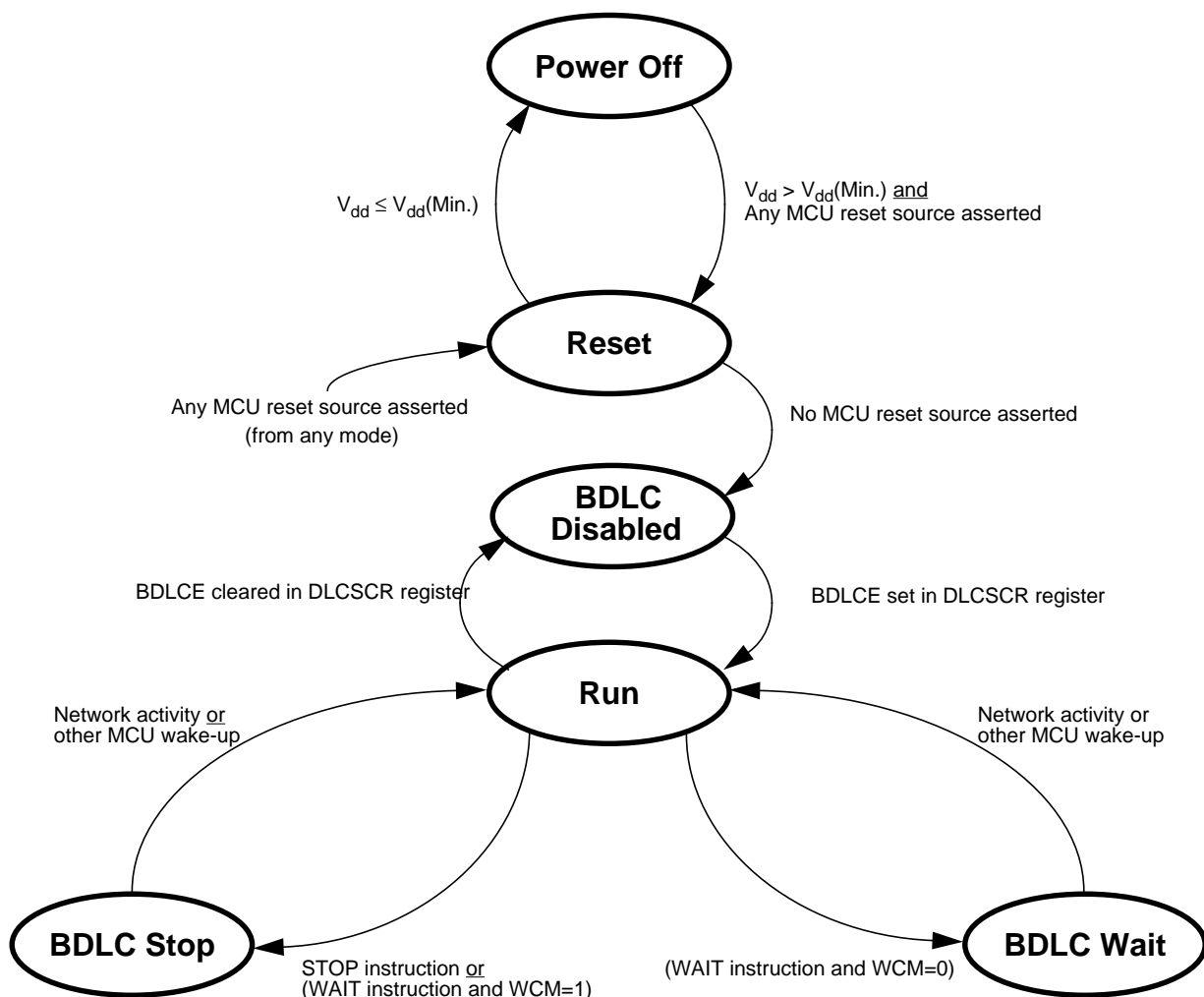


Figure 1-1 BDLC Operating Modes State Diagram

- **Power Off**

This mode is entered from the Reset mode whenever the BDLC module supply voltage  $V_{dd}$  drops below its minimum specified value for the BDLC module to guarantee operation. The BDLC module will be placed in the Reset mode by a system Low Voltage Reset (LVR) before being powered down. In this mode, the pin input and output specifications are not guaranteed.

- **Reset**

This mode is entered from the Power Off mode whenever the BDLC module supply voltage  $V_{dd}$  rises above its minimum specified value ( $V_{dd(MIN)}$ ) and some MCU reset source is asserted. To prevent the BDLC from entering an unknown state, the internal MCU reset is asserted while

powering up the BDLC module. BDLC Reset mode is also entered from any other mode as soon as one of the MCU's possible reset sources (e.g. LVR, POR, COP watchdog, Reset pin etc.) is asserted.

In this mode, the internal BDLC module voltage references are operative,  $V_{dd}$  is supplied to the internal circuits, which are held in their reset state and the internal BDLC module system clock is running. Registers will assume their reset condition. Outputs are held in their programmed Reset state, inputs and network activity are ignored.

- **BDLC Disabled**

This mode is entered from the Reset mode after all MCU reset sources are no longer asserted. It is entered from the Run mode whenever the BDLCE bit in the DLCSCR register is cleared.

In this mode the mux interface clock ( $f_{bdlc}$ ) is stopped to conserve power and allow the BDLC module to be configured for proper operation on the J1850 bus. The IP bus interface clocks are left running in this mode to allow access to all BDLC module registers for initialization.

- **Run**

This mode is entered from the BDLC Disabled mode when the BDLCE bit in the DLCSCR register is set. It is entered from the BDLC Wait mode whenever activity is sensed on the J1850 bus or some other MCU source wakes the CPU out of Wait mode.

It is entered from the BDLC Stop mode whenever network activity is sensed or some other MCU source wakes the CPU out of Stop mode. Messages will not be received properly until the clocks have stabilized and the CPU is also in the Run mode.

- **BDLC Wait (Core Specific)**

This power conserving mode is automatically entered from the Run mode whenever the CPU executes a WAIT instruction and if the WCM bit in the DLCBCR1 register is previously cleared. In this mode, the BDLC module internal clocks continue to run. Any activity on the J1850 network will cause the BDLC module to exit BDLC Wait mode and generate an unmaskable interrupt of the CPU. This wakeup interrupt state is reflected in the DLCBSVR, encoded as the highest priority interrupt. This interrupt can be cleared by the CPU with a read of the DLCBSVR.

- Wakeup from BDLC Wait with CPU in WAIT

If the CPU executes the WAIT instruction and the BDLC module enters the WAIT mode ( $WCM = 0$ ), the clocks to the BDLC module as well as the clocks in the MCU continue to run. Therefore, the message which wakes up the BDLC module from WAIT and the CPU from WAIT mode will also be received correctly by the BDLC module. This is because all of the required clocks continue to run in the BDLC module in WAIT mode. The wakeup behavior of the BDLC module applies regardless of whether the BDLC module is in normal or 4X mode when the WAIT instruction is executed.

- **BDLC Stop (Core Specific)**

This power conserving mode is automatically entered from the Run mode whenever the CPU executes a STOP instruction, or if the CPU executes a WAIT instruction and the WCM bit in the DLCBCR1 register is previously set. In this mode, the BDLC internal clocks are stopped. Any activity on the network will cause the BDLC module to exit BDLC Stop mode and generate an

unmaskable interrupt of the CPU. This wakeup interrupt state is reflected in the DLCBSVR, encoded as the highest priority interrupt. This interrupt can be cleared by the CPU with a read of the DLCBSVR. Depending upon which low-power mode instruction the CPU executes to cause the BDLC module to enter BDLC Stop, the message which wakes up the BDLC module (and the CPU) may or may not be received. There are two different possibilities, both of which is described below. These descriptions apply regardless of whether the BDLC module is in normal or 4X mode when the STOP or WAIT instruction is executed.

- Wakeup from BDLC Stop with CPU in STOP

When the CPU executes the STOP instruction, all clocks in the MCU, including clocks to the BDLC module, are turned off. Therefore, the message which wakes up the BDLC module and the CPU from STOP mode will not be received. This is due primarily to the amount of time required for the MCU's oscillator to stabilize before the clocks can be applied internally to the other MCU modules, including the BDLC module.

- Wakeup from BDLC Stop with CPU in WAIT

If the CPU executes the WAIT instruction and the BDLC module enters the Stop mode (WCM = 1), the clocks to the BDLC module are turned off, but the clocks in the MCU continue to run. Therefore, the message which wakes up the BDLC module from Stop and the CPU from WAIT mode will be received correctly by the BDLC module. This is because very little time is required for the CPU to turn the clocks to the BDLC module back on once the wakeup interrupt occurs.

**NOTE:** *While the BDLC module will correctly receive a message which arrives when the BDLC module is in Stop mode or Wait mode and the MCU is in WAIT mode, if the user enters this mode while a message is being received, the data in the message will become corrupted. This is due to the steps required for the BDLC module to resume operation upon exiting Stop mode or Wait mode, and its subsequent resynchronization with the SAE J1850 bus.*

- **Digital Loopback**

When a bus fault has been detected, the digital loopback mode is used to determine if the fault condition is caused by failure in the node's internal circuits or elsewhere in the network, including the node's analog physical interface. In this mode, the input to the digital filter is disconnected from the receive pin input (RXB). The input to the digital filter is then connected to the transmitter output to form the loopback connection. The transmit pin (TXB) is negated and will always drive a passive state onto the bus. Digital loopback mode is entered by setting the DLOOP bit in Section 3.3.3 BDLC Control Register 2 (DLCBCR2).

- **Normal and Emulation Mode Operation (Core Specific)**

The BDLC module operates in the same manner in all Normal and Emulation Modes. All BDLC module registers can be read and written except those that are reserved, unimplemented, or write once. The user must be careful not to unintentionally write a register when using 16-bit writes in order to avoid unexpected BDLC module behavior.

- **Special Mode Operation (Core Specific)**

Some aspects of BDLC module operation can be modified in special test mode. This mode is reserved for internal use only.

- **Low Power Options (Core Specific)**

The BDLC module can save power in Disabled, Wait, and Stop modes. A complete description of what the BDLC module does while in a low power mode can be found in **Section 1.3 Modes of Operation**.

## 1.4 Block Diagram

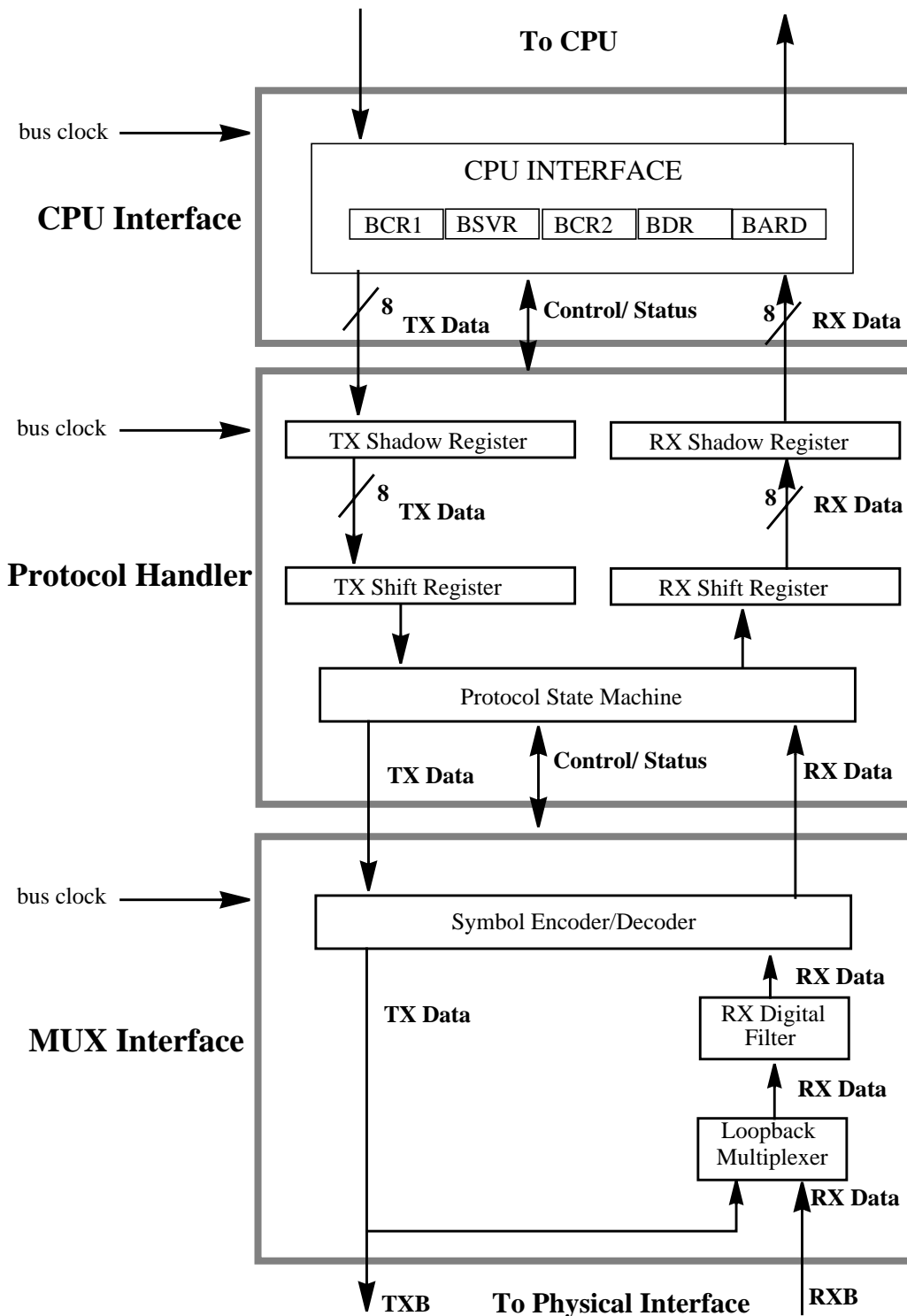


Figure 1-2 BDLC Block Diagram



**Figure 1-2** shows the organization of the BDLC module. The Buffers provide storage for data received and data to be transmitted onto the J1850 bus. The Protocol Handler is responsible for the encoding and decoding of data bits and special message symbols during transmission and reception. The MUX Interface provides the link between the BDLC digital section and the analog Physical Interface. The wave shaping, driving and digitizing of data is performed by the Physical Interface.

**NOTE:** *The Physical Interface is not implemented in the BDLC module and must be provided externally.*

*The main functional blocks of the BDLC module are explained in greater detail in the following sections.*

*Use of the BDLC module in message networking fully implements the “SAE Standard J1850 Class B Data Communication Network Interface” specification.*



## Section 2 Signal Description

### 2.1 Overview

The BDLC module has a total of 2 external pins.

### 2.2 Detailed Signal Descriptions

#### 2.2.1 TXB - BDLC Transmit Pin

The TXB pin serves as the transmit output channel for the BDLC module.

#### 2.2.2 RXB - BDLC Receive Pin

The RXB pin serves as the receive input channel for the BDLC module.



## Section 3 Memory Map and Registers

### 3.1 Overview

This section provides a detailed description of all memory and registers accessible to the end user.

### 3.2 Module Memory Map

Table 3-1 Module Memory Map

Address	Use	Access
Base + \$_00	BDLC Control Register 1 (DLCBCR1)	R/W
Base + \$_01	BDLC State Vector Register (DLCBSVR)	R/W
Base + \$_02	BDLC Control Register 2 (DLCBCR2)	R/W
Base + \$_03	BDLC Data Register (DLCBDR)	R/W
Base + \$_04	BDLC Analog RoundTrip Delay Register (DLCBARD)	R/W
Base + \$_05	BDLC Rate Select Register (DLCBRSR)	R/W
Base + \$_06	BDLC Control Register (DLCSCR)	R/W
Base + \$_07	BDLC Status Register (DLCBSTAT)	R/W

### 3.3 Register Descriptions

#### 3.3.1 BDLC Control Register 1 (DLCBCR1)

This register is used to configure and control the BDLC module.

Register Offset: \$\_00

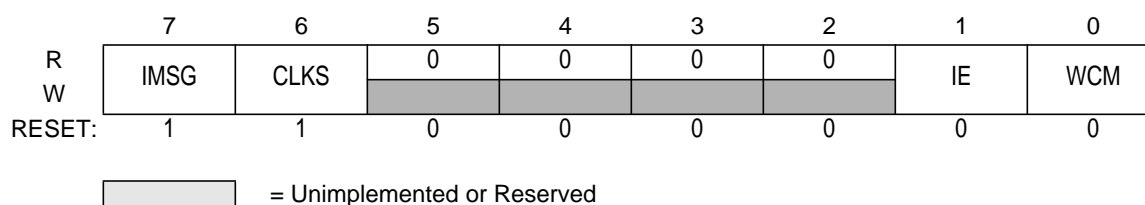


Figure 3-1 BDLC Control Register 1

READ: any time

WRITE: IMSC, IE, and WCM any time.

CLKS write once in normal and emulation modes.

CLKS bit has modified functionality in special test mode.

Writes to unimplemented bits 5-2 are ignored.

## IMSG — Ignore Message (Bit 7)

This bit allows the CPU to ignore messages by disabling updates of the DLCBSVR register until a new Start of Frame (SOF) or a BREAK symbol is detected. BDLC module transmitter and receiver operation are unaffected by the state of the IMSG bit.

- 1 = Disable DLCBSVR Updates. When set, all BDLC interrupt sources (exceptions are described below) will be prevented from updating DLCBSVR status bits. Setting IMSG does not clear pending interrupt flags, the behavior of which will still be as described in Section BDLC State Vector Register (DLCBSVR). If this bit is set while the BDLC is receiving or transmitting a message, state vector register updates will be inhibited for the rest of the message.
- 0 = Enable DLCBSVR Updates. This bit is automatically cleared by the reception of a SOF symbol or a BREAK symbol. It will then allow updates of the state vector register to occur.

There are two situations in which interrupts will not be masked by the IMSG bit: when a wakeup interrupt occurs; and when a receiver error occurs which causes a byte pending transmission to be flushed from the transmit shadow register. See Section 3.3.4 BDLC Data Register (DLCBDR) for a description of the conditions which cause a pending transmission to be flushed.

## CLKS — Clock Select (Bit 6)

The nominal BDLC operating frequency (mux interface clock frequency -  $f_{\text{bdlc}}$ ) **must** always be 1.048576 MHz or 1 MHz in order for J1850 bus communications to take place properly. The CLKS register bit is provided to allow the user to indicate to the BDLC module which frequency (1.048576 MHz or 1 MHz) is used so that each symbol time can be automatically adjusted.

The CLKS bit is a write once bit. All writes to this bit will be ignored after the first one.

- 1 = Binary frequency (1.048576 MHz) is used for  $f_{\text{bdlc}}$ .
- 0 = Integer frequency (1 MHz) is used. for  $f_{\text{bdlc}}$

Section 4.1.3 J1850 VPW Valid/Invalid Bits & Symbols on page 41 describes the transmitter and receiver VPW symbol timing for integer and binary frequencies.

## IE — Interrupt Enable (Bit 1)

This bit determines whether the BDLC module will generate CPU interrupt requests. It does not affect CPU interrupt requests when exiting the BDLC module Stop or Wait modes. Interrupt requests will be maintained until all of the interrupt request sources are cleared, by performing the specified actions upon the BDLC module's registers. Interrupts that were pending at the time that this bit is cleared may be lost.

- 1 = Enable interrupt requests from BDLC module
- 0 = Disable interrupt requests from BDLC module

If the programmer does not wish to use the interrupt capability of the BDLC module, the BDLC State Vector Register (DLCBSVR) can be polled periodically by the programmer to determine BDLC module states. Refer to Section 3.3.2 BDLC State Vector Register (DLCBSVR) on page 23 for a description of DLCBSVR register and how to clear interrupt requests.

WCM — Wait Clock Mode (Bit 0) (*Provided CPU has Low Power Mode Options*)

This bit determines how the BDLC module responds when the CPU enters WAIT mode. As described in Section 1.3 Modes of Operation on page 11, the BDLC module can respond by either entering BDLC\_STOP mode, where all internal clocks are stopped, or entering BDLC\_WAIT mode where internal clocks are allowed to run.

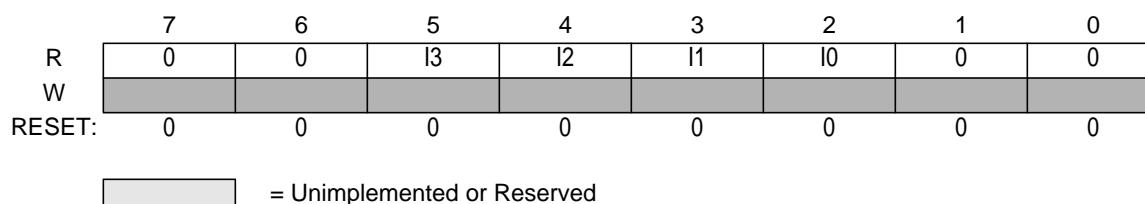
1 = Stop BDLC internal clocks during CPU wait mode (BDLC\_STOP)

0 = Run BDLC internal clocks during CPU wait mode (BDLC\_WAIT)

### 3.3.2 BDLC State Vector Register (DLCBSVR)

This register is provided to substantially decrease the CPU overhead associated with servicing interrupts while under operation of a MUX protocol. It provides a index offset that is directly related to the BDLC module's current state, which can be used with a user supplied jump table to rapidly enter an interrupt service routine. This eliminates the need for the user to maintain a duplicate state machine in software.

Register Offset: \$\_01



**Figure 3-2 BDLC State Vector Register**

READ: any time

WRITE: ignored

I[3:0] — Interrupt State Vector (Bits 5- 2)

These bits indicate the source of the interrupt request that is currently pending.

**Table 3-2 Interrupt Summary**

BSVR	I3	I2	I1	I0	Interrupt Source	Priority
\$00	0	0	0	0	No Interrupts Pending	0 (Lowest)
\$04	0	0	0	1	Received EOF	1
\$08	0	0	1	0	Received IFR byte	2
\$0C	0	0	1	1	Rx data register full	3
\$10	0	1	0	0	Tx data register empty	4
\$14	0	1	0	1	Loss of arbitration	5
\$18	0	1	1	0	CRC error	6
\$1C	0	1	1	1	Symbol invalid or out of range	7
\$20	1	0	0	0	Wakeup	8 (Highest)

The state encoding of the interrupt sources mean that only one interrupt source is dealt with at a time. Once the highest priority interrupt source is dealt with, if another interrupt event of a lower priority has also occurred, the value corresponding to that interrupt source appears in the BSVR. This continues until all BDLC interrupt sources have been dealt with and all bits in the BSVR are cleared.

- Wakeup

The BDLC has two different power-conserving modes, stop and wait. Wakeup from these modes is described below.

- Wakeup from BDLC Wait with CPU in Wait

If the CPU executes a WAIT instruction and the BDLC enters the BDLC wait mode, the clocks to the BDLC as well as the clocks in the MCU continue to run. The message which generates a Wake-up interrupt of the BDLC and the CPU will be received correctly.

- Wakeup from BDLC Stop with CPU in Wait

If the CPU executes a WAIT instruction and the BDLC enters the BDLC stop mode, the clocks to the BDLC are turned off, but the clocks in the MCU continue to run. The message which generates a Wake-up interrupt of the BDLC and the CPU will be received correctly. To ensure this, the EOF following the last message appearing on the bus must be received; otherwise, the message will not be received correctly.

- Wakeup from BDLC Stop with CPU in Stop

If the CPU executes a STOP all clocks to the BDLC as well as the clocks in the MCU are turned off including clocks to the BDLC. The message which generates a Wake-up interrupt of the BDLC and the CPU will not be received correctly.

- Symbol Invalid or Out of Range

- CRC Error

The Cyclical Redundancy Check Byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. If the message is not error free, the CRC error status is shown in the BSVR.

- Loss of Arbitration

The Loss of Arbitration status is entered when a loss of arbitration occurs while the BDLC is transmitting onto the bus.

- Tx Data Register Empty

The Tx Data Register Empty (TDRE) Byte is used to tell when data has been unloaded from the BDLC Data Register (BDR).

- Rx Data Register Full

The Rx Data Register Full (RDRF) Byte is used to tell when data has been loaded in the BDLC Data Register (BDR).

- Received IFR Byte

The BDLC can transmit and receive all four types of in-frame responses. As each byte of an IFR is received, the BSVR indicates this by setting this state.



- Received EOF

When a 280us passive period on the bus is received, it signifies an EOF. Whenever this occurs, the EOF flag is set.

- No Interrupts Pending

This interrupt cannot generate an interrupt of the CPU.

### 3.3.3 BDLC Control Register 2 (DLCBCR2)

This register controls transmitter operations of the BDLC module.

Register Offset: \$\_02

	7	6	5	4	3	2	1	0
R								
W								
	SMRST	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
RESET:	0	1	0	0	0	0	0	0

Figure 3-3 BDLC Control Register 2

READ: any time

WRITE: any time

**SMRST** — State Machine Reset (Bit 7)

The programmer can use this bit to reset the BDLC state machines to an initial state after the user put the off-chip analog transceiver in loop back mode.

- 1 = Setting SMRST arms the state machine reset generation logic. Setting SMRST does not affect BDLC module behavior in any way.
- 0 = Clearing SMRST after it has been set will cause the generation of a state machine reset. After SMRST is cleared, the BDLC requires the bus to be idle for a minimum of an End of Frame symbol (EOF) time before allowing the reception of a message. The BDLC requires the bus to be idle for a minimum of an Inter-Frame Separator symbol (IFS) time before allowing any message to be transmitted.

**DLOOP** — Digital Loopback Mode (Bit 6)

This bit determines the source to which the input of the digital filter is connected and can be used to isolate bus fault conditions. If a fault condition has been detected on the bus, this control bit allows the programmer to disconnect the digital filter from input from the receive pin (RXB) and connect it to the transmit output to the pin (TXB). In this configuration, data sent from the transmit buffer should be reflected back into the receive buffer. If no faults exist in the digital block, the fault is in the physical interface block or elsewhere on the J1850 bus.

- 1 = When set, digital filter input is connected to the transmitter output. The BDLC module is now in Digital Loopback Mode of operation. The transmit pin (TXB) is driven low and not driven by the transmitter output.
- 0 = When cleared, digital filter input is connected to receive pin (RXB) and the transmitter output is connected to the transmit pin (TXB). The BDLC module is taken out of Digital Loopback Mode and can now drive and receive from the J1850 bus normally. After writing DLOOP to zero, the BDLC module requires the bus to be idle for a minimum of an End of Frame symbol time before allowing a reception of a message. The BDLC module requires the bus to be idle for a minimum of an Inter-Frame Separator symbol time before allowing any message to be transmitted.

**NOTE:** *The DLOOP bit is a fault condition aid and should never be altered after the DLCBDR is loaded for transmission. Changing DLOOP during a transmission may cause corrupted data to be transmitted onto the J1850 network.*

#### RX4XE — Receive 4X Enable (Bit 5)

This bit determines if the BDLC operates at normal transmit and receive speed (10.4 kbps) or receive only at 41.6 kbps. This feature is useful for fast download of data into a J1850 node for diagnostic or factory programming of the node.

- 1 = When set, the BDLC module is put in 4X (41.6 kbps) receive only operation.
- 0 = When cleared, the BDLC module transmits and receives at 10.4 kbps. Reception of a BREAK symbol automatically clears this bit and sets the symbol invalid or out of range flag (DLCBSVR = \$1C).

The effect of 4X receive operation on receive symbol timing boundaries is described in Transmit and Receive Symbol Timing Specifications. The RX4XE bit is not affected by entry or exit from BDLC stop or wait modes.

#### NBFS — Normalization Bit Format Select (Bit 4)

This bit controls the format of the Normalization Bit (NB). SAE J1850 strongly encourages the use of an active long: '0' for In-Frame Responses containing CRC and active short, '1' for In-Frame Responses without CRC.

- 1 = NB that is received or transmitted is a '0' when the response part of an In-Frame Response (IFR) ends with a CRC byte. NB that is received or transmitted is a '1' when the response part of an In-Frame Response (IFR) does not end with a CRC byte.
- 0 = NB that is received or transmitted is a '1' when the response part of an In-Frame Response (IFR) ends with a CRC byte. NB that is received or transmitted is a '0' when the response part of an In-Frame Response (IFR) does not end with a CRC byte.

#### TEOD — Transmit End of Data (Bit 3)

This bit is set by the programmer to indicate the end of a message being sent by the BDLC. It will append an 8-bit CRC after completing transmission of the current byte in the Tx Shift Register followed by the EOD symbol. If the transmit shadow register (refer to Rx & Tx Shadow Registers for a description of the transmit shadow register) is full when TEOD is set, the CRC byte and EOD will be transmitted after the current byte in the Tx Shift Register and the byte in the Tx Shadow Register have been transmitted. Once TEOD is set, the transmit data register empty flag (TDRE) in the BDLC

State Vector Register (DLCBSVR) is cleared to allow lower priority interrupts to occur. This bit is also used to end an IFR. Bits TSIFR, TMIFR1, and TMIFR0 determine whether a CRC byte is appended before EOD transmission for IFRs.

- 1 = Transmit EOD symbol.
- 0 = The TEOD bit will be automatically cleared after the first CRC bit is sent, or if an error or loss of arbitration is detected on the bus. When TEOD is used to end an IFR transmission, TEOD is cleared when the BDLC receives back a valid EOD symbol, or an error condition or loss of arbitration occurs.

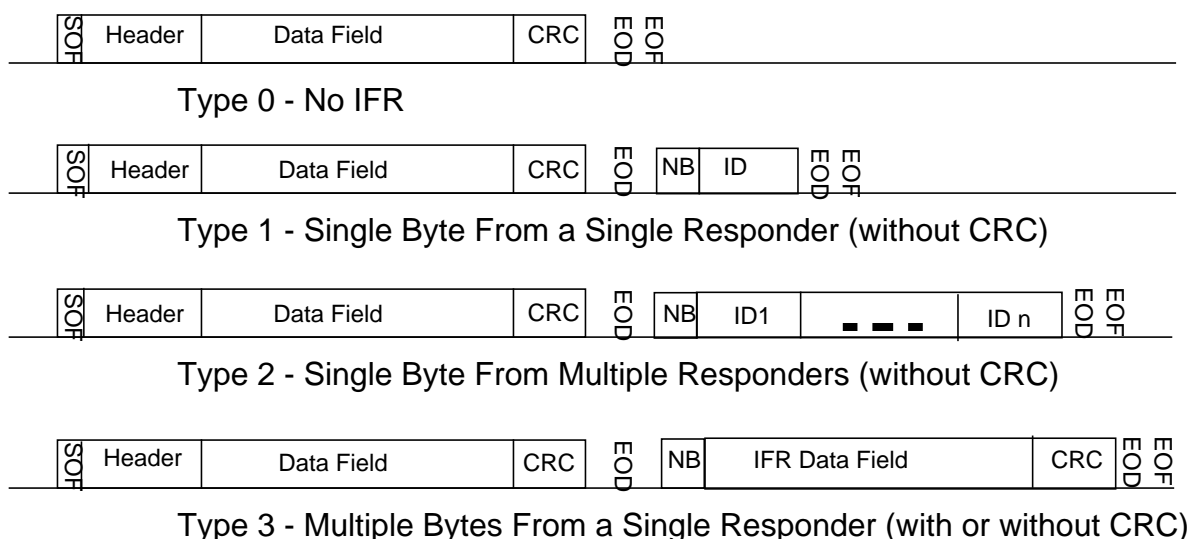
**TSIFR, TMIFR1, TMIFR0 — Transmit In-Frame Response Control (Bits 2-0)**

These three bits control the type of In-Frame Response being sent. The programmer should not set more than one of these control bits to a one at any given time. However, if more than one of these three control bits are set to one, the priority encoding logic will force the internal register bits to a known value as shown in the following table. But, when these bits are read, they will be the same as written earlier. For instance, if “011” is written to TSIFR, TMIFR1, TMIFR0, then internally, they’ll be encoded as “010”. However, when these bits are later read back, it’ll still be “011”.

**Table 3-3 Transmit In-Frame Response Control Bit Priority Encoding**

WRITE			READ			ACTUAL (internal register)		
TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0
0	0	0	0	0	0	0	0	0
1	X	X	1	X	X	1	0	0
0	1	X	0	1	X	0	1	0
0	0	1	0	0	1	0	0	1

The BDLC supports the In-frame Response (IFR) feature of J1850 by setting these bits correctly. The four types of J1850 IFR are shown in Figure 3-4. The purpose of the in-frame response modes is to allow single or multiple nodes to acknowledge receipt of the data by responding to a received message after they have seen the EOD symbol. For VPW modulation, the first bit of the IFR is always passive; therefore, an active normalization bit must be generated by the responder and sent prior to its ID/address byte. When there are multiple responders on the J1850 bus, only one normalization bit is sent which assists all other transmitting nodes to sync their responses.



**Figure 3-4 Types of In-Frame Response**

TSIFR — Transmit Single Byte IFR with no CRC (Type 1 or 2)

This bit is used to request the BDLC to transmit the byte in the BDLC Data Register (DLCBDR) as a single byte IFR with no CRC. Typically, the byte transmitted is a unique identifier or address of the transmitting (responding) node.

- 1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC module will attempt to transmit the appropriate normalization bit followed by the byte in the DLCBDR.
- 0 = The TSIFR bit will be automatically cleared once the EOD following one or more IFR bytes has been received or an error is detected on the bus.

The user must set the TSIFR bit before the EOF following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node transmits an IFR to this message, the user must set the TSIFR bit before the normalization bit is received or no IFR transmit attempts will be made for the message. If another node does transmit a successful IFR or a reception error occurs, the TSIFR bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.

If a loss of arbitration occurs when the BDLC module attempts transmission, after the IFR byte winning arbitration completes transmission, the BDLC module will again attempt to transmit the byte in the DLCBDR (with no normalization bit). The BDLC module will continue transmission attempts until an error is detected on the bus, or TEOD is set by the CPU, or the BDLC transmission is successful.

**NOTE:** Setting the TEOD bit before transmission of the IFR byte will direct the BDLC to make only one attempt at transmitting the byte.

If loss of arbitration occurs in the last bit of the IFR byte, two additional '1' bits **will not** be sent out because the BDLC will attempt to retransmit the byte in the transmit shift register after the IFR byte winning arbitration completes transmission.

#### TMIFR1 — Transmit Multiple Byte IFR with CRC (Type 3)

This bit requests the BDLC module to transmit the byte in the BDLC Data Register (DLCBDR) as the first byte of a multiple byte IFR with CRC or as a single byte IFR with CRC. Response IFR bytes are still subject to J1850 message length maximums.

- 1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received, the BDLC module will attempt to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into DLCBDR register. After TEOD has been set and the last IFR byte has been transmitted, the CRC byte is transmitted.
- 0 = The TMIFR1 bit will be automatically cleared once the BDLC module has successfully transmitted the CRC byte and EOD symbol, by the detection of an error on the multiplex bus, a transmitter underrun, or loss of arbitration.

After the byte in the DLCBDR has been loaded into the transmit shift register, the TDRE flag will be set in the DLCSVR register, similar to the main message transmit sequence. If the interrupt enable bit (IE in DLBCR1) is set, an interrupt request from the BDLC module is generated. The programmer should then load the next byte of the IFR into the DLCBDR for transmission. When the last byte of the IFR has been loaded into the DLCBDR, the programmer should set the TEOD bit in the BDLC control register 2 (DLBCR2). This will instruct the BDLC module to transmit a CRC byte once the byte in the DLCBDR is transmitted, and then transmit an EOD symbol, indicating the end of the IFR portion of the message frame.

However, if the programmer wishes to transmit a single byte followed by a CRC byte, the programmer should load the byte into the DLCBDR and then set the TMIFR1 bit before the EOD symbol has been received. Once the TDRE flag is set and interrupt occurs (if enabled), the programmer should then set the TEOD bit in DLBCR2. This will result in the byte in the DLCBDR being the only byte transmitted before the IFR CRC byte.

The user must set the TMIFR1 bit before the EOF following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node transmits an IFR to this message, the user must set the TMIFR1 bit before the normalization bit is received or no IFR transmit attempts will be made for the message. If another node does transmit a successful IFR or a reception error occurs, the TMIFR1 bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.

If a transmitter underrun error occurs during transmission (caused by the programmer not writing another byte to the DLCBDR following the TDRE flag being set) the BDLC module will automatically disable the transmitter after the byte currently in the shifter plus two extra 1-bits have been transmitted. The receiver will pick this up as an framing error and relay it in the State Vector Register as an invalid symbol error. The TMIFR1 bit will also be cleared.

If a loss of arbitration occurs when the BDLC module is transmitting a multiple byte IFR with CRC, the BDLC module will go to the loss of arbitration state, set the appropriate flag and cease transmission. The TMIFR1 bit will be cleared and no attempt will be made to retransmit the byte in the DLCBDR. If loss of arbitration occurs in the last bit of the IFR byte, two additional one bits (a passive long followed by an active short) **will** be sent out.

**NOTE:** *The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise on the J1850 bus from corrupting a message.*

#### TMIFR0 — Transmit Multiple Byte IFR with no CRC (Type 3)

This bit is used to request the BDLC module to transmit the byte in the BDLC Data Register (DLCBDR) as the first byte of a multiple byte IFR without CRC. Response IFR bytes are still subject to J1850 message length maximums.

- 1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC module will attempt to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into DLCBDR register. After TEOD has been set, the last IFR byte to be transmitted will be the last byte which was written into the DLCBDR register.
- 0 = The TMIFR0 bit will be automatically cleared once the BDLC module has successfully transmitted the EOD symbol, by the detection of an error on the multiplex bus, a transmitter underrun, or loss of arbitration.

After the byte in the DLCBDR has been loaded into the transmit shift register, the TDRE flag will be set in the DLCBSVR register, similar to the main message transmit sequence. If the interrupt enable bit (IE in DLCBCR1) is set, an interrupt request from the BDLC module is generated. The programmer should then load the next byte of the IFR into the DLCBDR for transmission. When the last byte of the IFR has been loaded into the DLCBDR, the programmer should set the TEOD bit in the DLCBCR2 register. This will instruct the BDLC to transmit an EOD symbol, indicating the end of the IFR portion of the message frame. The BDLC module will not append a CRC.

However, if the programmer wishes to transmit a single byte, the programmer should load the byte into the DLCBDR and then set the TMIFR0 bit before the EOD symbol has been received. Once the TDRE flag is set and interrupt occurs (if enabled), the programmer should then set the TEOD bit in DLCBCR2. This will result in the byte in the DLCBDR being the only byte transmitted.

The user must set the TMIFR0 bit before the EOF following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node transmits an IFR to this message, the user must set the TMIFR0 bit before the normalization bit is received or no IFR transmit attempts will be made for the message. If another node does transmit a successful IFR or a reception error occurs, the TMIFR0 bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.

If a transmitter underrun error occurs during transmission (caused by the programmer not writing another byte to the DLCBDR following the TDRE flag being set) the BDLC module will automatically disable the transmitter after the byte currently in the shifter plus two extra 1-bits have been transmitted. The receiver will pick this up as a framing error and relay it in the State Vector Register as an invalid symbol error. The TMIFR0 bit will also be cleared.

If a loss of arbitration occurs when the BDLC module is transmitting a multiple byte IFR without CRC, the BDLC module will go to the loss of arbitration state, set the appropriate flag and cease transmission. The TMIFR0 bit will be cleared and no attempt will be made to retransmit the byte in the DLCBDR. If loss of arbitration occurs in the last bit of the IFR byte, two additional one bits (a passive long followed by an active short) **will** be sent out.

**NOTE:** *The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise on the J1850 bus from corrupting a message.*

### 3.3.4 BDLC Data Register (DLCBDR)

This register is used to pass the data to be transmitted to the J1850 bus from the CPU to the BDLC module. It is also used to pass data received from the J1850 bus to the CPU.

Register Offset: \$\_03

	7	6	5	4	3	2	1	0
R	D7	D6	D5	D4	D3	D2	D1	D0
W								
RESET:	0	0	0	0	0	0	0	0

Figure 3-5 BDLC Data Register

READ: any time

WRITE: any time

D7:D0 — Receive/Transmit Data (Bits 7 - 0)

While transmitting, each data byte (after the first one) should be written only after a “Tx Data Register Empty” (TDRE) interrupt has occurred, or the DLCBSVR register has been polled indicating this condition.

Data read from this register will be the last data byte received from the J1850 bus. This received data should only be read after a “Rx Data Register Full” (RDRF) or “Received IFR byte” (RXIFR) interrupt has occurred or the DLCBSVR register has been polled indicating either of these two conditions.

The DLCBDR register is double buffered via a transmit shadow register and a receive shadow register. After the byte in the transmit shift register has been transmitted, the byte currently stored in the transmit shadow register is loaded into the transmit shift register. Once the transmit shift register has shifted the first bit out, the TDRE flag is set, and the shadow register is ready to accept the next byte of data.

The receive shadow register works similarly. Once a complete byte has been received, the receive shift register stores the newly received byte into the receive shadow register. The RDRF flag (or RXIFR flag if the received byte is part of an IFR) is set to indicate that a new byte of data has been received. The programmer has one BDLC module byte reception time to read the shadow register and clear the RDRF or RXIFR flag before the shadow register is overwritten by the newly received byte.

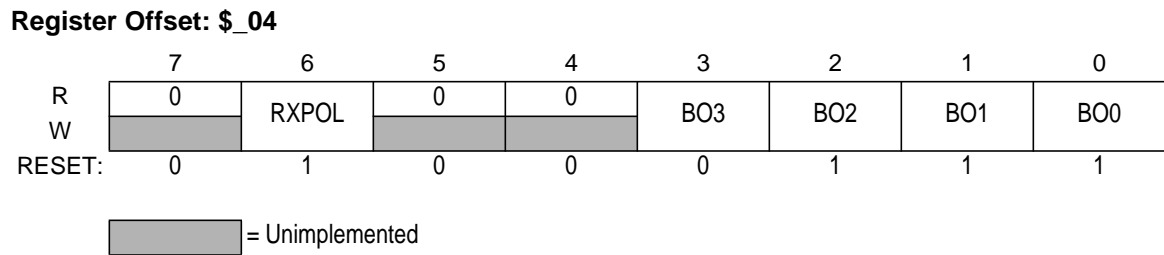
If the user writes the first byte of a message to be transmitted to the DLCBDR and then determines that a different message should be transmitted, the user can write a new byte to the DLCBDR up until the transmission begins. This new byte will replace the original byte in the DLCBDR.

From the time a byte is written to the DLCBDR until it is transferred to the transmit shift register, the transmit shadow register is considered full and the byte pending transmission. If one of the IFR transmission control bits (TSIFR, TMIFR1, or TMIFR0 in DLCBCR2) is also set, the byte is pending transmission as an IFR. A byte pending transmission will be flushed from the transmit shadow register and the transmission canceled if one of the following occurs: a loss of arbitration or transmitter error on the byte currently being transmitted; a symbol error, framing error, bus fault, or BREAK symbol is received. If the byte pending transmission is an IFR byte, the reception of a message with a CRC error will also cause the byte in the transmit shadow register to be flushed.

To abort an in-progress transmission, the programmer should simply stop loading more data into the BDR. This will cause a transmitter underrun error and the BDLC module will automatically disable the transmitter on the next non-byte boundary. This means that the earliest a transmission can be halted is after at least one byte (plus two extra 1-bits) has been transmitted. The receiver will pick this up as an error and relay it in the State Vector Register as an invalid symbol error.

### 3.3.5 BDLC Analog Round Trip Delay Register (DLCBARD)

This register is used to program the BDLC module so that it compensates for the round trip delays of different external transceivers. Also the polarity of the receive pin (RXB) is set in this register.



**Figure 3-6 BDLC Analog Round Trip Delay Register**

READ: any time

WRITE: write once in normal and emulation modes.

Register functionality modified in special test mode.

Writes to unimplemented bits 7, 5, 4 are ignored.

#### RXPOL — Receive Pin Polarity (Bit 6)

The Receive pin Polarity bit is used to select the polarity of incoming signal on the receive pin. Some external analog transceiver inverts the receive signal from the J1850 bus before feeding back to the digital receive pin.



1 = Select normal/true polarity; true non-inverted signal from J1850 bus, i.e., the external transceiver does not invert the receive signal.

0 = Select inverted polarity, where external transceiver inverts the receive signal.

#### BO3-BO0 — BDLC Analog Roundtrip Delay Offset Field (Bits 3-0)

BO[3:0] adjust the transmitted symbol timings to account for the differing roundtrip delays found in different SAE J1850 analog transceivers. The allowable delay range is from 9 ms to 24 ms, with a nominal target of 16 ms (reset value). Refer to Table 3-4 for the BO[3:0] values corresponding to the expected transceiver delays and the resultant transmitter timing adjustment (in mux interface clock periods ( $t_{bdlc}$ )). Refer to the analog transceiver device specification for the expected roundtrip delay through both the transmitter and the receiver. The sum of these two delays makes up the total roundtrip delay value.

**Table 3-4 BARD Values vs. Transceiver Delay and Transmitter Timing Adjustment**

BARD Offset Bits (BO3,BO2,BO1,BO0)	Corresponding Expected Transceiver's delays ( $\mu$ s)	Transmitter Symbol Timing Adjustment ( $t_{bdlc}$ ) <sup>1</sup>
0000	9	9
0001	10	10
0010	11	11
0011	12	12
0100	13	13
0101	14	14
0110	15	15
0111	16	16
1000	17	17
1001	18	18
1010	19	19
1011	20	20
1100	21	21
1101	22	22
1110	23	23
1111	24	24
NOTE: 1. The transmitter symbol timing adjustment is the same for binary and integer bus frequencies.		

### 3.3.6 BDLC Rate Select Register (DLCBRSR)

This register determines the divider prescaler value for the mux interface clock ( $f_{bdlc}$ ).

Register Offset: \$\_05

	7	6	5	4	3	2	1	0
R	0	0	R5	R4	R3	R2	R1	R0
W	Unimplemented							
RESET:	0	0	0	0	0	0	0	0


 = Unimplemented

Figure 3-7 BDLC Rate Select Register

READ: any time

WRITE: write once in normal and emulation modes.

Register functionality modified in special test mode.

Writes to unimplemented bits 7, 6 are ignored.

**NOTE:** After writing to the divide rate register, the divide counter will start counting *ONLY* after the next access to the BDLC register space. E.g. write the module enable bit after writing to the divide rate register.

R5-R0 — Rate Select (Bits 5-0)

These bits determine the amount by which the frequency of the system clock signal is divided to generate the MUX Interface clock ( $f_{\text{bdlc}}$ ) which defines the basic timing resolution of the MUX Interface. The value programmed into these bits is dependent on the chosen system clock frequency. See Table 3-5 and Table 3-6 for example rate selects for different bus frequencies. All divisor values from divide by 1 to divide by 64 are possible, but are not shown in the tables.

**NOTE:** Although the maximum divider is 64, a divider which will generate a 1 MHz or 1.048576 MHz  $f_{\text{bdlc}}$  must be selected in order for J1850 communications to occur.

Table 3-5 BDLC Rate Selection for Binary Frequencies [CLKS = 1]

IP bus clock frequency	R[5:0]	division	$f_{\text{bdlc}}$
$f_{\text{CLOCK}}=1.048576$ MHz	\$00	1	1.048576 MHz
$f_{\text{CLOCK}}=2.09715$ MHz	\$01	2	1.048576 MHz
$f_{\text{CLOCK}}=3.14573$ MHz	\$02	3	1.048576 MHz
$f_{\text{CLOCK}}=4.19430$ MHz	\$03	4	1.048576 MHz
$f_{\text{CLOCK}}=8.38861$ MHz	\$07	8	1.048576 MHz
$f_{\text{CLOCK}}=10.48576$ MHz	\$09	10	1.048576 MHz
$f_{\text{CLOCK}}=67.10886$ MHz	\$3F	64	1.048576 MHz

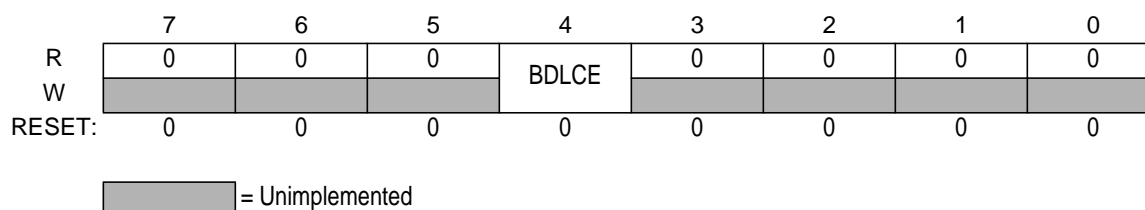
**Table 3-6 BDLC Rate Selection for Integer Frequencies [CLKS = 0]**

IP bus clock frequency	R[5:0]	division	$f_{bdlc}$
$f_{CLOCK}=1.00000$ MHz	\$00	1	1.000000 MHz
$f_{CLOCK}=2.00000$ MHz	\$01	2	1.000000 MHz
$f_{CLOCK}=3.00000$ MHz	\$02	3	1.000000 MHz
$f_{CLOCK}=4.00000$ MHz	\$03	4	1.000000 MHz
$f_{CLOCK}=8.00000$ MHz	\$07	8	1.000000 MHz
$f_{CLOCK}=10.00000$ MHz	\$09	10	1.000000 MHz
$f_{CLOCK}=64.00000$ MHz	\$3F	64	1.000000 MHz

### 3.3.7 BDLC Control Register (DLCSCR)

The following register enables the BLDC module.

Register Offset: \$\_06



**Figure 3-8 BDLC Control Register**

READ: any time

WRITE: any time

**BDLCE** — BDLC Enable (Bit 4)

This bit serves as a mux interface clock ( $f_{bdlc}$ ) enable/disable for power savings.

1 = The mux interface clock ( $f_{bdlc}$ ) and BDLC module are enabled to allow J1850 communications to take place.

0 = The mux interface clock ( $f_{bdlc}$ ) is disabled, shutting down the BDLC module for power saving. Bus clocks are still running allowing registers to be accessed.

### 3.3.8 BDLC Status Register (DLCBSTAT)

This register Indicates the status of the BLDC module.

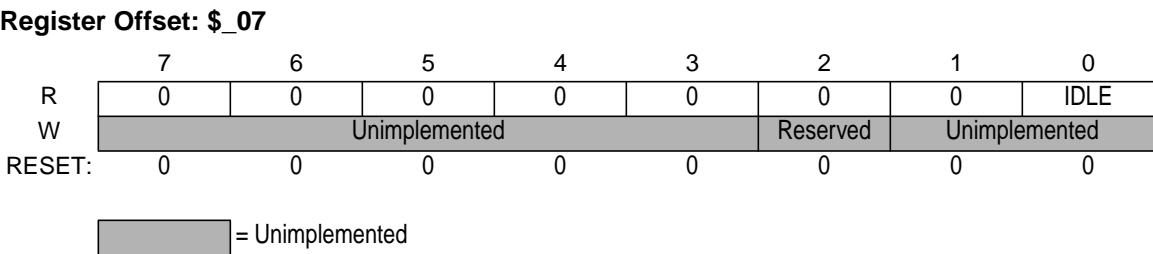


Figure 3-9 BDLC Status Register

READ: any time

WRITE: ignored in normal and emulation modes

Register functionality is modified in special test mode.

IDLE Idle (Bit 0)

This bit indicates when the BDLC module is idle.

- 1 = BDLC module has received IFS and no data is being transmitted or received.
- 0 = BDLC module is either transmitting or receiving data.

**NOTE:** BDLC module is only idle after receiving IFS. The IDLE bit is 0 during reset since the BDLC module needs to wait for an IFS before becoming idle. Noise on the bus will be filtered and the IDLE bit will remain unchanged.

# Section 4 Functional Description

## 4.1 General

The BDLC module is a serial communication module which allows the user to send and receive messages across a Society of Automotive Engineers (SAE) J1850 serial communication network. The user's software handles each transmitted or received message on a byte-by-byte basis, while the BDLC performs all of the network access, arbitration, message framing and error detection duties.

### 4.1.1 J1850 Frame Format

As noted above and in **Section 1.2 Features on page 11**, the BDLC module communicates across an SAE J1850 network. As such, all messages transmitted on the J1850 bus are structured using the format below. The following sections describe this format and it's meanings.



**Figure 4-1 J1850 Bus Message Format (VPW)**

SAE J1850 states that each message has a maximum length of 101 bit times or 12 bytes (excluding SOF, EOD, NB and EOF).

- **SOF - Start of Frame Symbol**  
All messages transmitted onto the J1850 bus must begin with an long active SOF symbol. This indicates to any listeners on the J1850 bus the start of a new message transmission. The SOF symbol is not used in the CRC calculation.
- **Data - In Message Data Bytes**  
The data bytes contained in the message include the message priority/type, message I.D. byte, and any actual data being transmitted to the receiving node. See SAE J1850 - Class B Data Communications Network Interface, for more information about 1 and 3 Byte Headers.  
Messages transmitted by the BDLC module onto the J1850 bus must contain at least one data byte, and therefore can be as short as one data byte and one CRC byte. Each data byte in the message is 8 bits in length, transmitted MSB to LSB.
- **CRC - Cyclical Redundancy Check Byte**

This byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. The BDLC calculates the CRC byte and appends it onto any messages transmitted onto the J1850 bus, and also performs CRC detection on any messages it receives from the J1850 bus.

CRC generation uses the divisor polynomial  $X^8+X^4+X^3+X^2+1$ . The remainder polynomial is initially set to all ones, and then each byte in the message after the SOF symbol is serially processed through the CRC generation circuitry. The one's complement of the remainder then becomes the 8-bit CRC byte, which is appended to the message after the data bytes, in MSB to LSB order.

When receiving a message, the BDLC uses the same divisor polynomial. All data bytes, excluding the SOF and EOD symbols, but including the CRC byte, are used to check the CRC. If the message is error free, the remainder polynomial will equal  $X^7+X^6+X^2$  (\$C4), regardless of the data contained in the message. If the calculated CRC does not equal \$C4, the BDLC will recognize this as a CRC error and set the CRC error flag in the BSVR register.

- EOD - End of Data Symbol

The EOD symbol is a long passive period on the J1850 bus used to signify to any recipients of a message that the transmission by the originator has completed. No flag is set upon reception of the EOD symbol.

- IFR - In Frame Response Bytes

The IFR section of the J1850 message format is optional. Users desiring further definition of in-frame response should review the "SAE J1850 Class B Data Communications Network Interface" specification.

- EOF - End of Frame Symbol

This symbol is a passive period on the J1850 bus, longer than an EOD symbol, which signifies the end of a message. Since an EOF symbol is longer than an EOD symbol, if no response is transmitted after an EOD symbol, it becomes an EOF, and the message is assumed to be completed. The EOF flag is set upon receiving the EOF symbol.

- IFS - Inter-Frame Separation Symbol

The IFS symbol is a passive period on the J1850 bus which allows proper synchronization between nodes during continuous message transmission. The IFS symbol is transmitted by a node following the completion of the EOF period.

When the last byte of a message has been transmitted onto the J1850 bus, and the EOF symbol time has expired, all nodes must then wait for the IFS symbol time to expire before transmitting an SOF, marking the beginning of another message.

However, if the BDLC module is waiting for the IFS period to expire before beginning a transmission and a rising edge is detected before the IFS time has expired, it will internally synchronize to that edge.

A rising edge may occur during the IFS period because of varying clock tolerances and loading of the J1850 bus, causing different nodes to observe the completion of the IFS period at different times. Receivers must synchronize to any SOF occurring during an IFS period to allow for individual clock tolerances.

- Break

If the BDLC module is transmitting at the time a BREAK is detected, it treats the BREAK as if a transmission error had occurred, and halts transmission. The BDLC module cannot transmit a BREAK symbol. If while receiving a message the BDLC module detects a BREAK symbol, it treats the BREAK as a reception error and sets the invalid symbol flag. If while receiving a message in 4X mode, the BDLC module detects a BREAK symbol, it treats the BREAK as a reception error, sets BSVR register to \$1C, and exits 4X mode. The RX4XE bit in BCR2 is automatically cleared upon reception of the BREAK symbol.

- Idle Bus

An idle condition exists on the bus during any passive period after expiration of the IFS period. Any node sensing an idle bus condition can begin transmission immediately.

### 4.1.2 J1850 VPW Symbols

Variable Pulse Width modulation (VPW) is an encoding technique in which each bit is defined by the time between successive transitions, and by the level of the bus between transitions, active or passive. Active and passive bits are used alternately. This encoding technique is used to reduced the number of bus transitions for a given bit rate. See **Section 1.2 Features on page 11**.

The symbol values shown below are nominal values. Refer to the electrical specification for a more complete description of symbol values. Each logic one or logic zero contains a single transition, and can be at either the active or passive level and one of two lengths, either 64 $\mu$ s or 128 $\mu$ s ( $T_{NOM}$  at 10.4kbps baud rate), depending upon the encoding of the previous bit. The SOF, EOD, EOF and IFS symbols will always be encoded at an assigned level and length. See Figure 4-2.

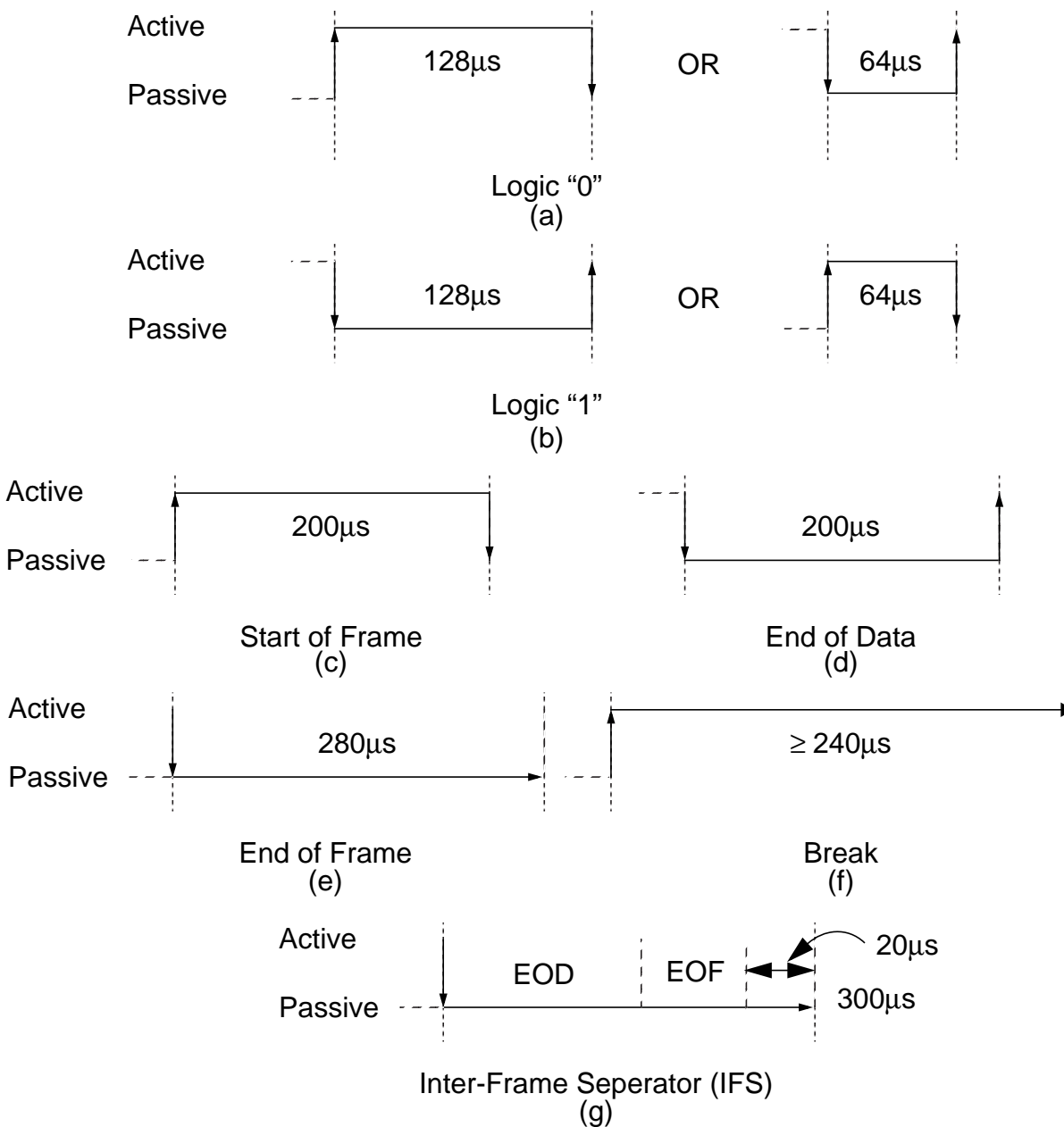


Figure 4-2 J1850 VPW Symbols

Each message will begin with an SOF symbol, an active symbol, and therefore each data byte (including the CRC byte) will begin with a passive bit, regardless of whether it is a logic one or a logic zero. All VPW bit lengths stated in the following descriptions are typical values at a 10.4kbps bit rate.

- Logic "0"



A logic zero is defined as either an active to passive transition followed by a passive period  $64\mu\text{s}$  in length, or a passive to active transition followed by an active period  $128\mu\text{s}$  in length (**Figure 4-2(a)**).

- Logic “1”

A logic one is defined as either an active to passive transition followed by a passive period  $128\mu\text{s}$  in length, or a passive to active transition followed by an active period  $64\mu\text{s}$  in length (**Figure 4-2(b)**).

- NB - Normalization Bit

The NB symbol has the same property as a logic “1” or a logic “0”. It is only used in IFR message responses. This bit is defined as an active bit.

- SOF - Start of Frame Symbol

The SOF symbol is defined as passive to active transition followed by an active period  $200\mu\text{s}$  in length (**Figure 4-2(c)**). This allows the data bytes which follow the SOF symbol to begin with a passive bit, regardless of whether it is a logic one or a logic zero.

- EOD - End of Data Symbol

The EOD symbol is defined as an active to passive transition followed by a passive period  $200\mu\text{s}$  in length (**Figure 4-2(d)**).

- EOF - End of Frame Symbol

The EOF symbol is defined as an active to passive transition followed by a passive period  $280\mu\text{s}$  in length (**Figure 4-2(e)**). If there is no IFR byte transmitted after an EOD symbol is transmitted, after another  $80\mu\text{s}$  the EOD becomes an EOF, indicating the completion of the message.

- IFS - Inter-Frame Separation Symbol

The IFS symbol is defined as a passive period  $300\mu\text{s}$  in length. The IFS symbol contains no transition, since when used it always follows an EOF symbol. (**Figure 4-2(g)**)

- BREAK - Break Signal

The BREAK signal is defined as a passive to active transition followed by an active period of at least  $240\mu\text{s}$  (**Figure 4-2(f)**).

- IDLE

An IDLE is defined as a passive period greater than  $300\mu\text{s}$  in length.

### 4.1.3 J1850 VPW Valid/Invalid Bits & Symbols

The timing tolerances for receiving data bits and symbols from the J1850 bus have been defined to allow for variations in oscillator frequencies. In many cases the maximum time allowed to define a data bit or symbol is equal to the minimum time allowed to define another data bit or symbol.

Since the minimum resolution of the BDLC module for determining what symbol is being received is equal to a single period of the MUX Interface clock, ( $t_{\text{bdlc}}$ ), i.e. the receiver symbol timing boundaries are subject to an uncertainty of  $1 t_{\text{bdlc}}$  due to sampling considerations.

This clock resolution of  $1 t_{\text{bdlc}}$  allows the BDLC module to properly differentiate between the different bits and symbols, without reducing the valid window for receiving bits and symbols from transmitters onto the J1850 bus having varying oscillator frequencies.

- Transmit and Receive Symbol Timing Specifications

Tables 4-1 through 4-6 contain the SAE J1850 transmit and receive symbol timing specifications for the BDLC module. The units used in these tables are mux interface clock periods ( $t_{\text{bdlc}}$ ). The mux interface clock is a divided down version of the bus clock input to the module (see Section 3.3.6 BDLC Rate Select Register (DLCBRSR)). The mux interface clock drives the transmit and receive counters which control symbol generation and identification. The symbol timing in effect during J1850 operations is dependent the state of two control bits: the CLKS bit DLCBCR1, which indicates whether the bus clock is an integer frequency or a binary frequency; the RX4XE bit in DLCBCR2, which is used to select 4X receiver operation.

Tables 4-1 and 4-3 indicate the transmit and receive timing for integer bus frequencies (CLKS = 0) and 4X receive operation disabled (RX4XE = 0). It is assumed that for integer bus frequencies the divided down mux interface clock frequency will be 1MHz ( $t_{\text{bdlc}} = 1 \text{ ms}$ ).

Tables 4-2 and 4-4 indicated the transmit and receive timing for binary bus frequencies (CLKS = 1) and 4X receive operation disabled (RX4XE = 0). It is assumed that for binary bus frequencies the divided down mux interface clock frequency will be 1.048576 MHz ( $t_{\text{bdlc}} = 0.953674 \text{ ms}$ ). The symbol timing values are adjusted to compensate for the shortening of the mux interface clock period.

Tables 4-5 and 4-6 show how the receive symbol timing values are adjusted when 4X receive operation is enabled (RX4XE = 1) for both integer bus frequencies (CLKS = 0) and binary bus frequencies (CLKS = 1), respectively.

The values specified in the tables are for the symbols appearing on the SAE J1850 bus. These values assume the BDLC module is communicating on the SAE J1850 bus using an external analog transceiver, and that the BDLC module analog roundtrip delay value programed into the DLCBARD register is the appropriate value for the transceiver being used. If these conditions are not met, the symbol timings being measured on the SAE J1850 bus will be significantly affected. For a detailed description of how symbol timings are measured on the SAE J1850 bus, refer to the appropriate SAE documents.

**Table 4-1 BDLC Transmitter VPW Symbol Timing for Integer Frequencies**

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	$T_{\text{tvp1}}$	62	64	66	$t_{\text{bdlc}}$
2	Passive Logic 1	$T_{\text{tvp2}}$	126	128	130	$t_{\text{bdlc}}$
3	Active Logic 0	$T_{\text{tva1}}$	126	128	130	$t_{\text{bdlc}}$
4	Active Logic 1	$T_{\text{tva2}}$	62	64	66	$t_{\text{bdlc}}$
5	Start of Frame (SOF)	$T_{\text{tva3}}$	198	200	202	$t_{\text{bdlc}}$
6	End of Data (EOD) <sup>1</sup>	$T_{\text{tvp3}}$	162	164	166	$t_{\text{bdlc}}$
7	End of Frame (EOF) <sup>1</sup>	$T_{\text{tv4}}$	238	240	242	$t_{\text{bdlc}}$
8	Inter-Frame Separator (IFS) <sup>1</sup>	$T_{\text{tv5}}$	298	300	302	$t_{\text{bdlc}}$

**Table 4-1 BDLT Transmitter VPW Symbol Timing for Integer Frequencies**

Number	Characteristic	Symbol	Min	Typ	Max	Unit
NOTE: 1. The transmitter timing for this symbol depends upon the minimum detection time of the symbol by the receiver.						

**Table 4-2 BDLT Transmitter VPW Symbol Timing for Binary Frequencies**

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	$T_{tvp1}$	65	67	69	$t_{bdlc}$
2	Passive Logic 1	$T_{tvp2}$	132	134	136	$t_{bdlc}$
3	Active Logic 0	$T_{tva1}$	132	134	136	$t_{bdlc}$
4	Active Logic 1	$T_{tva2}$	65	67	69	$t_{bdlc}$
5	Start of Frame (SOF)	$T_{tva3}$	208	210	212	$t_{bdlc}$
6	End of Data (EOD) <sup>1</sup>	$T_{tvp3}$	170	172	174	$t_{bdlc}$
7	End of Frame (EOF) <sup>1</sup>	$T_{tv4}$	250	252	254	$t_{bdlc}$
8	Inter-Frame Separator (IFS) <sup>1</sup>	$T_{tv5}$	313	315	317	$t_{bdlc}$
NOTE: 1. The transmitter timing for this symbol depends upon the minimum detection time of the symbol by the receiver.						

**Table 4-3 BDLT Receiver VPW Symbol Timing for Integer Frequencies**

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	$T_{rvp1}$	32	64	95	$t_{bdlc}$
2	Passive Logic 1	$T_{rvp2}$	96	128	163	$t_{bdlc}$
3	Active Logic 0	$T_{rva1}$	96	128	163	$t_{bdlc}$
4	Active Logic 1	$T_{rva2}$	32	64	95	$t_{bdlc}$
5	Start of Frame (SOF)	$T_{rva3}$	164	200	239	$t_{bdlc}$
6	End of Data (EOD)	$T_{rvp3}$	164	200	239	$t_{bdlc}$
7	End of Frame (EOF)	$T_{rv4}$	240	280	299	$t_{bdlc}$
8	Inter-Frame Separator (IFS)	$T_{rv5}$	300	---	---	$t_{bdlc}$
9	Break Signal (BREAK)	$T_{rv6}$	240	---	---	$t_{bdlc}$

NOTE:

The receiver symbol timing boundaries are subject to an uncertainty of 1  $t_{bdlc}$  due to sampling considerations.

**Table 4-4 BDLC Receiver VPW Symbol Timing for Binary Frequencies**

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	$T_{rvp1}$	34	67	100	$t_{bdlc}$
2	Passive Logic 1	$T_{rvp2}$	101	134	171	$t_{bdlc}$
3	Active Logic 0	$T_{rva1}$	101	134	171	$t_{bdlc}$
4	Active Logic 1	$T_{rva2}$	34	67	100	$t_{bdlc}$
5	Start of Frame (SOF)	$T_{rva3}$	172	210	251	$t_{bdlc}$
6	End of Data (EOD)	$T_{rvp3}$	172	210	251	$t_{bdlc}$
7	End of Frame (EOF)	$T_{rv4}$	252	293	314	$t_{bdlc}$
8	Inter-Frame Separator (IFS)	$T_{rv5}$	315	---	---	$t_{bdlc}$
9	Break Signal (BREAK)	$T_{rv6}$	252	---	---	$t_{bdlc}$

NOTE:

The receiver symbol timing boundaries are subject to an uncertainty of 1  $t_{bdlc}$  due to sampling considerations.

**Table 4-5 BDLC Receiver VPW 4X Symbol Timing for Integer Frequencies**

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	$T_{rvp1}$	8	16	23	$t_{bdlc}$
2	Passive Logic 1	$T_{rvp2}$	24	32	40	$t_{bdlc}$
3	Active Logic 0	$T_{rva1}$	24	32	40	$t_{bdlc}$
4	Active Logic 1	$T_{rva2}$	8	16	23	$t_{bdlc}$
5	Start of Frame (SOF)	$T_{rva3}$	41	50	59	$t_{bdlc}$
6	End of Data (EOD)	$T_{rvp3}$	41	50	59	$t_{bdlc}$
7	End of Frame (EOF)	$T_{rv4}$	60	70	74	$t_{bdlc}$
8	Inter-Frame Separator (IFS)	$T_{rv5}$	75	---	---	$t_{bdlc}$
9	Break Signal (BREAK)	$T_{rv6}$	60	---	---	$t_{bdlc}$

NOTE:

The receiver symbol timing boundaries are subject to an uncertainty of 1  $t_{bdlc}$  due to sampling considerations.

**Table 4-6 BDLC Receiver VPW 4X Symbol Timing for Binary Frequencies**

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	$T_{rvp1}$	9	17	25	$t_{bdlc}$
2	Passive Logic 1	$T_{rvp2}$	26	34	42	$t_{bdlc}$
3	Active Logic 0	$T_{rva1}$	26	34	42	$t_{bdlc}$

**Table 4-6 BDLC Receiver VPW 4X Symbol Timing for Binary Frequencies**

Number	Characteristic	Symbol	Min	Typ	Max	Unit
4	Active Logic 1	$T_{rva2}$	9	17	25	$t_{bdlc}$
5	Start of Frame (SOF)	$T_{rva3}$	43	53	62	$t_{bdlc}$
6	End of Data (EOD)	$T_{rvp3}$	43	53	62	$t_{bdlc}$
7	End of Frame (EOF)	$T_{rv4}$	63	74	78	$t_{bdlc}$
8	Inter-Frame Separator (IFS)	$T_{rv5}$	79	---	---	$t_{bdlc}$
9	Break Signal (BREAK)	$T_{rv6}$	63	---	---	$t_{bdlc}$

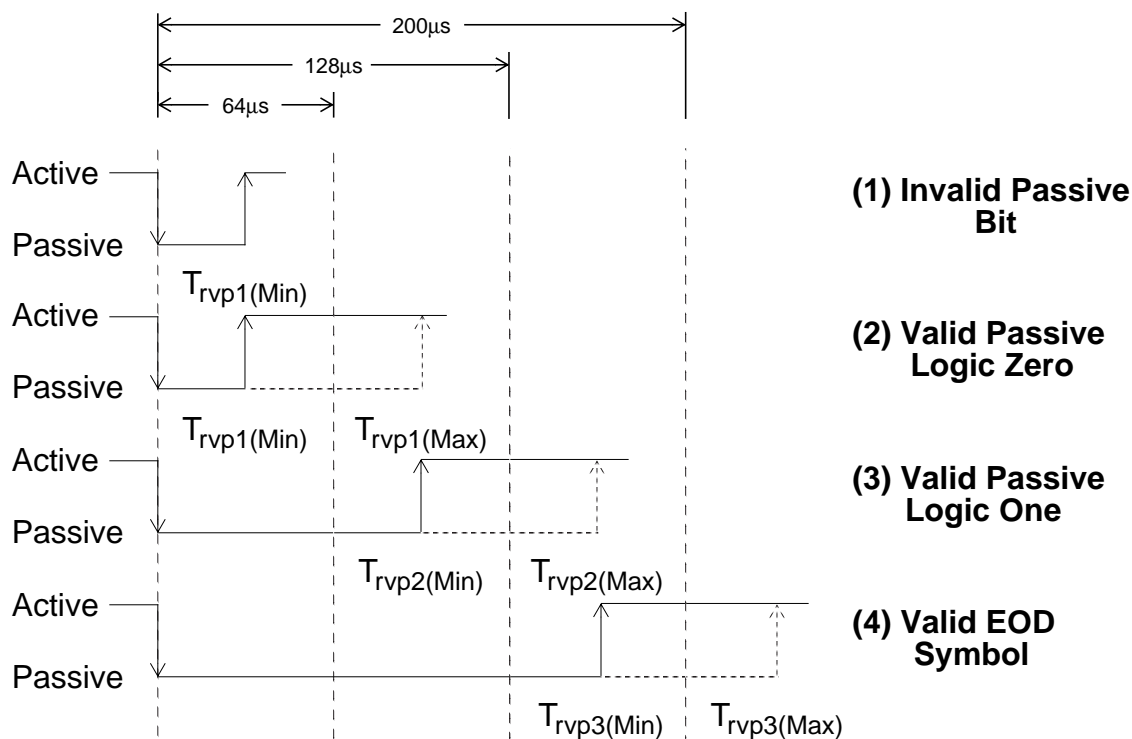
**NOTE:**

The receiver symbol timing boundaries are subject to an uncertainty of  $1 t_{bdlc}$  due to sampling considerations.

The min and max symbol limits shown in the following sections (Invalid Passive Bit - Valid BREAK Symbol) and figures (Figure 4-3 - Figure 4-6) refer to the values listed in Tables 4-1 through 4-6.

- Invalid Passive Bit

If the passive to active transition beginning the next data bit or symbol occurs between the active to passive transition beginning the current data bit or symbol and  $T_{rvp1(\text{Min})}$ , the current bit would be invalid. See **Figure 4-3(1)**.



**Figure 4-3 J1850 VPW Passive Symbols**

- Valid Passive Logic Zero

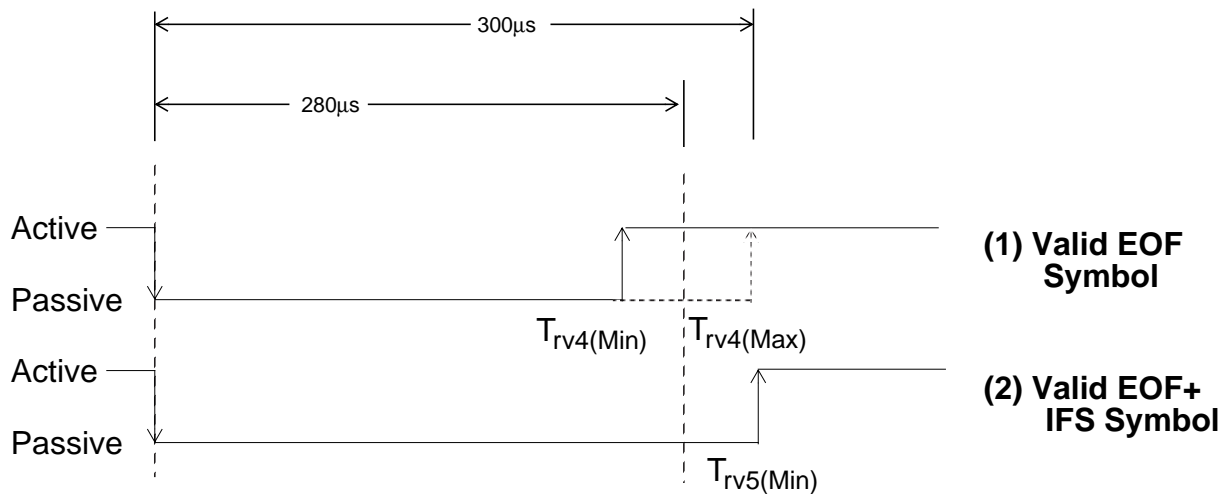
If the passive to active transition beginning the next data bit or symbol occurs between  $T_{rvp1(Min)}$  and  $T_{rvp1(Max)}$ , the current bit would be considered a logic zero. See **Figure 4-3(2)**.

- Valid Passive Logic One

If the passive to active transition beginning the next data bit or symbol occurs between  $T_{rvp2(Min)}$  and  $T_{rvp2(Max)}$ , the current bit would be considered a logic one. See **Figure 4-3(3)**.

- Valid EOD Symbol

If the passive to active transition beginning the next data bit or symbol occurs between  $T_{rvp3(Min)}$  and  $T_{rvp3(Max)}$ , the current symbol would be considered a valid EOD symbol. See **Figure 4-3(4)**.



**Figure 4-4 J1850 VPW EOF and IFS Symbols**

- Valid EOF & IFS Symbol

In **Figure 4-4(1)**, if the passive to active transition beginning the SOF symbol of the next message occurs between  $T_{rv4(Min)}$  and  $T_{rv4(Max)}$ , the current symbol will be considered a valid EOF symbol.

If the passive to active transition beginning the SOF symbol of the next message occurs after  $T_{rv5(Min)}$ , the current symbol will be considered a valid EOF symbol followed by a valid IFS symbol. See **Figure 4-4(2)**. All nodes must wait until a valid IFS symbol time has expired before beginning transmission. However, due to variations in clock frequencies and bus loading, some nodes may recognize a valid IFS symbol before others, and immediately begin transmitting. Therefore, anytime a node waiting to transmit detects a passive to active transition once a valid EOF has been detected, it should immediately begin transmission, initiating the arbitration process.

- Idle Bus

If the passive to active transition beginning the SOF symbol of the next message does not occur before  $T_{tv5(Min)}$ , the bus is considered to be idle, and any node wishing to transmit a message may do so immediately.

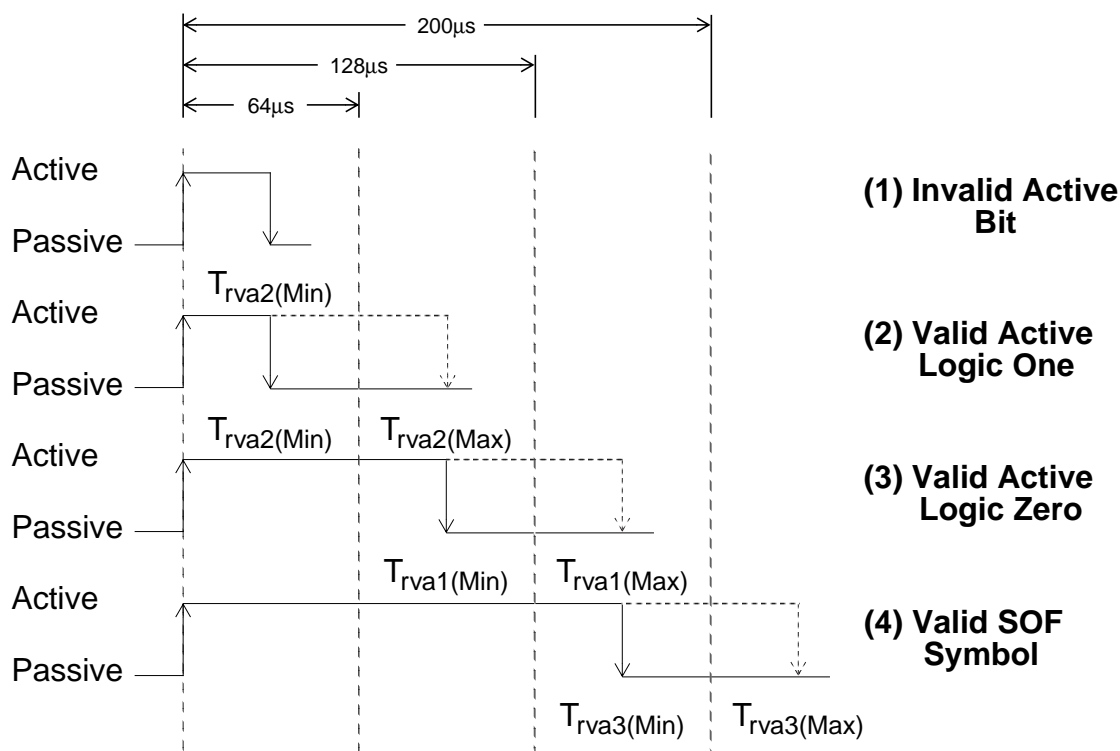


Figure 4-5 J1850 VPW Active Symbols

- Invalid Active Bit

If the active to passive transition beginning the next data bit or symbol occurs between the passive to active transition beginning the current data bit or symbol and  $T_{rva2}(\text{Min})$ , the current bit would be invalid. See **Figure 4-5(1)**.

- Valid Active Logic One

If the active to passive transition beginning the next data bit or symbol occurs between  $T_{rva2}(\text{Min})$  and  $T_{rva2}(\text{Max})$ , the current bit would be considered a logic one. See **Figure 4-5(2)**.

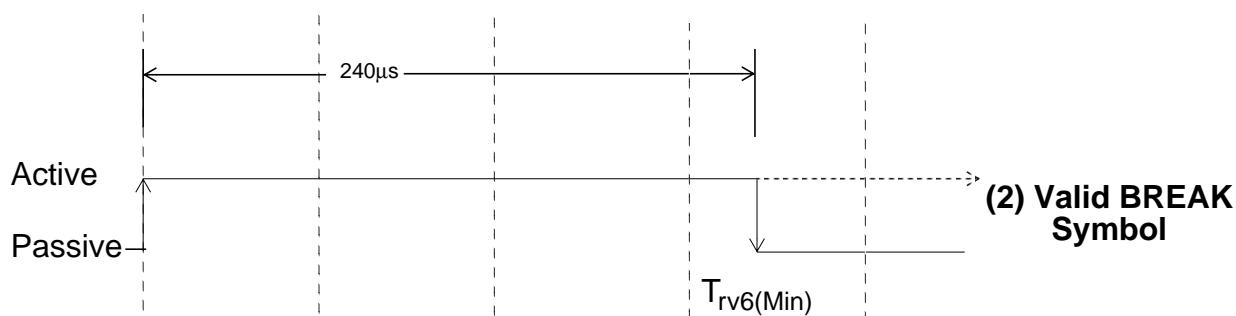
- Valid Active Logic Zero

If the active to passive transition beginning the next data bit or symbol occurs between  $T_{rva1}(\text{Min})$  and  $T_{rva1}(\text{Max})$ , the current bit would be considered a logic zero. See **Figure 4-5(3)**.

- Valid SOF Symbol

If the active to passive transition beginning the next data bit or symbol occurs between  $T_{rva3}(\text{Min})$  and  $T_{rva3}(\text{Max})$ , the current symbol would be considered a valid SOF symbol. See **Figure 4-5(4)**.





**Figure 4-6 J1850 VPW BREAK Symbol**

- Valid BREAK Symbol

If the next active to passive transition does not occur until after  $T_{rv6(Min)}$ , the current symbol will be considered a valid BREAK symbol. A BREAK symbol should be followed by a SOF symbol beginning the next message to be transmitted onto the J1850 bus. See **Figure 4-6**.

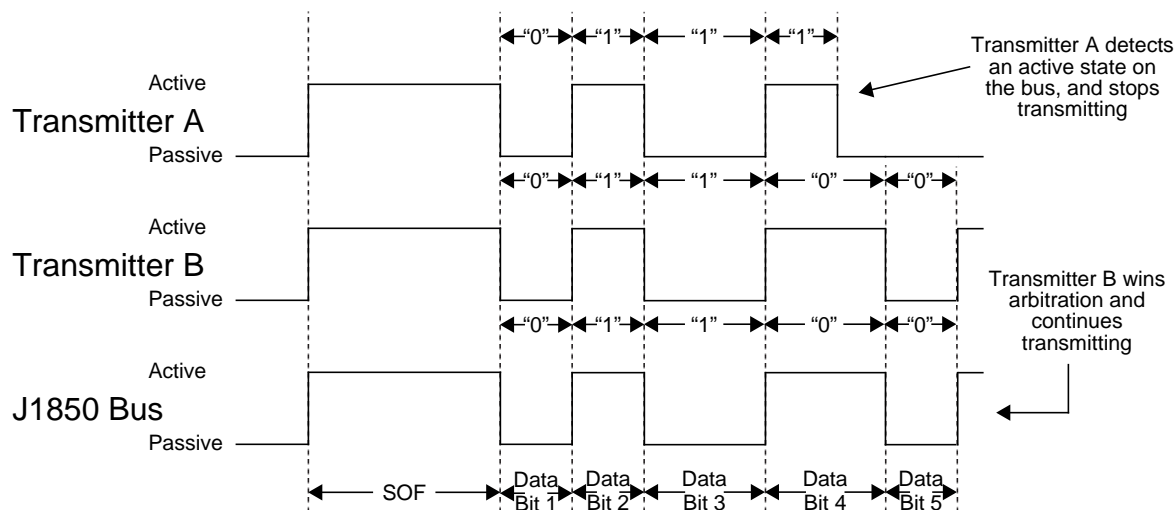
- Message Arbitration

Message arbitration on the J1850 bus is accomplished in a non-destructive manner, allowing the message with the highest priority to be transmitted, while any transmitters which lose arbitration simply stop transmitting and wait for an idle bus to begin transmitting again.

If the BDLC module wishes to transmit onto the J1850 bus, but detects that another message is in progress, it automatically waits until the bus is idle. However, if multiple nodes begin to transmit in the same synchronization window, message arbitration will occur beginning with the first bit after the SOF symbol and continue with each bit thereafter.

The VPW symbols and J1850 bus electrical characteristics are carefully chosen so that a logic zero (active or passive type) will always dominate over a logic one (active or passive type) simultaneously transmitted. Hence logic zeroes are said to be 'dominant' and logic ones are said to be 'recessive'.

Whenever a node transmits a recessive bit and detects a dominant bit, it loses arbitration, and immediately stops transmitting. This is known as 'bitwise arbitration'. The loss of arbitration flag (in DLCBSVR) is set when arbitration is lost. If the interrupt enable bit (IE in DLCBCR1) is set, an interrupt request from the BDLC module is generated. Reading the DLCBSVR register will clear this flag.



**Figure 4-7 J1850 VPW Bitwise Arbitrations**

During arbitration, or even throughout the transmitting message, when an opposite bit is detected, transmission is immediately stopped unless it occurs on the 8th bit of a byte. In this case the BDLC module will automatically append up to two extra 1 bits and then stop transmitting. These two extra bits will be arbitrated normally and thus will not interfere with another message. The second 1 bit will not be sent if the first loses arbitration. If the BDLC module has lost arbitration to another valid message then the two extra ones will not corrupt the current message. However, if the BDLC module has lost arbitration due to noise on the bus, then the two extra ones will ensure that the current message will be detected and ignored as a noise-corrupted message.

Since a “0” dominates a “1”, the message with the lowest value will have the highest priority, and will always win arbitration, i.e. a message with priority 000 will win arbitration over a message with priority 011. This method of arbitration will work no matter how many bits of priority encoding are contained in the message.

#### 4.1.4 J1850 Bus Errors

The BDLC module detects several types of transmit and receive errors which can occur during the transmission of a message onto the J1850 bus.

- Transmission Error

If the BDLC module is transmitting a message and the message received contains a symbol error, a framing error, a bus fault, a BREAK symbol, or a logic ‘1’ symbol when a logic ‘0’ is being transmitted, this constitutes a transmission error. Receiving a logic ‘0’ symbol when transmitting a logic ‘1’ is considered a loss of arbitration condition (See Message Arbitration) and not a transmission error. When a transmission error is detected, the BDLC module will immediately cease transmitting. Further transmission or reception will be disabled until a valid EOF symbol is

detected on the J1850 bus. The error condition is reflected by setting the symbol invalid or out of range flag in the DLCBSVR register. If the interrupt enable bit (IE in DLCBCR1) is set, an interrupt request from the BDLC module is generated. Reading the DLCBSVR register will clear this flag.

- CRC Error

A cyclical redundancy check (CRC) error is detected when the data bytes and CRC byte of a received message are processed, and the CRC calculation result is not equal to \$C4. The CRC code should detect any single and 2 bit errors, as well as all 8 bit burst errors, and almost all other types of errors. The CRC error flag (in DLCBSVR) is set when a CRC error is detected. If the interrupt enable bit (IE in DLCBCR1) is set, an interrupt request from the BDLC module is generated. Reading the DLCBSVR register will clear this flag.

- Symbol Error

A symbol error is detected when an abnormal (invalid) symbol is detected in a message being received from the J1850 bus. See sections Invalid Passive Bit and Invalid Active Bit which define invalid symbols. The symbol invalid or out of range flag (in DLCBSVR) is set when a symbol error is detected. If the interrupt enable bit (IE in DLCBCR1) is set, an interrupt request from the BDLC module is generated. Reading the DLCBSVR register will clear this flag.

- Framing Error

A framing error is detected when a received symbol occurs in an inappropriate location in the message frame. The following situations result in framing errors:

- An active logic “0” or logic “1” received as the first symbol of the frame.
- An SOF symbol received in any location other than the first symbol of a frame. Erroneous locations include: Within the data portion of a message or IFR; Immediately following the EOD in a message or IFR.
- An EOD symbol received on a non-byte boundary in a message or IFR.
- An active logic “0” or logic “1” received immediately following the EOD at the end of an IFR.

The symbol invalid or out of range flag (in DLCBSVR) is set when a framing error is detected. If the interrupt enable bit (IE in DLCBCR1) is set, an interrupt request from the BDLC module is generated. Reading the DLCBSVR register will clear this flag.

- Bus Fault

If a bus fault occurs, the response of the BDLC module will depend upon the type of bus fault.

If the bus is shorted to  $V_{DD}$ , the BDLC module will wait for the bus to fall to a passive state before it will attempt to transmit a message. As long as the short remains, the BDLC will never attempt to transmit a message onto the J1850 bus.

If the bus is shorted to ground, the BDLC module will see an idle bus, begin to transmit the message, and then detect a transmission error, since the short to ground would not allow the bus to be driven to the active (dominant) state. The BDLC module will wait for assertion of the receive pin for  $(64 - \text{analog round trip delay}) t_{\text{bdlc}}$  cycles, after assertion of the transmit pin, before detecting the error. If the transmission is an IFR, the BDLC module will wait for  $(280 - \text{analog round trip delay}) t_{\text{bdlc}}$  cycles before detecting an error. The “analog round trip delay” is determined by the value stored in the DLCBARD register. The BDLC module will set the symbol invalid or out of range flag (in

DLCBSVR), abort that transmission and wait for the next CPU command to transmit. In this case, the transmitter does not have to wait for an EOF symbol to be received to be enabled. If the interrupt enable bit (IE in DLCBCR1) is set, an interrupt request from the BDLC module is generated. Reading the DLCBSVR register will clear this flag.

In any case, if the bus fault is temporary, as soon as the fault is cleared, the BDLC module will resume normal operation. If the bus fault is permanent, it may result in permanent loss of communication on the J1850 bus.

- **BREAK - Break**

Any BDLC transmitting at the time a BREAK is detected will treat the BREAK as if a transmission error had occurred, and halt transmission.

If while receiving a message the BDLC module detects a BREAK symbol, it will treat the BREAK as a reception error.

If a BREAK symbol is received while the BDLC module is transmitting or receiving, the symbol invalid or out of range flag (in DLCBSVR) is set. Further transmission/reception will be disabled until the J1850 bus returns to the passive state and a valid EOF symbol is detected on the J1850 bus. If the interrupt enable bit (IE in DLCBCR1) is set, an interrupt request from the BDLC module is generated. Reading the DLCBSVR register will clear this flag.

The BDLC module cannot transmit a BREAK symbol. It can only receive a BREAK symbol from the J1850 bus.

- **Bus Error Summary**

The possible J1850 bus errors and the actions taken by the BDLC module are summarized in Table 4-7.

**Table 4-7 BDLC module J1850 Error Summary**

Error Condition	BDLC Module Function
Transmission Error	BDLC module will immediately cease transmitting. Further transmission and reception will be disabled until a valid EOF symbol is detected. The symbol invalid or out of range flag will be set and interrupt generated if enabled.
Cyclical Redundancy Check (CRC) Error	CRC error flag set and interrupt generated if enabled.
Symbol Error	The symbol invalid or out of range flag will be set and interrupt generated if enabled. Transmission and reception will be disabled until a valid EOF symbol is detected.
Framing Error	The symbol invalid or out of range flag will be set and interrupt generated if enabled. Transmission and reception will be disabled until a valid EOF symbol is detected.

**Table 4-7 BDLC module J1850 Error Summary**

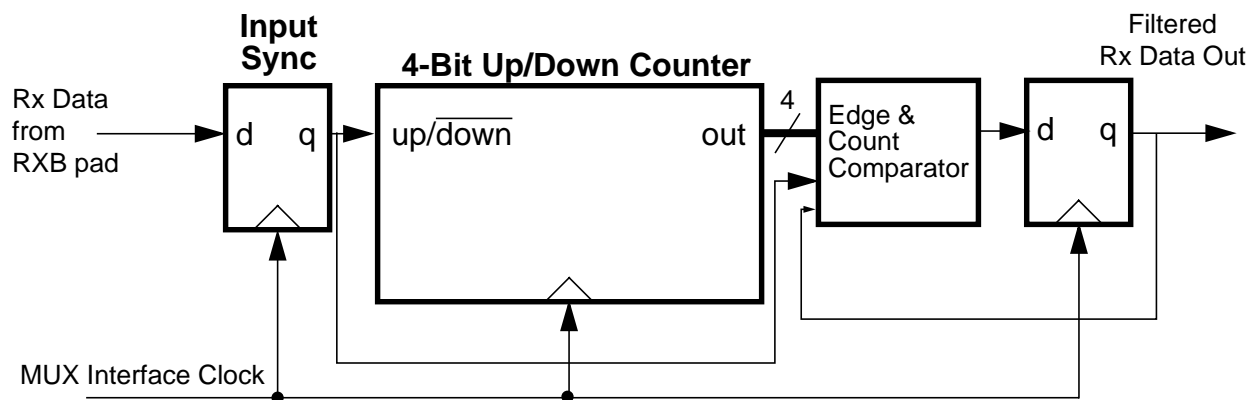
Error Condition	BDLC Module Function
Bus short to V <sub>DD</sub> .	The BDLC module will not transmit until short is corrected and a valid EOF is detected. Depending upon when short occurs and is corrected, this error condition may set the symbol invalid or out of range, crc error, or loss of arbitration flags.
Bus short to GND.	Short will be seen as an idle bus by BDLC module. If a transmission attempt is made before short is corrected, the symbol invalid or out of range flag will be set and interrupt generated if enabled. Another transmission can be initiated as soon as short is corrected.
BREAK symbol reception	If doing so, the BDLC module will immediately cease transmitting. Symbol invalid or out of range flag set and interrupt generated if enabled. Transmission and reception will be disabled until a valid EOF symbol is detected.

## 4.2 Mux Interface

The MUX Interface is responsible for bit encoding/decoding and digital noise filtering between the Protocol Handler and the Physical Interface. Refer to **Figure 1-2 BDLC Block Diagram on page 16**.

### 4.2.1 Mux Interface - Rx Digital Filter

The Receiver section of the BDLC module includes a digital low pass filter to remove narrow noise pulses from the incoming message. An outline of the digital filter is shown in Figure 4-8.



**Figure 4-8 BDLC Module Rx Digital Filter Block Diagram**

- **Operation**

The clock for the digital filter is provided by the MUX Interface clock. At each positive edge of the clock signal, the current state of the Receiver input signal from the RXB pad is sampled. The RXB signal state is used to determine whether the counter should increment or decrement at the next positive edge of the clock signal.

The counter will increment if the input data sample is high but decrement if the input sample is low. The counter will thus progress up towards '15' if, on average, the RXB signal remains high or progress down towards '0' if, on average, the RXB signal remains low.

When the counter eventually reaches the value '15', the digital filter decides that the condition of the RXB signal is at a stable logic level one and the Data Latch is set, causing the Filtered Rx Data signal to become a logic level one. Furthermore, the counter is prevented from overflowing and can only be decremented from this state.

Alternatively, should the counter eventually reach the value '0', the digital filter decides that the condition of the RXB signal is at a stable logic level zero and the Data Latch is reset, causing the Filtered Rx Data signal to become a logic level zero. Furthermore, the counter is prevented from underflowing and can only be incremented from this state.

The Data Latch will retain its value until the counter next reaches the opposite end point, signifying a definite transition of the RXB signal.

- **Performance**

The performance of the digital filter is best described in the time domain rather than the frequency domain.

If the signal on the RXB signal transitions, then there will be a delay before that transition appears at the Filtered Rx Data output signal. This delay will be between 15 and 16 clock periods, depending on where the transition occurs with respect to the sampling points. This ‘filter delay’ must be taken into account when performing message arbitration.

For example, if the frequency of the MUX Interface clock ( $f_{\text{bdlc}}$ ) is 1.0486MHz, then the period ( $t_{\text{bdlc}}$ ) is 954ns and the maximum filter delay in the absence of noise will be 15.259us.

The effect of random noise on the RXB signal depends on the characteristics of the noise itself. Narrow noise pulses on the RXB signal will be completely ignored if they are shorter than the filter delay. This provides a degree of low pass filtering.

If noise occurs during a symbol transition, the detection of that transition may be delayed by an amount equal to the length of the noise burst. This is just a reflection of the uncertainty of where the transition is truly occurring within the noise.

Noise pulses that are wider than the filter delay, but narrower than the shortest allowable symbol length will be detected by the next stage of the BDLC module’s receiver as an invalid symbol.

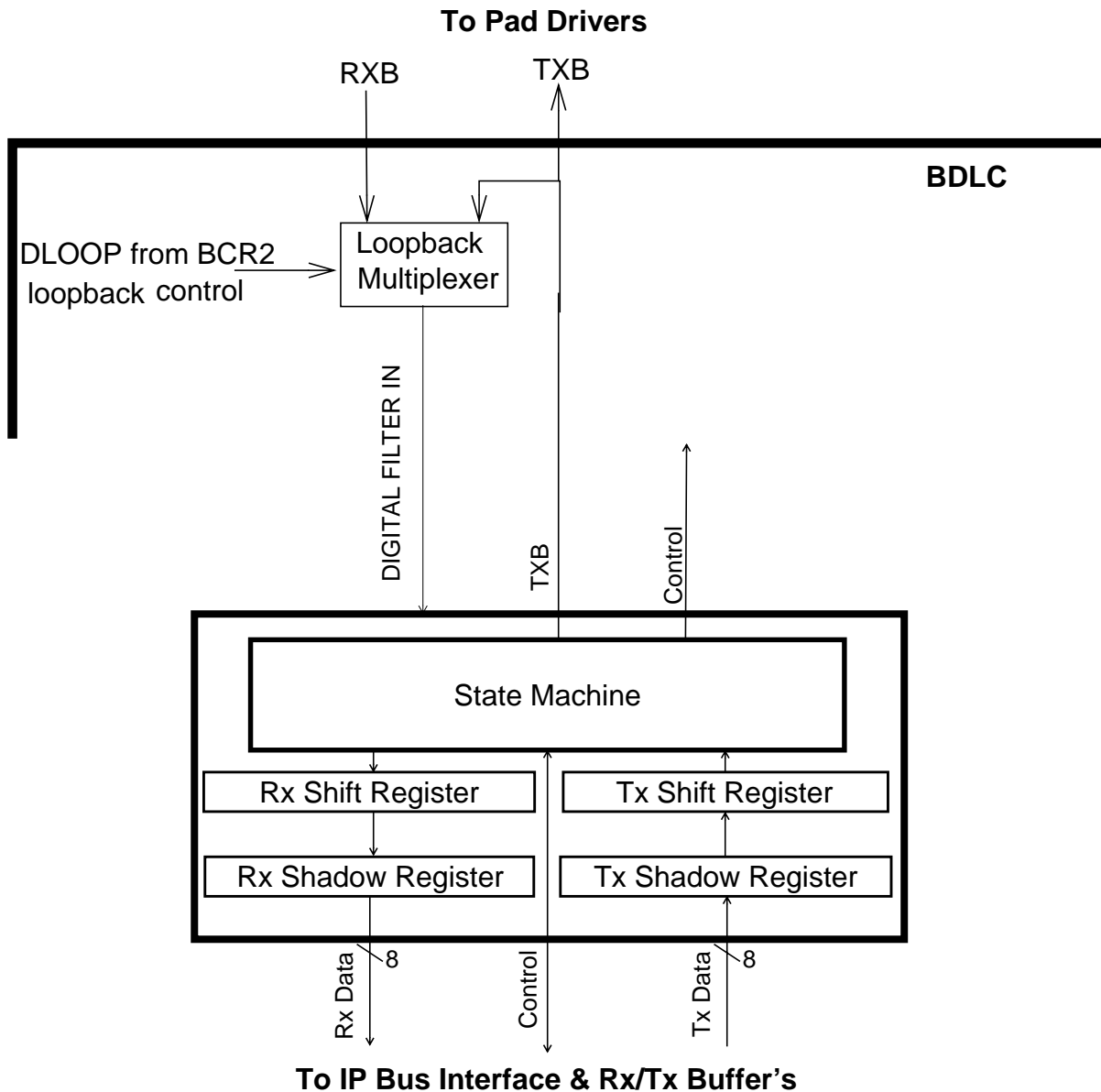
Noise pulses that are longer than the shortest allowable symbol length will normally be detected as an invalid symbol or as invalid data when the frame’s CRC is checked.

## 4.3 Protocol Handler

The Protocol Handler is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. The Protocol Handler conforms to SAE J1850 - Class B Data Communications Network Interface. Refer to **Figure 1-2 BDLC Block Diagram on page 16**

### 4.3.1 Protocol Architecture

The Protocol Handler contains the State Machine, Rx Shadow Register, Tx Shadow Register, Rx Shift Register, Tx Shift Register, and Loopback Multiplexer as shown in Figure 4-9 BDLC Protocol Handler Outline. Each block will now be described in more detail.



**Figure 4-9 BDLC Protocol Handler Outline**

- Rx & Tx Shift Registers

The Rx Shift Register gathers received serial data bits from the J1850 bus and makes them available in parallel form to the Rx Shadow Register. The Tx Shift Register takes data, in parallel form, from the Tx Shadow Register and presents it serially to the State Machine so that it can be transmitted onto the J1850 bus.

- Rx & Tx Shadow Registers



Immediately after the Rx Shift Register has completed shifting in a byte of data, this data is transferred to the Rx Shadow Register and RDRF or RXIFR is set and interrupt is generated if the interrupt enable bit (IE) in BCR1 is set. After the transfer takes place, this new data byte in the Rx Shadow Register is available to the CPU, and the Rx Shift Register is ready to shift in the next byte of data. Data in Rx Shadow Register must be retrieved by the CPU before it is overwritten by new data from the Rx Shift Register.

Once the Tx Shift Register has completed its shifting operation for the current byte, the data byte in the Tx Shadow Register is loaded into the Tx Shift Register. After this transfer takes place, the Tx Shadow Register is ready to accept new data from the CPU.

- Digital Loopback Multiplexer

The Digital Loopback Multiplexer connects the input of the receive digital filter (See Figure 4-9) to either the transmit signal out to the pad (TXB) or the receive signal from the pad (RXB), depending on the state of the DLOOP bit in DLCBCR2 register.

- State Machine

All of the functions associated with performing the protocol are executed or controlled by the State Machine. The State Machine is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. The following sections describe the BDLC module's actions in a variety of situations.

- 4X Mode

The BDLC module can exist on the same J1850 bus as modules which use a special 4X (41.6 kbps) mode of J1850 VPW operation. The BDLC module cannot transmit in 4X mode, but can receive messages in 4X mode, if the RX4X bit is set in BCR2 register. If the RX4X bit is not set in the BCR2 register, any 4X message on the J1850 bus is treated as noise by the BDLC module and is ignored. Likewise, 4X messages transmitted on the SAE J1850 bus when the BDLC module is in normal mode will be interpreted as noise on the network by the BDLC module.

- Receiving a Message in Block Mode

Although not a part of the SAE J1850 protocol, the BDLC module does allow for a special “Block Mode” of operation of the receiver. As far as the BDLC module is concerned, a Block Mode message is simply a long J1850 frame that contains an indefinite number of data bytes. All of the other features of the frame remain the same, including the SOF, CRC, and EOD symbols.

Another node wishing to send a Block Mode transmission must first inform all other nodes on the network that this is about to happen. This is usually accomplished by sending a special predefined message.

- Transmitting a Message in Block Mode

A Block mode message is transmitted inherently by simply loading the bytes one by one into the BDR register until the message is complete. The programmer should wait until the TDRE flag is set prior to writing a new byte of data into the BDR register. The BDLC module does not contain any predefined maximum J1850 message length requirement.

## 4.4 Transmitting A Message

The design of the BDLC module enables the user to easily handle message reception and message transmission separately. This can greatly simplify the communication software, as all received messages can be handled virtually the same, regardless of their origin.

This chapter will therefore describe only the steps necessary for transmitting a message, and will not address the resulting reception of that message by the BDLC module. Message reception is described in Section 4.5 Receiving A Message. Later sections will deal with transmitting and receiving In-Frame Responses on the SAE J1850 bus.

### 4.4.1 BDLC Transmission Control Bits

There is only one BDLC module control bit which is used when transmitting a message onto the SAE J1850 bus. This bit, the Transmit End of Data (TEOD) bit, is set by the user to indicate to the BDLC module that the last byte of that part of the message frame has been loaded into the DLCBDR. The TEOD bit, located in DLCBCR2, is also used when transmitting an In-Frame Response (IFR), but that usage is described in Section 4.6 Transmitting An In-Frame Response (IFR) on page 67. Setting the TEOD bit indicates to the BDLC module that the last byte written to the BDLC Data Register is the final byte to be transmitted, and that following this byte a CRC byte and EOD symbol should be transmitted automatically. Setting the TEOD bit will also inhibit any further TDRE interrupts until TEOD is cleared. The TEOD bit will be cleared on the rising edge of the first bit of the transmitted CRC byte, or if an error or loss of arbitration is detected on the bus.

- BDLC Data Register

The BDLC Data Register is a double-buffered register which is used for handling the transmitted and received message bytes. Bytes to be transmitted onto the SAE J1850 bus are written to the DLCBDR, and bytes received from the bus by the BDLC module are read from the DLCBDR. Since this register is double buffered, bytes written into it cannot be read by the CPU. If this is attempted, the byte which is read will be the last byte placed in the DLCBDR by the BDLC module, not the last byte written to the DLCBDR by the CPU. For an illustration of the DLCBDR, refer to Section 3.3.4 BDLC Data Register (DLCBDR) on page 31.

- Transmitting a Message with the BDLC

To transmit a message using the BDLC module, the user just writes the first byte of the message to be transmitted into the DLCBDR, initiating the transmission process. When the TDRE status appears in the DLCBSVR, the user writes the next byte into the DLCBDR. Once all of the bytes have been loaded into the DLCBDR, the user sets the TEOD bit, and the BDLC module completes the message transmission. What follows is an overview of the basic steps required to transmit a message onto an SAE J1850 network using the BDLC module. For an illustration of this sequence, refer to Figure 4-10 Basic BDLC Transmit Flowchart on page 62.

**NOTE:** Due to the byte-level architecture of the BDLC module, the 12-byte limit on message length as defined in SAE J1850 must be enforced by the user's software. The number of bytes in a message (transmitted or received) has no meaning to the BDLC module.

- Step 1: Write the First Byte into the DLCBDR

To initiate a message transmission, the CPU simply loads the first byte of the message to be transmitted into the DLCBDR. The BDLC module will then perform the necessary bus acquisition duties to determine when the message transmission can begin.

Once the BDLC module determines that the SAE J1850 bus is free, a Start of Frame (SOF) symbol will be transmitted, followed by the byte written to the DLCBDR. Once the BDLC module readies this byte for transmission, the DLCBSVR will reflect that the next byte can be written to the DLCBDR (TDRE interrupt).

**NOTE:** *If the user writes the first byte of a message to be transmitted to the DLCBDR and then determines that a different message should be transmitted, the user can write a new byte to the DLCBDR up until the transmission begins. This new byte will replace the original byte in the DLCBDR.*

- Step 2: When TDRE is Indicated, Write the Next Byte into the DLCBDR

When a TDRE state is reflected in the BSVR, the CPU writes the next byte to be transmitted into the BDR. This step is repeated until the last byte to be transmitted is written to the DLCBDR.

**NOTE:** *Due to the design and operation of the BDLC module, when transmitting a message the user may write two, or possibly even three of the bytes to be transmitted into the DLCBDR before the first RDRF interrupt occurs. For this reason, the user should never use receive interrupts to control the sequencing of bytes to be transmitted.*

- Step 3: Write the Last Byte to the DLCBDR and Set TEOD

Once the user has written the last byte to be transmitted into the DLCBDR, the user then sets the TEOD bit in DLCBCR2. When the TEOD bit is set, once the byte written to the DLCBDR is transmitted onto the bus, the BDLC module will begin transmitting the 8-bit CRC byte, as specified in SAE J1850. Following the CRC byte, the BDLC module will transmit an EOD symbol onto the SAE J1850 bus, indicating that this part of the message has been completed. If no IFR bytes are transmitted following the EOD, an EOF will be recognized and the message will be complete.

Setting the TEOD bit is the last step the CPU needs to take to complete the message transmission, and no further transmission-related interrupts will occur. Once the message has been completely received by the BDLC module, an EOF interrupt will be generated. However, this is technically a receive function which can be handled by the message reception routine.

**NOTE:** *While the TEOD bit is typically set immediately following the write of the last byte to the BDR, it is also acceptable to wait until a TDRE interrupt is generated before setting the TEOD bit. While the example flowchart in Figure 4-10 shows the TEOD bit being set after the write to the BDR, either method is correct. If a TDRE interrupt is pending, it will be cleared when the TEOD bit is set.*

## 4.4.2 Transmitting Exceptions

While this is the basic transmit flow, at times the message transmit process will be interrupted. This can be due to a loss of arbitration to a higher priority message or due to an error being detected on the network. For the transmit routine, either of these events can be dealt with in a similar manner.

- Loss of Arbitration

If a loss of arbitration (LOA) occurs while the BDLC module is transmitting onto the SAE J1850 bus, the BDLC module will immediately stop transmitting, and a LOA status will be reflected in the DLCBSVR. If the loss of arbitration has occurred on a byte boundary, an RDRF interrupt may also be pending once the LOA interrupt is cleared.

When a loss of arbitration occurs, the J1850 message handling software should immediately switch into the receive mode. If the TEOD bit was set, it will be cleared automatically. **If another attempt is to be made to transmit the same message, the user must start the transmit sequence over from the beginning of the message.**

- Error Detection

Similar to a loss of arbitration, if any error (except a CRC error) is detected on the SAE J1850 bus during a transmission, the BDLC module will stop transmitting immediately. The byte which was being transmitted will be discarded, and the “Symbol Invalid or Out of Range” status will be reflected in the DLCBSVR. As with the loss of arbitration, if the TEOD bit was set, it will be cleared automatically, and any attempt to transmit the same message will have to start from the beginning.

If a CRC error occurs following a transmission, this will also be reflected in the DLCBSVR. However, since the CRC error is really a receive error based on the received CRC byte, at this point all bytes of the message will have been transmitted. It is therefore up to the user’s software to determine if another attempt should be made to transmit the message in which the error occurred.

- Transmitter Underrun

A transmitter underrun can occur when a TDRE interrupt is not serviced in a timely fashion. If the last byte loaded into the DLCBDR is completely transmitted onto the network before the next byte is loaded into the BDR, a transmitter underrun will occur. If this does happen, the BDLC module will transmit two additional logic ones to ensure that the partial message which was transmitted onto the bus does not end on a byte boundary. This will be followed by an EOD and EOF symbol. The only indication to the CPU that an underrun occurred is the Symbol Invalid or Out of Range error which will be indicated in the DLCBSVR. As with the other errors, it is up to the user’s software to determine if another transmission attempt should be made.

- In-Frame Response to a Transmitted Message

If an In-Frame Response (IFR) is received following the transmission of a message, the status indicating that an IFR byte has been received will be indicated in the DLCBSVR before an EOF is indicated. Refer to Section 4.7 Receiving An In-Frame Response (IFR) on page 78 for a description of how to handle the reception of IFR bytes.

### 4.4.3 Aborting a Transmission

The BDLC module does not have a mechanism designed specifically for aborting a transmission. Since the module transmits each message on a byte-by-byte basis, there is little need to implement an abort mechanism. If the user has loaded a byte into the DLCBDR to initiate a message transmission and decides to send a different message, the byte in the DLCBDR can be replaced, right up to the point that the message transmission begins.

If the user has loaded a byte into the DLCBDR and then decides not to send any message at all, the user can let the byte transmit, and when the TDRE interrupt occurs let the transmitter underrun. This will cause two extra logic ones followed by an EOF to be transmitted. While this method may require a small amount of bus bandwidth, the need to do this should be very rare. Replacing the byte originally written to the BDR with \$FF will also increase the probability of the transmitter losing arbitration if another node begins transmitting at the same time, also reducing the bus bandwidth needed.

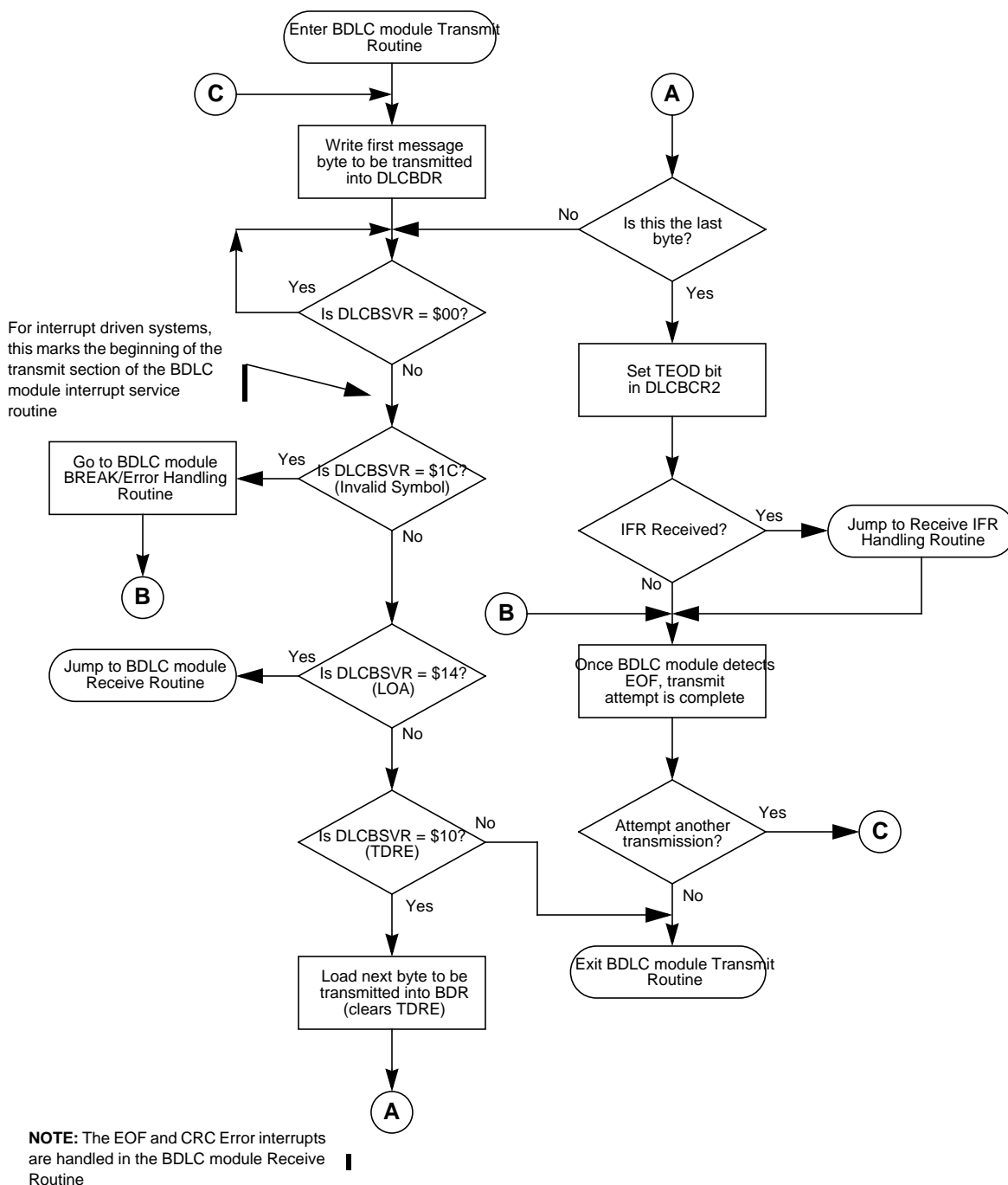


Figure 4-10 Basic BDLC Transmit Flowchart

## 4.5 Receiving A Message

The design of the BDLC module makes it especially easy to use for receiving messages off of the SAE J1850 bus. When the first byte of a message comes in, the DLCBSVR will indicate to the CPU that a byte

has been received. As each successive byte is received, that will in turn be reflected in the DLCBSVR. When the message is complete and the EOF has been detected on the bus, the DLCBSVR will reflect this, indicating that the message is complete.

The basic steps required for receiving a message from the SAE J1850 bus are outlined below. For more information on receiving IFR bytes, refer to Section 4.7 Receiving An In-Frame Response (IFR) on page 78.

### 4.5.1 BDLC Reception Control Bits

The only control bit which is used for message reception, the IMMSG bit, is actually used to prevent message reception. When the IMMSG bit is set BDLC module interrupts of the CPU are inhibited until the next SOF symbol is received. This allows the BDLC module to ignore the remainder of a message once the CPU has determined that it is of no interest. This helps reduce the amount of CPU overhead used to service messages received from the SAE J1850 network, since otherwise the BDLC module would require attention from the CPU for each byte broadcast on the network. The IMMSG bit is cleared when the BDLC module receives an SOF symbol, or it can also be cleared by the CPU.

**NOTE:** While the IMMSG bit can be used to prevent the CPU from having to service the BDLC module for every byte transmitted on the SAE J1850 bus, **the IMMSG bit should never be used to ignore the BDLC module's own transmission.** Because setting the IMMSG bit prevents all DLCBSVR bits from being updated until an SOF is received, the user would not receive any further transmit-related interrupts until another SOF was received, making it very difficult for the CPU to complete the transmission correctly.

### 4.5.2 Receiving a Message with the BDLC module

Receiving a message using the BDLC module is extremely straight-forward. As each byte of a message is received and placed into the DLCBDR, the BDLC module will indicate this to the CPU with an Rx Data Register Full (RDRF) status in the DLCBSVR. When an EOF symbol is received, indicating to the CPU that the message is complete, this too will be reflected in the DLCBSVR.

Outlined below are the basic steps to be followed for receiving a message from the SAE J1850 bus with the BDLC module. For an illustration of this sequence, refer to Basic BDLC Receive Flowchart on page 66.

- Step 1: When RDRF Interrupt Occurs, Retrieve Data Byte

When the first byte of a message following a valid SOF symbol is received that byte is placed in the DLCBDR, and an RDRF state is reflected in the DLCBSVR. No indication of the SOF reception is made, since the end of the previous message is marked by an EOF indication. The first RDRF state following this EOF indication should allow the user to determine when a new message begins.

The RDRF interrupt is cleared when the received byte is read from the DLCBDR. Once this is done, no further CPU intervention is necessary until the next byte is received, and this step is repeated.

All bytes of the message, including the CRC byte, will be placed into the DLCBDR as they are received for the CPU to retrieve.

- Step 2: When an EOF is Received, the Message is Complete

Once all bytes (including the CRC byte) have been received from the bus, the bus will be idle for a time period equal to an EOD symbol. Once the EOD symbol is received, the BDLC module will verify that the CRC byte is correct. If the CRC byte is not correct, this will be reflected in the DLCBSVR.

If no In-Frame Response bytes are transmitted following the EOD symbol, the EOD will transition into an EOF symbol. When the EOF is received it will be reflected in the DLCBSVR, indicating to the user that the message is complete. If IFR bytes do follow the first EOD symbol, once they are complete another EOD will be transmitted, followed by an EOF.

Once the EOF state is reflected in the DLCBSVR, this indicates to the user that the message is complete, and that when another byte is received it is the first byte of a new message.

### 4.5.3 Filtering Received Messages

No message filtering hardware is included on the BDLC module, so all message filtering functions must be performed in software. Because the BDLC module handles each message on a byte-by-byte basis, message filtering can be done as each byte is received, rather than after the entire message is complete. This enables the CPU to decide while a message is still in progress whether or not that message is of any interest.

At any point during a message, if the CPU determines that the message is of no interest the IMMSG bit can be set. Setting the IMMSG bit commands the BDLC module not to update the DLCBSVR until the next valid SOF is received. This prevents the CPU from having to service the BDLC module for each byte of every message sent over the network.

### 4.5.4 Receiving Exceptions

As with a message transmission, this basic message reception flow can be interrupted if errors are detected by the BDLC module. This can occur if an incorrect CRC is detected or if an invalid or out of range symbol appears on the SAE J1850 bus. A problem can also arise if the CPU fails to service the DLCBDR in a timely manner during a message reception.

- Receiver Overrun

Once a message byte has been received, the CPU must service the DLCBDR before the next byte is received, or the first byte will be lost. If the DLCBDR is not serviced quickly enough, the next byte received will be written over the previous byte in the DLCBDR. No receiver overrun indication is made to the CPU. If the CPU fails to service the BDLC module during the reception of an entire message, the byte remaining in the DLCBDR will be last byte received (usually a CRC byte).

Once a receiver overrun occurs, there is no way for the CPU to recover the lost byte(s), so the entire message should be discarded. To prevent receiver overrun, the user should ensure that a BDLC RDRF interrupt will be serviced before the next byte can be received. When polling the DLCBSVR, the user should select a polling interval which will provide timely monitoring of the BDLC module.

- CRC Error



If a CRC error is detected during a message reception, this will be reflected in the BSVR once an EOD time is recognized by the BDLC module. Since all bytes of the message will have been received when this error is detected, it is up to the user to ensure that all the received message bytes are discarded.

- Invalid or Out of Range Symbol

If an invalid or out of range symbol, a framing error or a BREAK symbol is detected on the SAE J1850 bus during the reception of a message, the BDLC module will immediately stop receiving the message and discard any partially received byte. The “Symbol Invalid or Out of Range” status will immediately be reflected in the DLCBSVR. Following this the BDLC module will wait until the bus has been idle for a time period equal to an EOF symbol before receiving another message. As with the CRC error, the user should discard any partially received message if this occurs.

- In-Frame Response to a Received Message

As mentioned above, if one or more IFR bytes are received following the reception of a message, the status indicating the reception of the IFR byte(s) will be indicated in the DLCBSVR before the EOF is indicated. Refer to Receiving An In-Frame Response (IFR) on page 78 for a description of how to deal with the reception of IFR bytes.

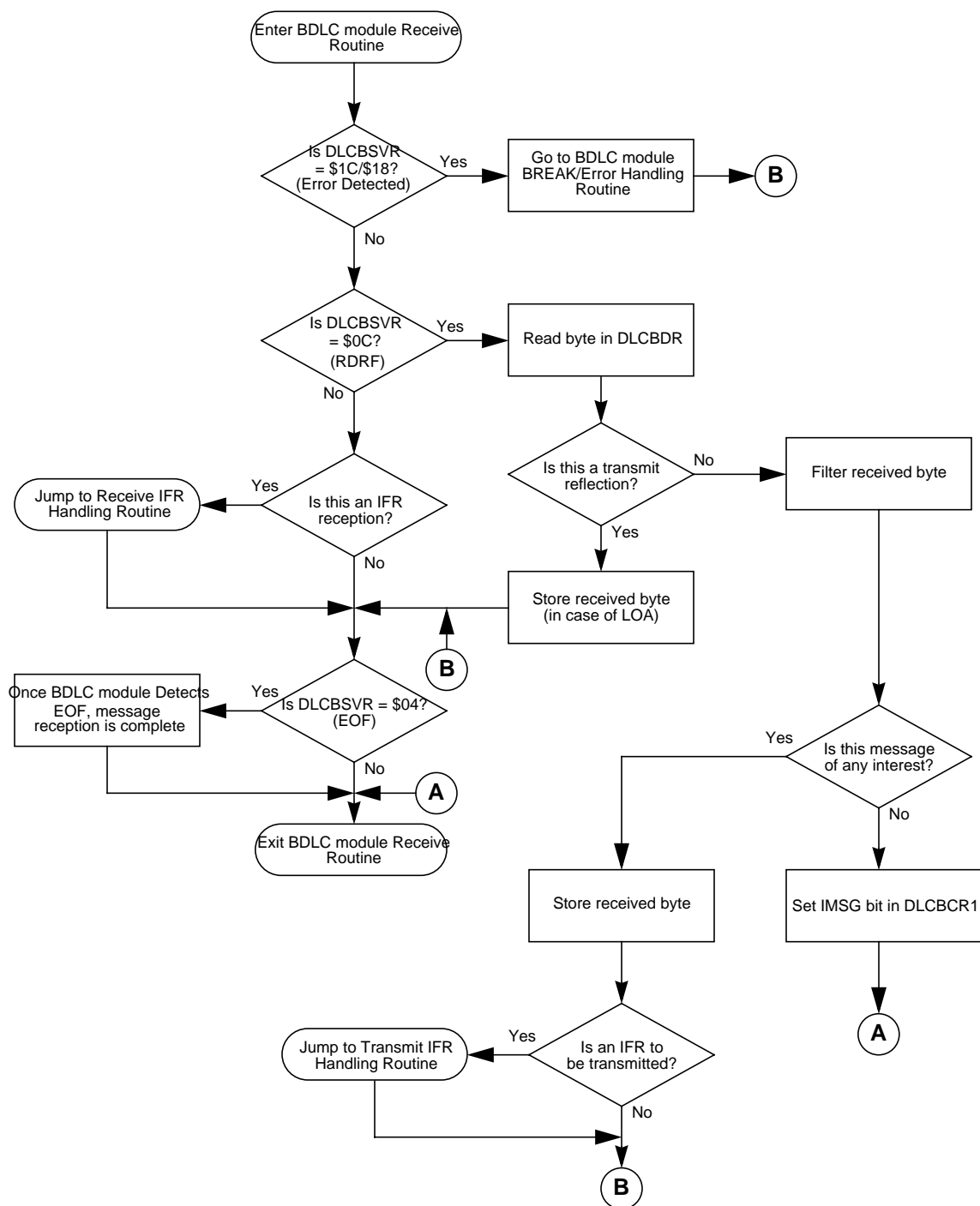


Figure 4-11 Basic BDLC Receive Flowchart

## 4.6 Transmitting An In-Frame Response (IFR)

The BDLC module can be used to transmit all four types of In-Frame Response (IFR) which are defined in SAE J1850. A very brief definition of each IFR type is given below. For a more detailed description of each, refer the SAE J1850 document.

The explanation regarding IFR support by the BDLC module which assumes the user is familiar with the use of IFRs as defined in SAE J1850, and understands the message header bit encoding and normalization bit formats which are used with the different types of IFRs. For more information on this, refer to the SAE J1850 document.

### 4.6.1 IFR Types Supported by the BDLC module

SAE J1850 defines four distinct types of IFR. The first (and most basic) IFR is Type 0, or no IFR. IFR types 1, 2 & 3 are each made up of one or more bytes and, depending upon the type used, may be followed by a CRC byte. The BDLC module is designed to allow the user to transmit and receive all types of SAE J1850 IFRs, but only the network framing/error checking/bus acquisition duties are performed by the BDLC module. The user is responsible for determining the type of IFR to be transmitted, the number of retries to be made (if allowed), and the allowable number of bytes to be transmitted.

- IFR Type 0: No Response

The most basic type of IFR is no IFR. The Type 0 IFR, as defined in SAE J1850, is no response. The EOD and EOF symbols follow directly after the CRC byte at the end of the message frame being transmitted. This type of IFR is, of course, inherently supported by the BDLC module with no additional user intervention required.

- IFR Type 1: Single Byte from a Single Responder

SAE J1850 defines the Type 1 IFR as a single byte from a single receiver. This type of IFR is used to acknowledge to the transmitter that the message frame was transmitted successfully on the network, and that at least one receiver received it correctly. A Type 1 IFR generally consists of the physical node ID of the receiver responding to the message, with no CRC byte appended. This type of response is used for Broadcast-type messages, where there may be several intended receivers for a message but the transmitter only wants to know that at least one node received it. In this case, all receivers will begin transmitting their node ID following the EOD. Since all nodes on an SAE J1850 network have a unique node ID, if multiple nodes begin transmitting their node ID simultaneously, arbitration takes place. The node with the highest priority (lowest value) ID wins this arbitration process, and that node's ID makes up the IFR. No retries are attempted by the nodes which lose arbitration during a Type 1 IFR transmission.

A Type 1 IFR can also be used as a response to a physically addressed message, where the only intended receiver is the one which responds. In this case, no arbitration would take place during the IFR transmission, but the resulting IFR would still consist of a single byte.

- IFR Type 2: Single Byte from Multiple Responders

The Type 2 IFR, as defined in SAE J1850, is a series of single bytes, each transmitted by a different responder. This IFR type not only acknowledges to the transmitter that the message was transmitted successfully, but also reveals which receivers actually received the message. As with the Type 1 IFR, no CRC byte is appended to the end of a Type 2 IFR.

This IFR type is typically used with Function-type messages, where the original transmitter may need to know which nodes actually received the message. The basic difference between this type of IFR and the Type 1 IFR is that the nodes which lose arbitration while attempting to transmit their node ID during a Type 2 IFR wait until the byte which wins arbitration is transmitted and then again attempt to transmit their node ID onto the bus. The result is a series of node IDs, one from each receiver of the original message.

- IFR Type 3: Multiple Bytes from a Single Responder

The last type of IFR defined by SAE J1850 is the Type 3 IFR. This IFR type consists of one or more bytes from a single responder. This type of IFR is used to return data to the original transmitter within the original message frame. This type of IFR may or may not have a CRC byte appended to it.

The Type 3 IFR is typically used with Function Read-type or Function Query-type messages, where the original transmitter is requesting data from the intended receiver. The node requesting the data transmits the initial portion of the message, and the intended receiver responds by transmitting the desired data in an IFR. In most cases, the original message requiring a Type 3 IFR is addressed to one particular node, so no arbitration should take place during the IFR portion of the message.

#### 4.6.2 BDLC IFR Transmit Control Bits

The BDLC module has three bits which are used to control the transmission of an In-Frame Response. These bits, all located in DLCBCR2, are TSIFR, TMIFR1 and TMIFR0. Each is used in conjunction with the TEOD bit to transmit one of three IFR types defined in SAE J1850. What follows is a brief description of each bit.

Because each of the bits used for transmitting an IFR with the BDLC module is used to transmit a particular type of IFR, only one bit should be set by the CPU at a time. However, should more than one of these bits get set at one time, a priority encoding scheme is used to determine which type of IFR is sent. This scheme prevents unpredictable operation caused by conflicting signals to the BDLC module. Table 4-8 illustrates which IFR bit will actually be acted upon by the BDLC module should multiple IFR bits get set at the same time.

**NOTE:** As with transmitted messages, IFRs transmitted by the BDLC module will also be received by the BDLC module. For a description of how IFR bytes received by the BDLC module should be handled, refer to Section 4.7 Receiving An In-Frame Response (IFR) on page 78.

Table 4-8 IFR Control Bit Priority Encoding

READ/WRITE			ACTUAL		
TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0
0	0	0	0	0	0
1	X	X	1	0	0
0	1	X	0	1	0
0	0	1	0	0	1

### 4.6.3 Transmit Single Byte IFR

The Transmit Single Byte IFR (TSIFR) bit in DLCBCR2 is used to transmit Type 1 and Type 2 IFRs onto the SAE J1850 bus. If this bit is set after a byte is loaded into the BDR, the BDLC module will attempt to send that byte, preceded by the appropriate Normalization Bit, as a single byte IFR without a CRC. If arbitration is lost, the BDLC module will automatically attempt to transmit the byte again (without a Normalization Bit) as soon as the byte winning arbitration completes transmission. Attempts to transmit the byte will continue until either the byte is successfully transmitted, the TEOD bit is set by the user or an error is detected on the bus.

The user must set the TSIFR bit before the EOD following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node does transmit an IFR to this message or a reception error occurs, the TSIFR bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.

The TSIFR bit will be automatically cleared once the EOD following one or more IFR bytes has been received or an error is detected on the bus.

### 4.6.4 Transmit Multi-Byte IFR 1

The Transmit Multi-Byte IFR 1 (TMIFR1) bit is used to transmit an SAE J1850 Type 3 IFR with a CRC byte appended. If this bit is set after the user has loaded the first byte of a multi-byte IFR into the DLCBDR, the BDLC module will begin transmitting that byte, preceded by the appropriate Normalization Bit, onto the SAE J1850 bus. Once this happens a TDRE interrupt will occur, indicating to the user that the next IFR byte should be loaded into the DLCBDR. When the last byte to be transmitted is written to the DLCBDR, the user sets the TEOD bit. This will cause a CRC byte and an EOD symbol to be transmitted following the last IFR byte.

As with the TSIFR bit, the TMIFR1 bit must be set before the EOD symbol is received, or it will remain cleared and no IFR transmit attempt will be made. The TMIFR1 bit will be cleared once the CRC byte and EOD are transmitted, if an error is detected on the bus, if a loss of arbitration occurs during the IFR transmission or if a transmitter underrun occurs when the user fails to service the TDRE interrupt in a timely manner. If a loss of arbitration occurs while the Type 3 IFR is being transmitted, transmission will halt immediately and the loss of arbitration will be indicated in the DLCBSVR.

## 4.6.5 Transmit Multi-Byte IFR 0

The Transmit Multi-Byte IFR 0 (TMIFR0) bit is used to transmit an SAE J1850 Type 3 IFR without a CRC byte appended. If this bit is set after the user has loaded the first byte of a multi-byte IFR into the DLCBDR, the BDLC module will begin transmitting that byte, preceded by the appropriate Normalization Bit, onto the SAE J1850 bus. Once this happens a TDRE interrupt will occur, indicating to the user that the next IFR byte should be loaded into the DLCBDR. When the last byte to be transmitted is written to the DLCBDR, the user sets the TEOD bit. This will cause an EOD symbol to be transmitted following the last IFR byte.

As with the TSIFR and TMIFR1 bits, the TMIFR0 bit must be set before the EOD symbol is received, or it will remain cleared and no IFR transmit attempt will be made. The TMIFR0 bit will be cleared once the CRC byte and EOD are transmitted, if an error is detected on the bus, if a loss of arbitration occurs during the IFR transmission or if a transmitter underrun occurs when the user fails to service the TDRE interrupt in a timely manner. If a loss of arbitration occurs while the Type 3 IFR is being transmitted, transmission will halt immediately and the loss of arbitration will be indicated in the DLCBSVR.

**NOTE:** *The TMIFR0 bit should not be used to transmit a Type 1 IFR. If a loss of arbitration occurs on the last bit of a byte being transmitted using the TMIFR0 bit, two extra logic ones will be transmitted to ensure that the IFR will not end on a byte boundary. This can cause an error in a Type 1 IFR.*

## 4.6.6 Transmitting An IFR with the BDLC module

While the design of the BDLC module makes the transmission of each type of IFR similar, the steps necessary for sending each will be discussed. Again, a discussion of the bytes making up any particular IFR is not within the scope of this document. For a more detailed description of the use of IFRs on an SAE J1850 network, refer to the SAE J1850 document.

- Transmitting a Type 1 IFR

To transmit a Type 1 IFR, the user loads the byte to be transmitted into the DLCBDR and sets both the TSIFR bit and the TEOD bit. This will direct the BDLC module to attempt transmitting the byte written to the DLCBDR one time, preceded by the appropriate Normalization Bit. If the transmission is not successful, the byte will be discarded and no further transmission attempts will be made. For an illustration of the steps described below, refer to Section 4-12 Transmitting A Type 1 IFR on page 72.

- Step 1: Load the IFR Byte into the BDR

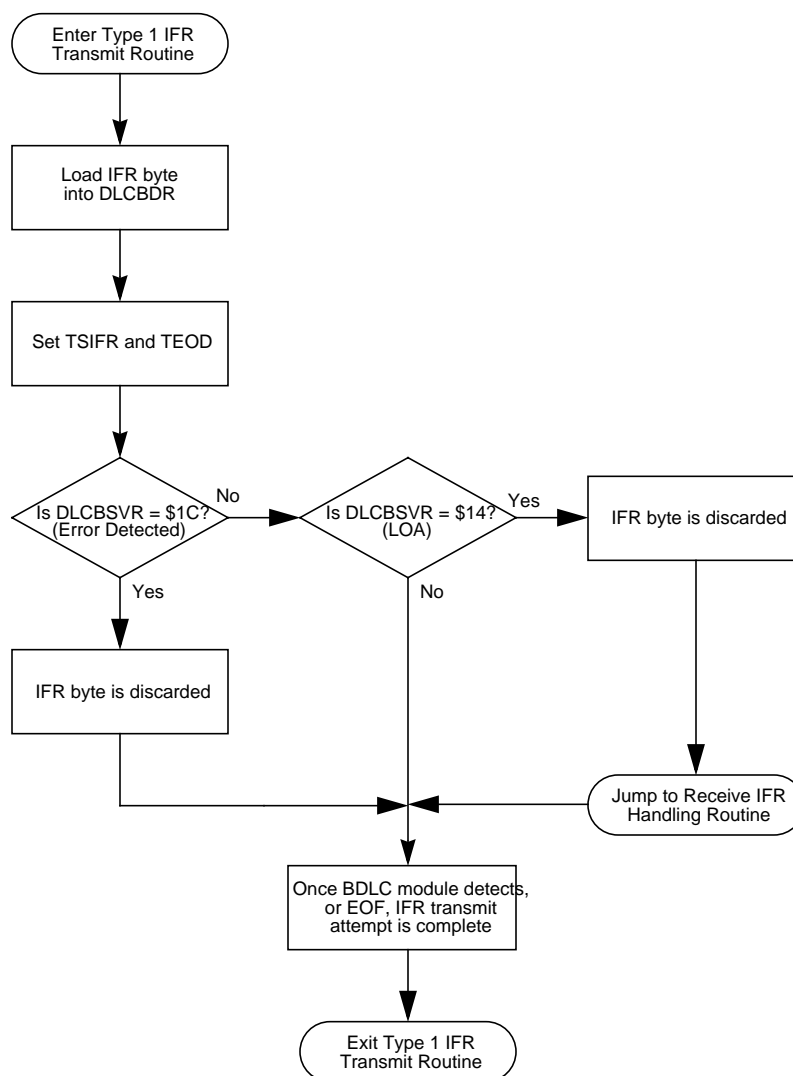
The user begins initiation of a Type 1 IFR by loading the desired IFR byte into the DLCBDR. If a byte has already been written into the DLCBDR for transmission as a new message, the user can simply write the IFR byte to the DLCBDR, replacing the previously written byte. This must be done before the first EOD symbol is received.

- Step 2: Set the TSIFR and TEOD Bits

The final step in transmitting a Type 1 IFR with the BDLC module is to set the TSIFR and TEOD bits in DLCBCR2. Setting both bits will direct the BDLC module to make one attempt at transmitting the byte in the DLCBDR as an IFR. If the byte is transmitted successfully, or if an error or loss of arbitration occurs, TEOD and TSIFR will be cleared and no further transmit attempts will be made.

- Transmitting a Type 2 IFR

To transmit a Type 2 IFR, the user loads the byte to be transmitted into the DLCBDR and sets the TSIFR bit. Once this is done, the BDLC module will attempt to transmit the byte in the DLCBDR as a single byte IFR, preceded by the appropriate Normalization Bit. If the first BDLC module loses arbitration on the first attempt, it will make repeated attempts to transmit this byte until it is successful, an error occurs or the user sets the TEOD bit.



**Figure 4-12 Transmitting A Type 1 IFR**

- Step 1: Load the IFR Byte into the BDR

As with the Type 1 IFR, the user begins initiation of a Type 2 IFR by loading the desired IFR byte into the DLCBDR. If a byte has already been written into the DLCBDR for transmission as a new message, the user can simply write the IFR byte to the DLCBDR, replacing the previously written byte. This must be done before the first EOD symbol is received.

- Step 2: Set the TSIFR Bit



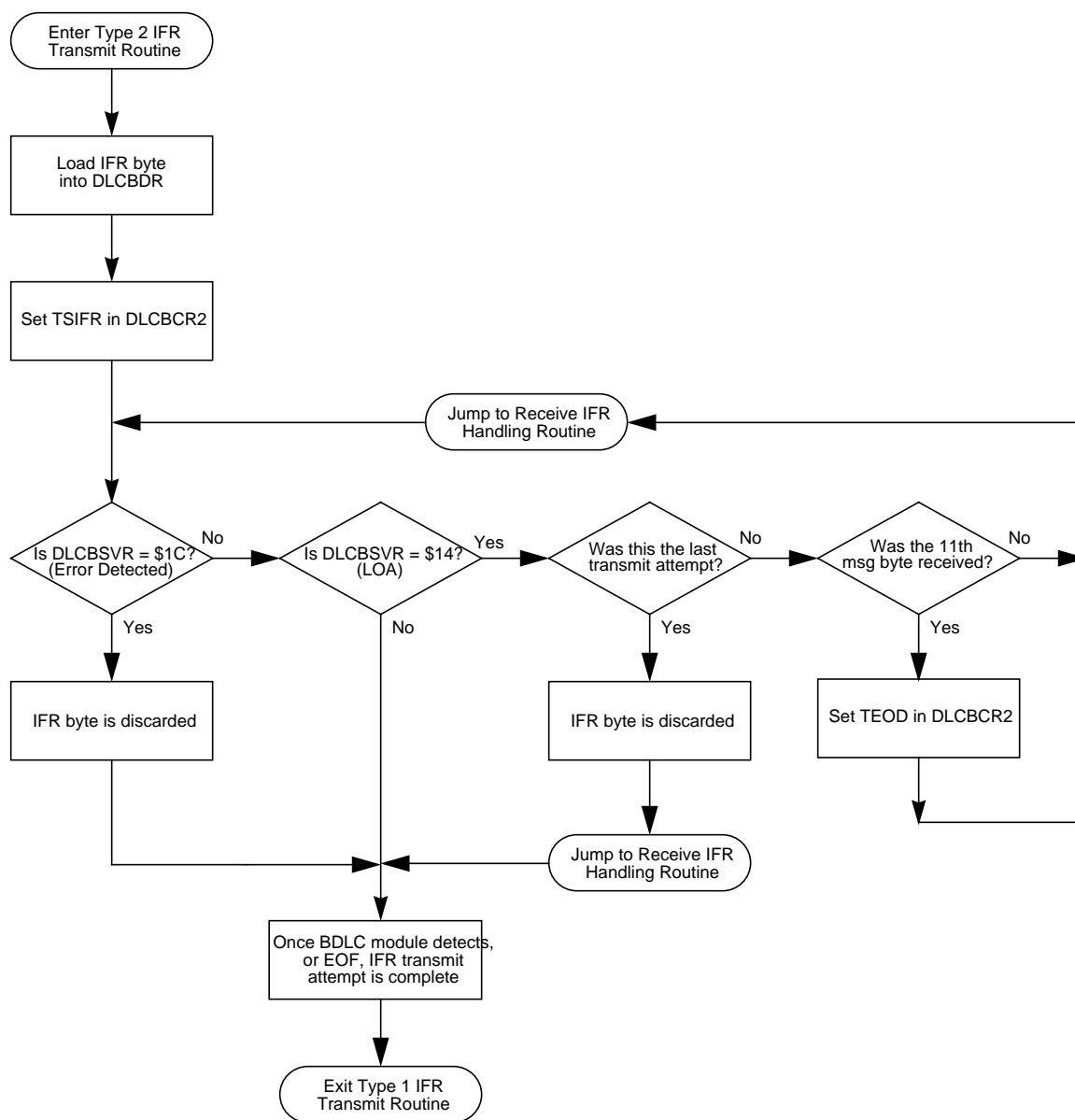
The second step necessary for transmitting a Type 2 IFR is to set the TSIFR bit in DLCBCR2. Setting this bit will direct the BDLC module to attempt to transmit the byte in the DLCBDR as an IFR until it is successful. If the byte is transmitted successfully, or if an error or loss of arbitration occurs, TSIFR will be cleared and no further transmit attempts will be made.

– Step 3: If Necessary, Set the TEOD Bit

The third step in transmitting a Type 2 IFR is only necessary if the user wishes to halt the transmission attempts. This may be necessary if the BDLC module's attempt to transmit the byte loaded into the DLCBDR continually loses arbitration, and the overall message length approaches the 12-byte limit as defined in SAE J1850.

If it becomes necessary to halt the IFR transmission attempts, the user simply sets the TEOD bit in BCR2. If the BDLC module is between transmission attempts, it will make one more attempt to transmit the IFR byte. If it is transmitting the byte when TEOD is set, the BDLC module will continue the transmission until it is successful or it loses arbitration to another transmitter. At this point it will then discard the byte and make no more transmit attempts.

**NOTE:** *When transmitting a Type 2 IFR, the user should monitor the number of IFR bytes received to ensure that the overall message length does not exceed the 12-byte limit for the length of SAE J1850 messages. The user should set the TEOD bit when the 11th byte is received, which will prevent the 12-byte limit from being exceeded.*



**Figure 4-13 Transmitting A Type 2 IFR**

- Transmitting a Type 3 IFR

Transmitting a Type 3 IFR, with or without a CRC byte, is done in a fashion similar to transmitting a message frame. The user loads the first byte to be transmitted into the DLCBDR and then sets the appropriate TMIFR bit, depending upon whether a CRC byte is desired. When the last byte is written to the BDR, the TEOD bit is set, and a CRC byte (if desired) and an EOD are then transmitted. Because the two versions of the Type 3 IFR are transmitted identically, the description which follows will discuss both. For an illustration of the Type 3 IFR transmit sequence, refer to Figure 4-14 Transmitting A Type 3 IFR on page 77.

- Step 1: Load the First IFR Byte into the DLCBDR

The user begins initiation of a Type 3 IFR, as with each of the other IFR types, by loading the desired IFR byte into the DLCBDR. If a byte has already been written into the DLCBDR for transmission as a new message, the user can simply write the first IFR byte to the DLCBDR, replacing the previously written byte. This must be done before the first EOD symbol is received.

- Step 2: Set the TMIFR Bit

The second step necessary for transmitting a Type 3 IFR is to set the desired TMIFR bit in DLCBCR2, depending upon whether or not a CRC is desired. As previously described in Section 4.6.2 BDLC IFR Transmit Control Bits on page 68, the TMIFR1 bit should be set if the user requires a CRC byte to be appended following the last byte of the Type 3 IFR, and TMIFR0 if no CRC byte is required.

Setting the TMIFR1 or TMIFR0 bit will direct the BDLC module to transmit the byte in the BDR as the first byte of a single or multi-byte IFR preceded by the appropriate Normalization Bit. Once this has occurred, the DLCBSVR will reflect that the next byte of the IFR can be written to the DLCBDR (TDRE interrupt).

**NOTE:** *The user must set the TMIFR1 or TMIFR0 bit before the EOD following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node does transmit an IFR to this message or a reception error occurs, the TMIFR1 or TMIFR0 bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.*

- Step 3: When TDRE is Indicated, Write the Next IFR Byte into the DLCBDR

When a TDRE state is reflected in the DLCBSVR, the CPU writes the next IFR byte to be transmitted into the DLCBDR, clearing the TDRE interrupt. This step is repeated until the last IFR byte to be transmitted is written to the DLCBDR.

**NOTE:** *As when transmitting a message, when transmitting a Type 3 IFR the user may write two, or possibly even three of the bytes to be transmitted into the DLCBDR before the first RxIFR interrupt occurs. For this reason, the user should never use receive IFR byte interrupts to control the sequencing of IFR bytes to be transmitted.*

- Step 4: Write the Last IFR Byte into the DLCBDR and Set TEOD

Once the last IFR byte to be transmitted is written to the DLCBDR, the CPU then sets the TEOD bit in DLCBCR2. Once the TEOD bit is set, after the last IFR byte written to the DLCBDR is transmitted onto the bus, if the TMIFR1 bit has been set the BDLC module will begin

transmitting the CRC byte, followed by an EOD. If the TMIFR0 bit has been set, the last IFR byte will immediately be followed by the transmission of an EOD. Following the EOD, and EOF will be recognized and the message will be complete.

If at any time during the transmission of a Type 3 IFR a loss of arbitration occurs, the TMIFR bit which is set and the TEOD bit (if set) will be cleared, any IFR byte being transmitted will be discarded and the loss of arbitration state will be reflected in the DLCBSVR. Likewise, if an error is detected during the transmission of a Type 3 IFR the IFR control bits will be cleared, the byte being transmitted will be discarded and the DLCBSVR will reflect the detected error.

**NOTE:** *If the Type 3 IFR being transmitted is made up of a single byte, the appropriate TMIFR bit and the TEOD bit can be set at the same time. The BDLC module will then treat that byte as both the first and last IFR byte to be sent.*

#### 4.6.7 Transmitting IFR Exceptions

This basic IFR transmitting flow can be interrupted for the same reasons as a normal message transmission. The IFR transmit process can be adversely affected due to a loss of arbitration, an Invalid or Out of Range Symbol, or due to a transmitter underrun caused by the CPU failing to service a TDRE interrupt in a timely fashion. For a description of how these exceptions can affect the IFR transmit process, refer to Section 4.4.2 Transmitting Exceptions on page 60.

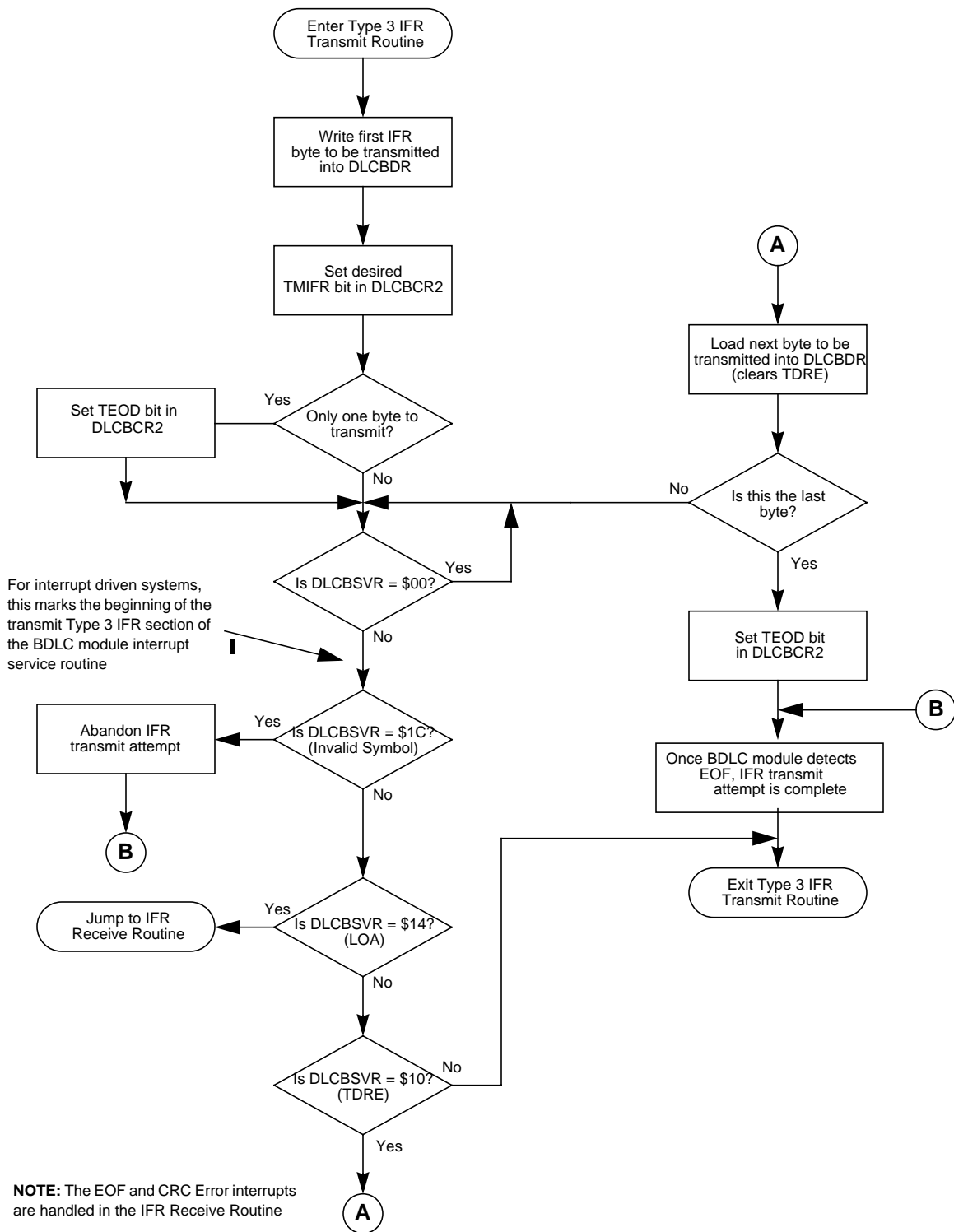


Figure 4-14 Transmitting A Type 3 IFR

## 4.7 Receiving An In-Frame Response (IFR)

Receiving an In-Frame Response with the BDLC module is very similar to receiving a message frame. As each byte of an IFR is received, the DLCBSVR will indicate this to the CPU. An EOF indication in the DLCBSVR indicates that the IFR (and message) is complete. Also, the IMMSG bit can also be used to command the BDLC module to mask any further network activity from the CPU, including IFR bytes being received, until the next valid SOF is received.

**NOTE:** *As with a message transmission, the IMMSG bit should never be used to ignore the BDLC module's own IFR transmissions. This is again due to the DLCBSVR bits being inhibited from updating until IMMSG is cleared, preventing the CPU from detecting any IFR-related state changes which may be of interest.*

### 4.7.1 Receiving an IFR with the BDLC module

Receiving an IFR from the SAE J1850 bus requires the same procedure that receiving a message does, except that as each byte is received the Received IFR Byte (RxIFR) state is indicated in the DLCBSVR. All other actions are the same. For an illustration of the steps described below, refer to Figure 4-15 Receiving An IFR With the BDLC module on page 79.

- Step 1: When RxIFR Interrupt Occurs, Retrieve IFR Byte

When the first byte of an IFR following a valid EOD symbol is received that byte is placed in the DLCBDR, and an RxIFR state is reflected in the DLCBSVR. No indication of the EOD reception is made, since the RxIFR state will indicate that the main portion of the message has ended and the IFR portion has begun.

The RxIFR interrupt is cleared when the received IFR byte is read from the DLCBDR. Once this is done, no further CPU intervention is necessary until the next IFR byte is received, and this step is repeated. As with a message reception, all bytes of the IFR, including the CRC byte, will be placed into the DLCBDR as they are received for the CPU to retrieve.

- When an EOF is Received, the IFR (and Message) is Complete

Once all IFR bytes (including the possible CRC byte) have been received from the bus, the bus will again be idle for a time period equal to an EOD symbol. Following this, the BDLC module will determine whether or not the last byte of the IFR is a CRC byte, and if so verify that the CRC byte is correct. If the CRC byte is not correct, this will be reflected in the DLCBSVR.

After an additional period of time the EOD symbol will transition into an EOF symbol. When the EOF is received it will be reflected in the DLCBSVR, indicating to the user that the IFR, and the message, is complete.

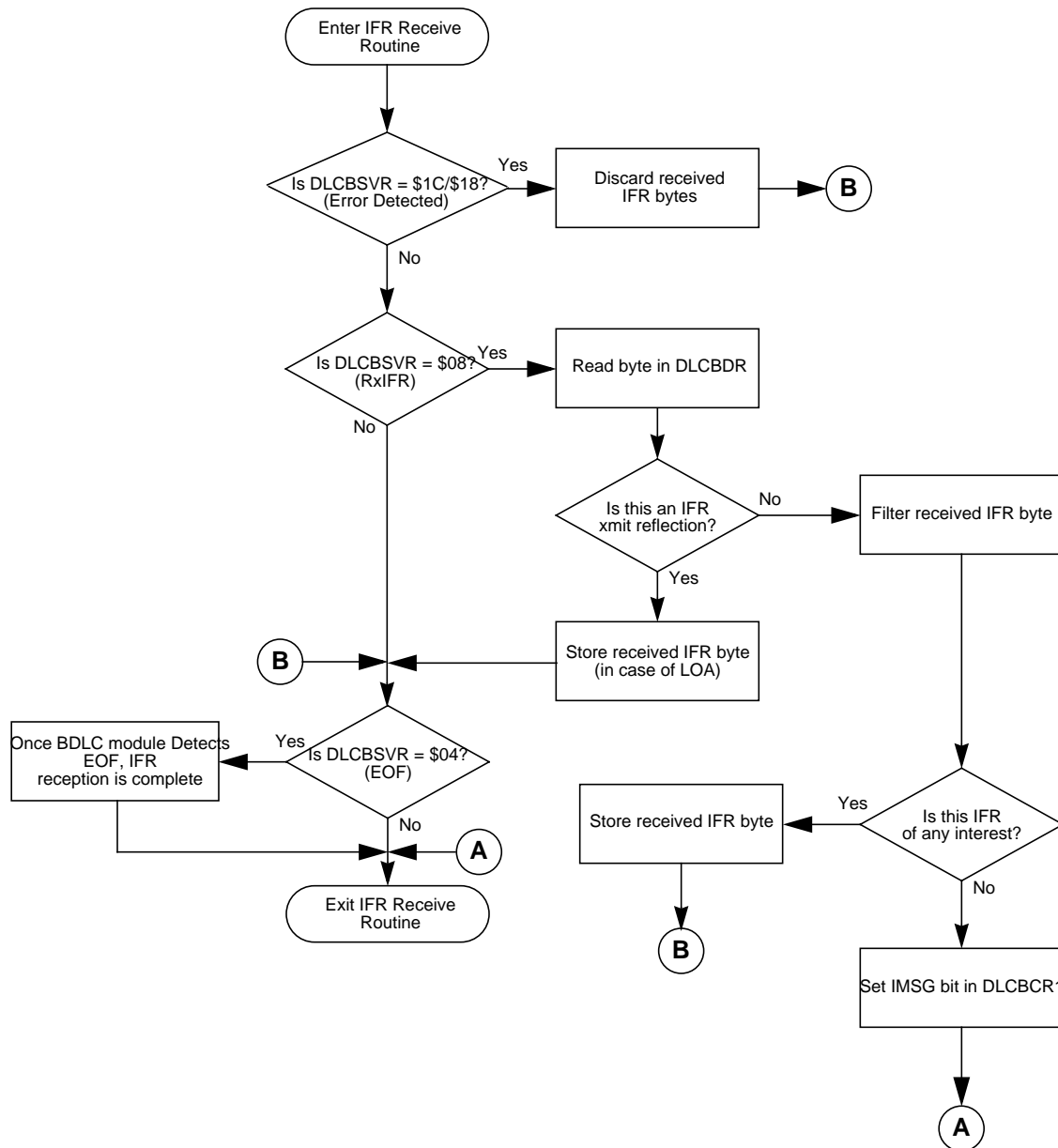


Figure 4-15 Receiving An IFR With the BDLC module

### 4.7.2 Receiving IFR Exceptions

This basic IFR receiving flow can be interrupted for the same reasons as a normal message reception. The IFR receiving process can be adversely affected due to a CRC error, an Invalid or Out of Range Symbol or due to a receiver overrun caused by the CPU failing to service an RxIFR interrupt in a timely fashion.

For a description of how these exceptions can affect the IFR receiving process, refer to Section 4.5.4 Receiving Exceptions on page 64.

## 4.8 Special BDLC Module Operations

There are a few special operations which the BDLC module can perform. What follows is a brief description of each of these functions and when they might be used.

### 4.8.1 Transmitting Or Receiving A Block Mode Message

The BDLC module, because it handles each message on a byte-by-byte basis, has the inherent capability of handling messages any number of bytes in length. While during normal operation this requires the user to carefully monitor message lengths to ensure compliance with SAE J1850 message limits, often in a production or diagnostic environment messages which exceed the SAE J1850 limits can be beneficial. This is especially true when large amounts of configuration data need to be downloaded over the SAE J1850 network.

Because of the BDLC module's architecture, it can both transmit and receive messages of unlimited length. The CRC calculations, both for transmitting and receiving, are not limited to eight bytes, but will instead be calculated and verified using all bytes in the message, regardless of the number. All control bits, including TEOD and MSG, also work in an identical manner, regardless of the length of the message.

To transmit or receive these "Block Mode" messages, no extra BDLC module control functions must be performed. The user simply transmits or receives as many bytes as desired in one message frame, and the BDLC module will operate just as if a message of normal length was being used.

### 4.8.2 Receiving A Message In 4X Mode

In a diagnostic or production environment large amounts of data may need to be downloaded across the network to a component or module. This data is often sent in a large "Block Mode" message (see above) which violates the SAE J1850 limit for message length. In order to speed up the downloading of these large blocks of data, they are sometimes transmitted at four times (4X) the normal bit rate for the Variable Pulse Width modulation version of SAE J1850. This higher speed transmission, nominally 41.6kbps, allows these large blocks to be transmitted much more quickly.

The BDLC module is designed to receive (but not transmit) messages transmitted at this higher speed. By setting the RX4XE bit in DLCBCR2, the user can command the BDLC module to receive any message sent over the network at a 4X rate.

If the BDLC module is placed in this 4X mode, messages transmitted at the normal bit rate will not be received correctly. Likewise, 4X messages transmitted on the SAE J1850 bus when the BDLC module is in normal mode will be interpreted as noise on the network by the BDLC module. The RX4XE bit is not affected by entry or exit from BDLC module stop or wait modes. For more information on the RX4XE bit, refer to Section • 4X Mode on page 57.



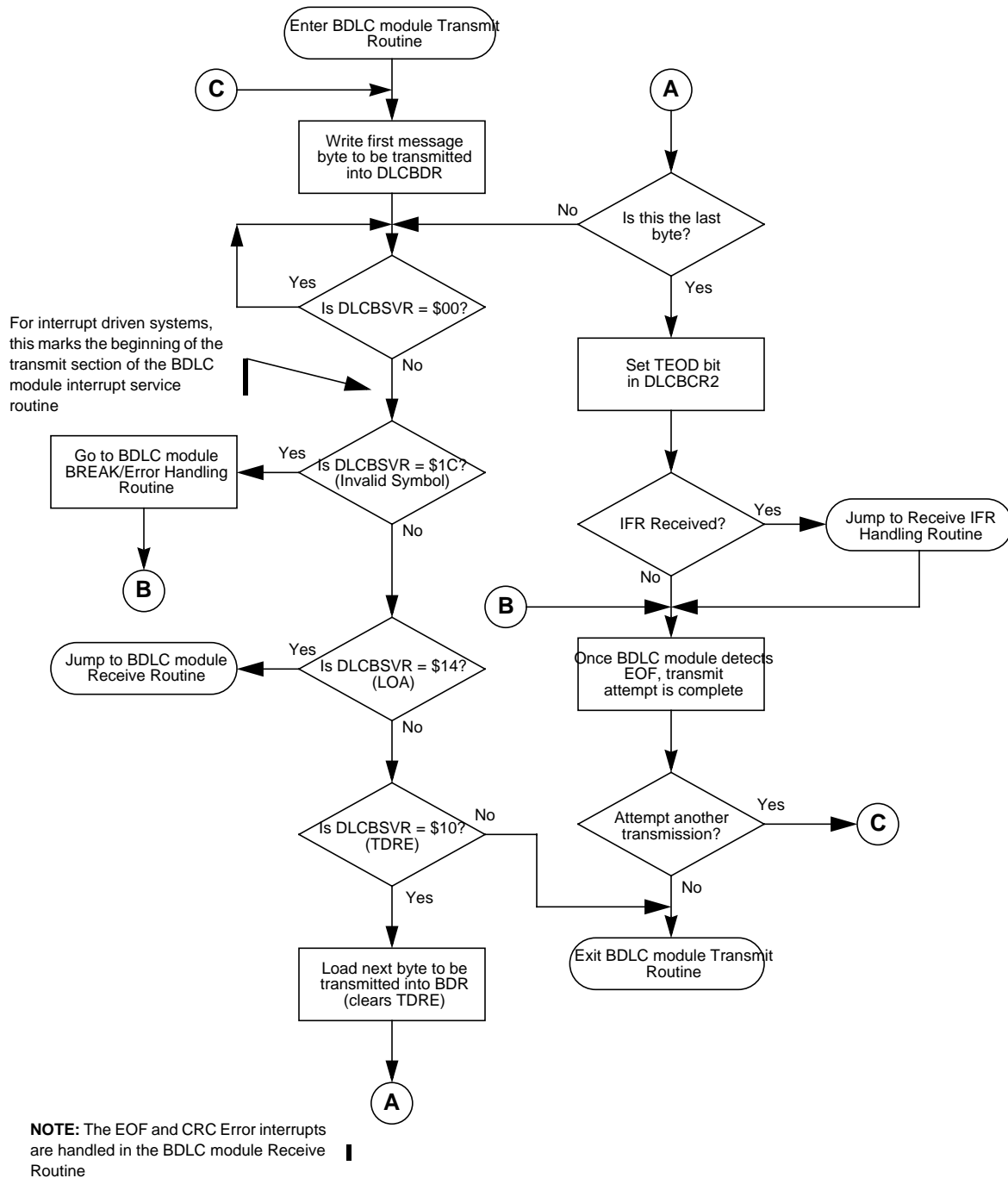


Figure 4-16 Basic BDLC Module Transmit Flowchart

## 4.9 BDLC Module Initialization

This section includes sample flows for initializing the BDLC module and using it to transmit and receive messages.

## 4.9.1 Initialization Sequence

To initialize the BDLC module, the user should first write the desired data to the configuration bits. The BDLC module should then be taken out of digital and analog loopback mode and enabled. Exiting from loopback mode will entail change of state indications in the DLCBSVR which must be dealt with. Once this is complete, CPU interrupts can be enabled (if desired), and then the BDLC module is capable of SAE J1850 serial network communication. For an illustration of the sequence necessary for initializing the BDLC module, refer to Figure 4-17 Basic BDLC Module Initialization Flowchart on page 85.

## 4.9.2 Initializing the Configuration Bits

The first step necessary for initializing the BDLC module following an MCU reset is to write the desired values to each of the BDLC module control registers. This is best done by storing predetermined initialization values directly into these registers. The following description outlines a basic flow for initializing the BDLC module. This basic flow does not detail more elaborate initialization routines, such as performing digital and analog loopback tests before enabling the BDLC module for SAE J1850 communication. However, from the following descriptions and the BDLC module specification, the user should be able to develop routines for performing various diagnostic procedures such as loopback tests.

- Step 1 - Initialize DLCBARD

Begin initialization of the configuration bits by writing the desired analog transceiver configuration data into the DLCBARD register. Following this write to DLCBARD, all of these bits will become read only.

- Step 2- Initialize DLCBRSR

The next step in BDLC module initialization is to write the desired bus clock divisor minus one into the DLCBRSR register. The divisor should be chosen to generate a 1 MHz or 1.048576 MHz mux interface clock ( $f_{bdlc}$ ). Following this write to DLCBRSR, all of these bits will become read only.

- Step 3- Initialize DLCBCR2

The next step in BDLC module initialization should be writing the configuration bits into the DLCBCR2 register. This initialization description assumes that the BDLC module will be put into normal mode (not 4X mode), and that the BDLC module should not yet exit either digital or analog loopback mode. Therefore, this step should write SMRST and DLOOP as logic ones, RX4XE as a logic zero, write NBFS to the desired level, and write TEOD, TSIFR, TMIFR1 and TMIFR0 as logic zeros. These last four bits MUST be written as logic zeros in order to prevent undesired operation of the BDLC module.

- Step 4- Initialize DLCBCR1

The next step in BDLC module initialization is to write the configuration bits in DLCBCR1. The CLKS bit should be written to its desired values at this time, following which it will become read-only. The IE bit should be written as a logic zero at this time so BDLC module interrupts of the CPU will remain masked for the time being. The IMMSG bit should be written as a logic one to prevent any receive events from setting the DLCBSVR until a valid SOF (or BREAK) symbol has been received by the BDLC module.

### 4.9.3 Exiting Loopback Mode and Enabling the BDLC module

Once the configuration bits have been written to the desired values, the BDLC module should be taken out of loopback and connected to the SAE J1850 bus. This is done by clearing the DLOOP bit and then setting the BDLCE bit in the DLCSCR.

- Step 5- Perform Loopback Tests (optional)

Once the BDLC module is configured for desired operation, the user may wish to perform digital and/or analog loopback tests to determine the integrity of the link to the SAE J1850 network. This would involve leaving the DLOOP bit (DLCBCR2) set, setting the BDLCE bit, performing the desired loopback tests and finally exiting digital loopback mode by clearing DLOOP in the DLCBCR2.

- Step 6- Exit Loopback Mode and enable the BDLC module

If loopback mode tests are not to be performed the BDLC module can be removed from digital loopback mode by clearing the DLOOP bit. The BDLC module can then be enabled by setting the BDLCE bit in the DLCSCR.

Once DLOOP is cleared and BDLCE is set, the BDLC module is ready for SAE J1850 communication. However, to ensure that the BDLC module does not attempt to receive a message already in progress or to transmit a message while another device is transmitting, the BDLC module must first observe an EOF symbol on the bus before the receiver will be activated. To activate the transmitter, the BDLC module will need to observe an Inter-Frame Separator symbol.

### 4.9.4 Enabling BDLC Interrupts

The final step in readying the BDLC module for proper communication is to clear any pending interrupt sources and then, if desired, enable BDLC module interrupts of the CPU.

- Step 7- Clear Pending BDLC Interrupts

In order to ensure that the BDLC module does not immediately generate a CPU interrupt when interrupts are enabled, the user should read the DLCBSVR to determine if any BDLC module interrupt sources are pending before setting the IE bit in the BCR1. If the BSVR reads as a %00000000, no interrupts are pending and the user is free to enable BDLC interrupts, if desired.

If the DLCBSVR indicates that an interrupt is pending, the user should perform whatever actions are necessary to clear the interrupt source before enabling the interrupts. Whether any interrupts are pending will depend primarily upon how much time passes between the exit from loopback modes and enabling the BDLC module and the enabling of interrupts. It is a good practice to always clear any source of interrupts before enabling interrupts on any MCU subsystem.

If any interrupts are pending (DLCBSVR not %00000000), then each interrupt source should be dealt with accordingly. Once all of the interrupt sources have been dealt with, the DLCBSVR should read %00000000, and the user is then free to enable BDLC interrupts.

- Step 8- Enable BDLC Interrupts

The last step in initializing the BDLC module is to enable interrupts to the CPU, if so desired. This is done by simply setting the IE bit in the DLCBCR1. Following this, the BDLC module is ready for operating in interrupt mode. If the user chooses not to enable interrupts, the DLCBSVR must be polled periodically to ensure that state changes in the BDLC module are detected and dealt with appropriately.

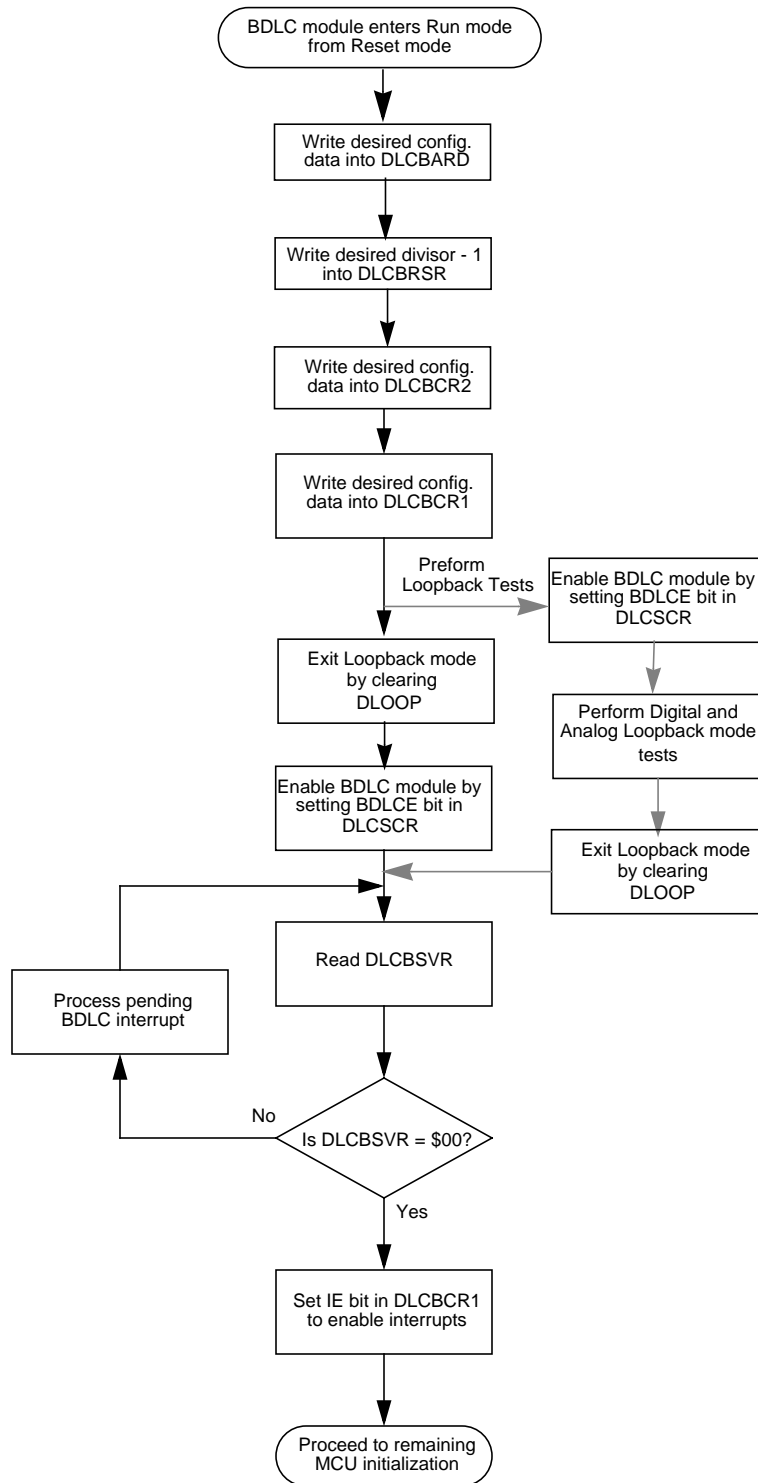


Figure 4-17 Basic BDLC Module Initialization Flowchart



## Section 5 Resets

### 5.1 General

The reset state of each individual bit is listed within **Section 3 Memory Map and Registers** which details the registers and their bit-fields.





## Section 6 Interrupts

### 6.1 General

Each change in status of the BDLC is encoded into the BDLC state vector register, (BSVR). Each state reflected in the BSVR can generate a CPU interrupt through the *ipi\_bdlc\_int* output, if the BDLC interrupts are enabled (IE = 1 in BCR1 Control register)

**Table 6-1** shows this interrupt information for *ipi\_bdlc\_int*.

**Table 6-1 Interrupt Summary**

Interrupt	Interrupt Source	Priority
ipi_bdlc_int	Refer to table below	determined at chip-level

Refer to **3.3.2** for a listing of the interrupt sources.



## Appendix A Electrical Specifications

N/A



## User Guide End Sheet

**FINAL PAGE OF  
94  
PAGES**