BYTE DATA LINK CONTROLLER
REFERENCE MANUAL

PRELIMINARY

TABLE OF CONTENTS

Paragraph **Title** **Page**

SECTION 1INTRODUCTION

1.1 BDLC Features 1-1
1.2 General 1-1
1.2.1 CPU Interface 1-3
1.2.2 Protocol Handler 1-3
1.2.3 MUX Interface 1-3

SECTION 2 BDLC OPERATING MODES

2.1 Power Off Mode 2-1
2.2 Reset Mode 2-1
2.3 Run Mode 2-2
2.4 BDLC Wait Mode 2-2
2.5 BDLC Stop Mode 2-2

SECTION 3 CPU INTERFACE ARCHITECTURE

3.1 BDLC Register Map 3-1
3.1.1 BDLC Control Register 1 (BCR1) 3-1
3.1.2 BDLC State Vector Register (BSVR) 3-1
3.1.3 BDLC Control Register 2 (BCR2) 3-2
3.1.4 BDLC Data Register (BDR) 3-2
3.1.5 BDLC Analog Roundtrip Delay Register (BARD) 3-2

SECTION 4INITIALIZATION

4.1 BDLC Configuration Bits 4-1
4.1.1 BDLC Control Register 1 (BCR1) 4-1
4.1.1.1 Ignore Message Bit (IMSG) 4-2
4.1.1.2 Clock Select Bit (CLKS) 4-2
4.1.1.3 Rate Select Field (RS[1:0]) 4-2
4.1.1.4 Interrupt Enable Bit (IE) 4-4
4.1.1.5 Wait Clock Mode Bit (WCM) 4-4
4.1.2 BDLC Control Register 2 (BCR2) 4-4
4.1.2.1 Analog Loopback Mode Bit (ALOOP) 4-4
4.1.2.2 Digital Loopback Mode Bit (DLOOP) 4-5
4.1.2.3 Receive 4X Mode Enable Bit (RX4XE) 4-5
4.1.2.4 Normalization Bit Format Select Bit (NBFS) 4-6
4.1.3 BDLC Analog Roundtrip Delay Register (BARD) 4-6
4.1.3.1 Analog Transceiver Enable Bit (ATE) 4-6
4.1.3.2 Receive Polarity Bit (RXPOL) 4-7

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

4.1.3.3 BDLC Analog Roundtrip Delay Offset Field (BO[3:0]) 4-7
4.2 BDLC Initialization 4-8
4.2.1 Configuration Bit Initialization 4-10
4.2.1.1 Step 1 - Initialize BARD 4-10
4.2.1.2 Step 2 - Initialize BCR2 4-10
4.2.1.3 Step 3 - Initialize BCR1 4-10
4.2.2 Exiting Loopback Mode 4-11
4.2.2.1 Step 4 - Perform Loopback Tests (Optional) 4-11
4.2.2.2 Step 5 - Exit Loopback Mode 4-11
4.2.3 Enabling BDLC Interrupts 4-11
4.2.3.1 Step 6 - Clear Pending BDLC Interrupts 4-11
4.2.3.2 Step 7 - Enable BDLC Interrupts 4-12

SECTION 5 BDLC STATE VECTOR REGISTER

5.1 BSVR Status Encoding 5-1
5.2 BSVR Interrupt Source Clearing Mechanisms 5-2
5.3 State Vector Handling Software 5-3
5.3.1 Example 1 (M68HC05/M68HC08) 5-3
5.3.2 Example 2 (M68HC11/M68HC12) 5-4
5.3.3 Example 3 (M68HC11/M68HC12) 5-5
5.4 Polling the BSVR 5-7

SECTION 6 PROTOCOL HANDLER ARCHITECTURE

6.1 Protocol State Machine 6-1
6.2 RX Shift Register 6-2
6.3 TX Shift Register 6-2
6.4 RX Shadow Register 6-2
6.5 TX Shadow Register 6-2

SECTION 7 MUX INTERFACE

7.1 Symbol Encoder/Decoder 7-1
7.2 RX Digital Filter Block 7-2
7.2.1 Operation 7-2
7.2.2 Performance 7-3
7.3 Digital Loopback Multiplexer 7-3

SECTION 8 TRANSMITTING A MESSAGE

8.1 BDLC Transmission Control Bits 8-1
8.2 BDLC Data Register (BDR) 8-1

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

8.3 Transmitting a Message with the BDLC 8-1
8.3.1 Step 1: Write the First Byte into the BDR 8-3
8.3.2 Step 2: When TDRE is Indicated, Write the Next Byte into the BDR 8-4
8.3.3 Step 3: Write the Last Byte to the BDR and Set TEOD 8-4
8.4 Transmitting Exceptions 8-5
8.4.1 Current SAE J1850 Bus Activity 8-5
8.4.2 Loss of Arbitration 8-6
8.4.3 Transmission Errors 8-6
8.4.3.1 Symbol Error 8-6
8.4.3.2 Framing Error 8-7
8.4.3.3 Bus Fault 8-7
8.4.3.4 BREAK 8-7
8.4.3.5 CRC Error 8-8
8.4.4 Transmitter Underrun 8-8
8.4.5 In-Frame Response to a Transmitted Message 8-8
8.5 Aborting a Transmission 8-8

SECTION 9 RECEIVING A MESSAGE

9.1 BDLC Reception Control Bits 9-1
9.2 Receiving a Message with the BDLC 9-1
9.2.1 Step 1: When RDRF Interrupt Occurs, Retrieve Data Byte 9-2
9.2.2 Step 2: When an EOF is Received, the Message is Complete 9-3
9.3 Filtering Received Messages 9-3
9.4 Receiving Exceptions 9-4
9.4.1 Reception Errors 9-4
9.4.1.1 Symbol Error 9-4
9.4.1.2 Framing Error 9-4
9.4.1.3 Bus Fault 9-4
9.4.1.4 BREAK 9-5
9.4.2 Receiver Overrun 9-5
9.5 In-Frame Response to a Received Message 9-5

SECTION 10 TRANSMITTING AN IN-FRAME RESPONSE

10.1 IFR Types Supported by the BDLC 10-1
10.1.1 IFR Type 0 – No Response 10-2
10.1.2 IFR Type 1 – Single Byte from a Single Responder 10-2
10.1.3 IFR Type 2 – Single Byte from Multiple Responders 10-2
10.1.4 IFR Type 3 – Multiple Bytes from a Single Responder 10-2
10.2 BDLC IFR Transmit Control Bits 10-3
10.2.1 Transmit Single Byte IFR Bit (TSIFR) 10-3

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

10.2.2 Transmit Multi-Byte IFR 1 Bit (TMIFR1) 10-4
10.2.3 Transmit Multi-Byte IFR 0 Bit (TMIFR0) 10-5
10.3 Transmitting an IFR 10-5
10.3.1 Transmitting a Type 1 IFR 10-6
10.3.1.1 Step 1 – Load the IFR Byte into the BDR 10-7
10.3.1.2 Step 2 – Set the TSIFR and TEOD Bits 10-7
10.3.2 Transmitting a Type 2 IFR 10-7
10.3.2.1 Step 1 – Load the IFR Byte into the BDR 10-8
10.3.2.2 Step 2 – Set the TSIFR Bit 10-9
10.3.2.3 Step 3 – If Necessary, Set the TEOD Bit 10-9
10.3.3 Transmitting a Type 3 IFR 10-9
10.3.3.1 Step 1: Load the First IFR Byte into the BDR 10-11
10.3.3.2 Step 2: Set the TMIFR Bit 10-11
10.3.3.3 Step 3: When TDRE is Indicated, Write the Next IFR Byte into the BDR 10-11
10.3.3.4 Step 4: Write the Last IFR Byte into the BDR and Set TEOD 10-11
10.4 Transmitting IFR Exceptions 10-12

SECTION 11 RECEIVING AN IN-FRAME RESPONSE

11.1 Receiving an IFR with the BDLC 11-1
11.1.1 Step 1: When RXIFR Interrupt Occurs, Retrieve IFR Byte 11-2
11.1.2 Step 2: When an EOF is Received, the IFR (and Message) is Complete 11-3
11.2 Receiving IFR Exceptions 11-3

SECTION 12SPECIAL OPERATIONS

12.1 Transmitting or Receiving a Block Mode Message 12-1
12.2 Receiving a Message in 4X Mode 12-1
12.3 Wakeup on Network Activity 12-2
12.3.1 Wakeup from BDLC Stop with CPU in Stop 12-2
12.3.2 Wakeup from BDLC Stop with CPU in Wait 12-2
12.3.3 Wakeup from BDLC Wait with CPU in Wait 12-3
12.4 Digital Loopback Mode 12-3
12.5 Analog Loopback Mode 12-3

APPENDIX ASAE J1850 SYMBOL TIMINGS

APPENDIX BSAE J1850 PROTOCOL

B.1 J1850 Frame Format B-1

TABLE OF CONTENTS
(Continued)

Paragraph	Title	Page
B.1.1	J1850 VPW Symbols	B-3
B.1.1.1	Logic 0	B-3
B.1.1.2	Logic 1	B-4
B.1.1.3	Normalization Bit (NB)	B-4
B.1.1.4	Break Signal (BREAK)	B-5
B.1.1.5	Start of Frame Symbol (SOF)	B-5
B.1.1.6	End of Data Symbol (EOD)	B-5
B.1.1.7	End of Frame Symbol (EOF)	B-5
B.1.1.8	Inter-Frame Separation Symbol (IFS)	B-5
B.1.1.9	Idle Bus	B-5
B.1.2	J1850 VPW Valid/Invalid Bits and Symbols	B-5
B.1.2.1	Invalid Passive Bit	B-6
B.1.2.2	Valid Passive Logic 0	B-6
B.1.2.3	Valid Passive Logic 1	B-6
B.1.2.4	Valid EOD Symbol	B-6
B.1.2.5	Valid EOF and IFS Symbol	B-7
B.1.2.6	Idle Bus	B-7
B.1.2.7	Invalid Active Bit	B-8
B.1.2.8	Valid Active Logic 1	B-8
B.1.2.9	Valid Active Logic 0	B-8
B.1.2.10	Valid SOF Symbol	B-8
B.1.2.11	Valid BREAK Symbol	B-9
B.1.3	Message Arbitration	B-9

APPENDIX CREGISTER SUMMARY

C.1	BDLC Register Map	C-1
C.1.1	BDLC Control Register 1	C-1
C.1.2	BDLC State Vector Register	C-3
C.1.3	BDLC Control Register 2	C-4
C.1.4	BDLC Data Register	C-6
C.1.5	BDLC Analog Roundtrip Delay Register	C-6

LIST OF ILLUSTRATIONS

Figure	Title	Page
1-1	BDLC Block Diagram	1-2
2-1	BDLC Operating Modes	2-1
4-1	Basic BDLC Initialization Flowchart	4-9
6-1	BDLC Protocol Handler Block Diagram	6-1
7-1	BDLC MUX Interface Block Diagram	7-1
7-2	BDLC RX Digital Filter Block Diagram	7-2
8-1	Basic BDLC Transmit Flowchart	8-3
9-1	Basic BDLC Receive Flowchart	9-2
10-1	IFR Types	10-1
10-2	Transmitting a Type 1 IFR	10-6
10-3	Transmitting a Type 2 IFR	10-8
10-4	Transmitting a Type 3 IFR	10-10
11-1	Receiving an IFR with the BDLC	11-2
A-1	BDLC Variable Pulse Width Modulation (VPW) Symbol Timing	A-2
B-1	J1850 Bus Message Format (VPW)	B-1
B-2	J1850 VPW Symbols with Nominal Symbol Times	B-4
B-3	J1850 VPW Received Passive Symbol Times	B-6
B-4	J1850 VPW Received Passive EOF and IFS Symbol Times	B-7
B-5	J1850 VPW Received Active Symbol Times	B-8
B-6	J1850 VPW Received BREAK Symbol Times	B-9
B-7	J1850 VPW Bitwise Arbitrations	B-10

LIST OF TABLES

Table **Title** **Page**

3-1 BDLC Register Map 3-1
4-1 BDLC Rate Selection for Binary Frequencies (CLKS = 1) 4-3
4-2 BDLC Rate Selection for Integer Frequencies (CLKS = 0) 4-3
4-3 BO[3:0] Offset Values 4-7
5-1 BSVR Interrupt Sources 5-2
10-1 IFR Control Bit Priority Encoding 10-3
A-1 BDLC Transmitter VPW Symbol Timing A-1
A-2 BDLC Receiver VPW Symbol Timing A-1
C-1 BDLC Register Map C-1
C-2 BDLC Rate Selection for Integer Frequencies (CLKS = 0) C-3
C-3 BDLC Rate Selection for Binary Frequencies (CLKS = 1) C-3
C-4 BSVR Interrupt Sources C-4
C-5 BO[3:0] Offset Values C-7

LIST OF TABLES
(Continued)

Table	Title	Page

SECTION 1INTRODUCTION

The byte data link controller (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.

This document is intended as an aid for developing the software for using the BDLC module to perform SAE J1850 communication. As such, it provides an implementation-independant description of the operation of the BDLC. Because some implementations of the BDLC may provide enhanced capabilites, it is strongly recommended that you refer to the BDLC module specification included in the MCU specification of the device you are using for a detailed hardware description.

This guide is also not intended as a tutorial on the SAE J1850 protocol. This document assumes the reader is familiar with the requirements and operation of the SAE J1850 protocol, as specified in the document *SAE Standard J1850 Class B Data Communication Network Interface*. For more detailed information on the SAE J1850 protocol, refer to the applicable SAE documents.

1.1 BDLC Features

Features of the BDLC module include:

- SAE J1850 compatible
- 10.4 kbps variable pulse width (VPW) modulation format
- Digital noise filter
- Collision detection
- Hardware cyclical redundancy check (CRC) generation and checking
- Two power saving modes with automatic wake up on network activity
- Polling and CPU interrupts with vector lookup available
- Receive and transmit block mode supported
- Supports 4X receive mode (41.6 kbps)
- Digital and Analog loopback modes
- In-frame response (IFR) types 0, 1, 2, and 3 supported
- Dedicated register for symbol timing adjustments

1.2 General

The BDLC consists of three functional blocks:

- The CPU interface block
- The protocol handler block
- The MUX interface block

Figure 1-1 displays a block diagram of the BDLC.

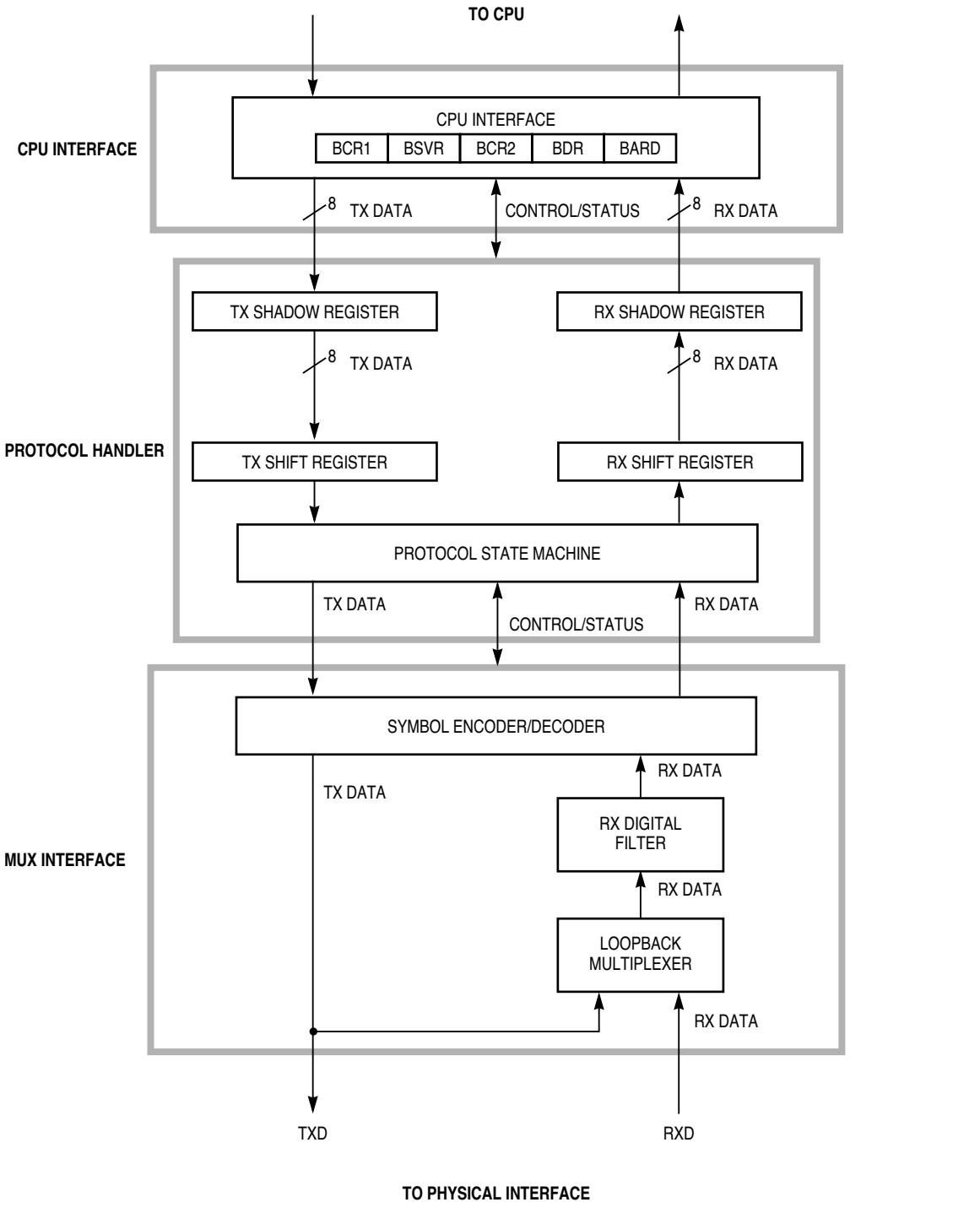

Figure 1-1 BDLC Block Diagram

1.2.1 CPU Interface

The CPU interface block handles the transfer of information between the CPU and the BDLC and consists of five user registers. For more information, refer to **SECTION 3 CPU INTERFACE ARCHITECTURE**.

1.2.2 Protocol Handler

The protocol handler block provides for receive and transmit byte buffering, message framing, arbitration, cyclic redundancy check (CRC) genertion/checking, and error detection. For more information, refer to SECTION 6 PROTOCOL HANDLER ARCHITECTURE.

1.2.3 MUX Interface

The MUX interface block provides for bit encoding/decoding and digital noise filtering between the protocol handler block and the physical interface. For more information, refer to SECTION 7 MUX INTERFACE.

SECTION 2 BDLC OPERATING MODES

The BDLC has five main modes of operation including power off mode, reset mode, run mode, BDLC wait mode, and BDLC stop mode. **Figure 2-1** shows the interactions of the various modes.

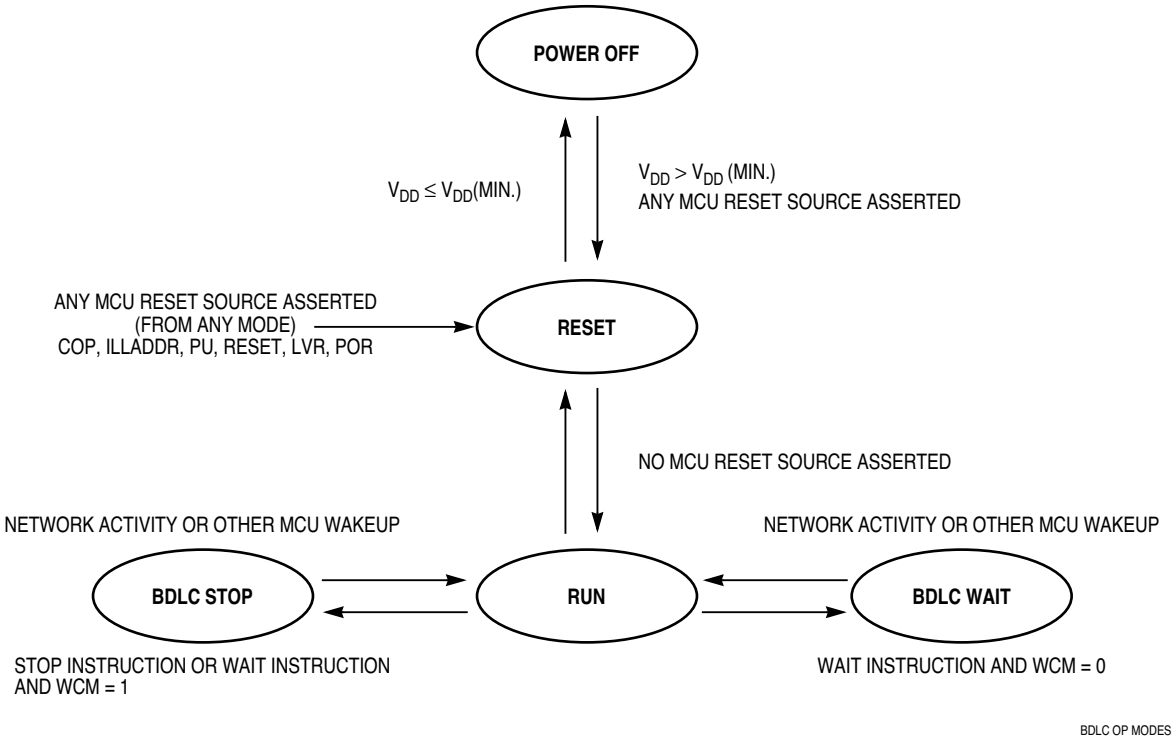

Figure 2-1 BDLC Operating Modes

2.1 Power Off Mode

This mode is entered from reset mode whenever the MCU supply voltage (V_{DD}) drops below its minimum specified value for the BDLC to guarantee operation. In this mode, the pin input and output specifications are not guaranteed.

2.2 Reset Mode

This mode is entered from the power off mode whenever V_{DD} rises above its minimum specified value and some MCU reset source is asserted. Reset mode is also entered from any other mode as soon as one of the MCU's possible reset sources (such as LVR, POR, COP watchdog, reset pin, etc.) is asserted.

The following conditions occur during reset mode:

- The internal BDLC voltage references are operative.
- V_{DD} is supplied to the internal circuits which are held in their reset state.
- The internal BDLC system clock runs.
- Registers assume their reset condition.
- Outputs are held in their programmed reset state, therefore inputs and network activity are ignored.

2.3 Run Mode

This mode is entered from the reset mode after all MCU reset sources are no longer asserted. Run mode is entered from the BDLC wait mode whenever activity is sensed on the J1850 bus.

Run mode is entered from the BDLC stop mode whenever network activity is sensed, although messages are not received properly until the clocks have stabilized and the CPU is also in run mode.

Normal network operation takes place in run mode. The user should ensure that all BDLC transmissions have ceased before exiting run mode.

2.4 BDLC Wait Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a WAIT instruction and if the WCM bit in the BCR1 register is cleared previously. In BDLC wait mode, the BDLC cannot transmit.

In this mode, the BDLC internal clocks continue to run. A subsequent successfully received byte, including one that is in progress at the time that this mode is entered, will cause the BDLC to wake up and generate a CPU interrupt request if the interrupt enable (IE) bit in the BDLC control register 1 (BCR1) is previously set. Additionally, if the BDLC receives a valid EOF symbol while in wait mode, the BDLC generates a CPU interrupt request which wakes up the BDLC and the CPU. Refer to 4.1.1.5 Wait Clock Mode Bit (WCM) for more information.

NOTE

To ensure correct operation of the BDLC, the user must wait until all network communication is complete before executing a WAIT instruction.

2.5 BDLC Stop Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a STOP instruction, or if the CPU executes a WAIT instruction and the WCM bit in the BCR1 register is set previously.

In this mode, the BDLC internal clocks are stopped. A passive-to-active transition on the J1850 bus causes the BDLC to wake up and generate a non-maskable CPU interrupt request. When a STOP instruction is used to put the BDLC in stop mode, the BDLC is not guaranteed to correctly receive the message which woke it up, since it may take some time for the BDLC internal operating clocks to restart and stabilize. If a WAIT instruction is used to put the BDLC in stop mode, the BDLC is guaranteed to correctly receive the byte which woke it up, if and only if an EOF symbol has been detected prior to issuing the WAIT instruction by the CPU. Otherwise, the BDLC does not correctly receive the byte that woke it up.

If this mode is entered while the BDLC is receiving a message, the first subsequent received edge causes the BDLC to wake up immediately, generate a CPU interrupt request, and wait for the BDLC internal operating clocks to restart and stabilize before normal communications can resume. Therefore, the BDLC is not guaranteed to receive that message correctly.

NOTE

To ensure correct operation of the BDLC, the user must wait until all network communication is complete, and an EOF symbol has been recognized by the BDLC, before executing a STOP or WAIT instruction.

SECTION 3 CPU INTERFACE ARCHITECTURE

The CPU interface provides the user interface between the CPU and the BDLC module. It is made up of five user registers: BCR1, BSVR, BCR2, BDR and BARD.

3.1 BDLC Register Map

Table 3-1 shows the BDLC register map. These registers are common to all implementations of the BDLC. Certain MCU implementations contain additional registers which may be used to enable the BDLC or modify its outputs. For information on these device-specific registers, refer to the appropriate device specification.

Table 3-1 BDLC Register Map

Address	7 0
$######	BDLC Control Register 1 (BCR1)
$######	BDLC State Vector Register (BSVR)
$######	BDLC Control Register 2 (BCR2)
$######	BDLC Data Register (BDR)
$######	BDLC Analog Roundtrip Delay Register (BARD)

NOTE

Depending upon the MCU implementation, the BDLC registers may not be implemented in this order, or may not even be contiguous in the MCU register map. Refer to the specific device register map for BDLC register placement.

3.1.1 BDLC Control Register 1 (BCR1)

BCR1 configures and controls the BDLC. Functions include selecting the BDLC module time-base, enabling interrupts, configuring the module clocks during wait mode, and indicating to the BDLC that a message being received should be ignored.

3.1.2 BDLC State Vector Register (BSVR)

BSVR indicates to the user the current status of the BDLC. Each state indicated in the BSVR can optionally generate a CPU interrupt, except for Wake-Up, which generates a non-maskable CPU interrupt.

3.1.3 BDLC Control Register 2 (BCR2)

BCR2 controls various transmitter operations of the BDLC. Functions include controlling the entry and exit into the loopback modes, configuring the IFR Normalization Bit, controlling entry into receive 4X mode, indicating to the BDLC when the last byte of a transmitted message or IFR has been written to the BDR, and controlling which type of IFR is to be transmitted.

3.1.4 BDLC Data Register (BDR)

The BDR register reads and writes data bytes to the BDLC. Bytes to be transmitted onto the SAE J1850 bus are written to the BDR, and bytes received from the SAE J1850 bus are read from the BDR.

3.1.5 BDLC Analog Roundtrip Delay Register (BARD)

The BARD register configures the BDLC for use with a variety of physical layer transceivers. Functions include selection of integrated or external transceiver, received data polarity, and physical layer round trip delay.

SECTION 4INITIALIZATION

Initialization of the BDLC module is typically performed in two stages. The first stage involves writing the desired values to the control bits which configure how the BDLC operates. These control bits include (but are not limited to) the clock select and rate select bits, the BDLC analog roundtrip delay bits, and the interrupt enable bit. The second stage of BDLC initialization removes the BDLC from the default loopback condition and (optionally) enables interrupts, thus enabling the BDLC for SAE J1850 communication. This involves not only writing values to certain control bits but also dealing with changes in the BDLC state vector register (BSVR).

To make these descriptions generic for any MCU implementation of the BDLC module, no register or bit locations are identified in relation to a particular device's memory/register map. For continuity, all registers which appear in any MCU implementation should retain the same register and bit designations.

4.1 BDLC Configuration Bits

The first stage of BDLC initialization involves writing the desired values to the control bits in the BDLC which configure the module for proper operation. These bit are referred to as the configuration bits, since most are generally written one time and are not dealt with again. In fact, many of these bits can only be written once following a reset of the MCU.

4.1.1 BDLC Control Register 1 (BCR1)

BCR1 contains five configuration bits, four of which (CLKS, RS[1:0], and WCM) can be written only once following a reset of the MCU. The interrupt enable (IE) bit, which can be written at any time, may be set just once or set and cleared repeatedly, depending upon the application. The IMSG bit is used actively by the SAE J1850 communication software. IMSG is not considered a configuration bit, but for convenience it is discussed in this section.

BCR1 — BDLC Control Register 1

7	6	5	4	3	2	1	0
IMSG	CLKS	RS[1:0]		0	0	IE	WCM
RESET							
1	1	1	0	0	0	0	0

4.1.1.1 Ignore Message Bit (IMSG)

The IMSG bit is used when the user has determined that a message being received is of no interest. The user sets the IMSG bit, and no further interrupts of the CPU occur until after the next SOF symbol is detected by the BDLC. A reception of an SOF symbol clears the IMSG bit, reenabling BDLC interrupts.

0 = Enable receiver. IMSG is cleared automatically by the reception of an SOF symbol or a BREAK symbol. It then generates interrupt requests and allows changes of the status register to occur. However, these interrupts may still be masked by the interrupt enable (IE) bit.
1 = Disable receiver. When IMSG is set, all BDLC interrupt requests are masked and the status bits are held in their reset state. If this bit is set while the BDLC is receiving a message, the rest of the incoming message is ignored.

NOTE

Since all receive and transmit status bits are held in their reset state when the IMSG bit is set, the user should never set the IMSG bit when transmitting a message. The user can also clear the IMSG bit, but this is not recommended.

4.1.1.2 Clock Select Bit (CLKS)

The CLKS bit allows selection between two base clock frequencies for BDLC operation: 1.049 MHz or 1.00 MHz. This gives the user the flexibility to choose between binary oscillator frequencies or integer oscillator frequencies, depending upon the application requirements, while still allowing proper SAE J1850 communication. This bit is used in conjunction with the rate select bits to prescale the MCU system clock frequency (f_{sys}) to the BDLC operating clock (f_{bdlc}) frequency.

0 = Integer frequency (1.00 MHz) selected for f_{bdlc}.
1 = Binary frequency (1.049 MHz) selected for f_{bdlc}.

4.1.1.3 Rate Select Field (RS[1:0])

RS[1:0] are used with the clock select bit to properly prescale f_{sys} to produce f_{bdlc}. While the clock select bit indicates to the prescaler circuitry whether a binary or integer base frequency is used, the rate select bits provide the necessary prescale to divide the system clock frequency down to the required BDLC operating frequency as defined in 4.1.1.2 Clock Select Bit (CLKS).

The f_{sys} used to produce the BDLC operating clock is either based on the oscillator frequency (f_{osc}) or on the oscillator frequency ÷ 2 ($f_{osc}/2$), depending upon the MCU family.

The BDLC operating clock is derived using the equation:

$$f_{bdlc} = f_{sys} \div \text{PRESCALER VALUE}$$

with f_{sys} derived as follows, depending upon the MCU family:

HC05: $f_{sys} = f_{osc} \div 2$

HC08: $f_{sys} = f_{osc}$

HC12: $f_{sys} = f_{osc} \div 2$

Refer to the particular device specification for more information on the system clock frequency used to arrive at the BDLC operating frequency.

Tables 4-1 and **4-2** give the appropriate RS[1:0] values for selecting the correct prescaler value necessary to achieve f_{bdlc}, depending upon f_{sys} as derived above.

NOTE

On MCUs which contain a PLL circuit to generate the internal operating frequency of the MCU, the frequency input into the BDLC prescaling circuit should not be derived from the PLL.

Table 4-1 BDLC Rate Selection for Binary Frequencies (CLKS = 1)

f_{sys}	R1	R0	Prescale Value	f_{bdlc}
1.049 MHz	0	0	1	1.049 MHz
2.097 MHz	0	1	2	1.049 MHz
4.194 MHz	1	0	4	1.049 MHz
8.389 MHz	1	1	8	1.049 MHz

Table 4-2 BDLC Rate Selection for Integer Frequencies (CLKS = 0)

f_{sys}	R1	R0	Prescale Value	f_{bdlc}
1.00 MHz	0	0	1	1.00 MHz
2.00 MHz	0	1	2	1.00 MHz
4.00 MHz	1	0	4	1.00 MHz
8.00 MHz	1	1	8	1.00 MHz

NOTE

Some of the system frequencies listed above are not achievable on all Motorola MCUs. Please refer to the MCU specification for a list of the achievable system frequencies.

4.1.1.4 Interrupt Enable Bit (IE)

The IE bit activates BDLC interrupts of the CPU. If this bit is set, all BDLC interrupt sources can cause a CPU interrupt. If this bit is clear, only a wakeup interrupt from wait (with WCM = 1) or stop mode can cause a CPU interrupt. All other interrupts are masked, and the user must poll the BSVR to determine the status of the BDLC. Interrupt requests are maintained until all of the interrupt request sources are cleared by performing the specified actions upon the BDLC's registers. Interrupts that are pending at the time that this bit is cleared may be lost.

0 = Disable interrupt requests from BDLC.
1 = Enable interrupt requests from BDLC.

4.1.1.5 Wait Clock Mode Bit (WCM)

The WCM bit determines the operation of the BDLC internal clocks during the CPU wait mode. If this bit is set, the BDLC internal clocks stop when a WAIT instruction is executed by the CPU. If this bit is cleared, the BDLC internal clocks continue to run when the CPU is in WAIT mode. For more information on the use of the WCM bit, refer to 2.4 BDLC Wait Mode.

0 = Run BDLC internal clocks during CPU wait mode.
1 = Stop BDLC internal clocks during CPU wait mode.

4.1.2 BDLC Control Register 2 (BCR2)

BCR2 contains only one configuration bit (NBFS). Three other control bits in BCR2 (ALOOP, DLOOP and RX4XE) are used at various times for putting the BDLC into special modes of operation. The four remaining bits (TEOD, TSIFR, TMIFR1, and TMFR0) are used during normal operation and are described in SECTION 8 TRANSMITTING A MESSAGE, SECTION 10 TRANSMITTING AN IN-FRAME RESPONSE, and SECTION 11 RECEIVING AN IN-FRAME RESPONSE.

BCR2 — BDLC Control Register 2

7	6	5	4	3	2	1	0
ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
RESET							
1	1	0	0	0	0	0	0

4.1.2.1 Analog Loopback Mode Bit (ALOOP)

The ALOOP bit indicates to the BDLC that the analog transceiver has been placed into an analog loopback mode, where its output driver is bypassed internally and the transmitted signal is looped back to the receive comparator. The operation of the BDLC is not significantly affected.

When this bit is set, the BDLC continues to operate normally. Once the ALOOP bit is cleared, the BDLC waits for the bus to be idle for at least one EOF time period before it begins to receive messages. The BDLC waits for the bus to be idle for at least one inter-frame separation time period before it begins to transmit messages. This ensures that the BDLC does not begin participating in network message traffic while a message transmission is in progress.

When ALOOP is cleared, once the bus has been idle for one EOF time, an EOF interrupt is generated (if interrupts are enabled), indicating to the user that the BDLC is ready to receive valid SAE J1850 messages.

0 = Indicates to the BDLC that the analog transceiver has exited loopback mode.
1 = Indicates to the BDLC that the analog transceiver is in loopback mode.

NOTE

To ensure the correct operation of the BDLC, the user must wait until all transmitter activity has been completed and an EOF symbol has been received before entering or exiting analog loopback mode.

4.1.2.2 Digital Loopback Mode Bit (DLOOP)

The DLOOP bit isolates the analog transceiver in order to determine whether a node's transceiver is causing a disruption on the bus. When the DLOOP bit is set, the analog transceiver is by-passed, with the transmitted digital signal being passed directly back to the digital receive circuitry. This results in the data placed into the BDR for transmission being transmitted directly back into the BDR to be received.

Once the DLOOP bit is cleared, the BDLC waits for the bus to be idle for at least one EOF time period before it begins to receive messages, and for the bus to be idle for at least one inter-frame separation time period before it begins to transmit messages.

When DLOOP is cleared and the bus has been idle for one EOF time, an EOF interrupt is generated, indicating to the user that the BDLC is ready to receive valid SAE J1850 messages. To prevent the handling of individual EOF interrupts following the clearing of ALOOP and DLOOP, the user can clear both bits simultaneously if both bits are set. This results in only a single EOF interrupt being generated by the BDLC.

0 = Indicates that the digital transmit output is connected directly to the transmit pin, and the receive pin is connected to the digital receive input.
1 = Indicates that the digital transmit output is connected internally to the digital receive input. The transmit and receive pins are not connected.

4.1.2.3 Receive 4X Mode Enable Bit (RX4XE)

The RX4XE bit allows the BDLC to receive SAE J1850 - VPW messages transmitted at four times the normal rate (41.6 kbps). This type of message is used primarily in a diagnostic or manufacturing environment, not during normal operation. The BDLC can only receive messages in 4X mode, it cannot transmit them.

If a BREAK symbol is received by the BDLC while it is in 4X mode (RX4XE set), the RX4XE bit is automatically cleared, the BSVR is updated to reflect the detection of the BREAK (“symbol invalid or out of range” error), and the BDLC returns to normal operating mode. If the RX4XE bit is not set and a 4X mode message is transmitted onto the SAE J1850 network, the BDLC interprets the message as noise, indicating to the host via the BSVR that an invalid symbol was detected on the bus.

0 = Indicates that the BDLC transmits and receives at 10.4 kbps.
1 = Indicates that the BDLC is in 4X receive-only operation.

4.1.2.4 Normalization Bit Format Select Bit (NBFS)

The NBFS bit determines the format of the normalization bit used by the BDLC while transmitting or receiving in-frame responses on the SAE J1850 network. The normalization bit precedes the first byte of an IFR, and is used to return the bus to an active state so the first bit of the IFR can be a passive bit, as defined in SAE J1850. The NBFS bit is the only bit in BCR2 which is typically written during the initialization of the BDLC, and not modified afterwards.

0 = NB that is received or transmitted is a logic 1 when the response part of an in-frame response (IFR) ends with a CRC byte.
1 = NB that is received or transmitted is a logic 0 when the response part of an IFR ends with a CRC byte.

4.1.3 BDLC Analog Roundtrip Delay Register (BARD)

The BARD register contains six configuration bits which are used to control BDLC interaction with the SAE J1850 analog transceiver. All of these bits can be written only once following a reset of the MCU, after which they become read-only bits.

BARD — BDLC Analog Roundtrip Delay Register

7	6	5	4	3	2	1	0
ATE	RXPOL	0	0	BO[3:0]			
RESET							
1	1	0	0	0	1	1	1

4.1.3.1 Analog Transceiver Enable Bit (ATE)

The ATE bit selects either an integrated or stand-alone SAE J1850 analog transceiver. For MCUs which contain an integrated analog transceiver, this transceiver is selected when the ATE bit is set, and the external analog transceiver is selected when this bit is clear. If no integrated transceiver is present, this bit has no effect on BDLC operation.

0 = Select off-chip analog transceiver.
1 = Select on-board analog transceiver.

4.1.3.2 Receive Polarity Bit (RXPOL)

The RXPOL bit selects the polarity of the digital receive data coming from an external SAE J1850 transceiver. If this bit is set, the receive data from the external transceiver is expected to be a true, non-inverted signal. If the RXPOL bit is cleared, the data coming from the analog transceiver is expected to be inverted from true polarity. When using an integrated analog transceiver, this bit has no affect on BDLC operation.

0 = Select inverted polarity, where an external transceiver inverts the receive signal from the SAE J1850 bus.
1 = Select normal/true polarity where the external transceiver does not invert the receive signal from the SAE J1850 bus.

4.1.3.3 BDLC Analog Roundtrip Delay Offset Field (BO[3:0])

BO[3:0] adjust the transmitted bit and symbol timings to account for the differing roundtrip delays found in different SAE J1850 analog transceivers. The allowable delay range is from 9 µs to 24 µs, with a nominal target of 16 µs (reset value). Refer to **Table 4-3** for the BO[3:0] values corresponding to the expected transceiver delays. Refer to the analog transceiver device specification for the expected roundtrip delay though both the transmitter and the receiver. The sum of these two delays makes up the total roundtrip delay value.

Table 4-3 BO[3:0] Offset Values

BO3	BO2	BO1	BO0	Delay
0	0	0	0	9 µs
0	0	0	1	10 µs
0	0	1	0	11 µs
0	0	1	1	12 µs
0	1	0	0	13 µs
0	1	0	1	14 µs
0	1	1	0	15 µs
0	1	1	1	16 µs
1	0	0	0	17 µs
1	0	0	1	18 µs
1	0	1	0	19 µs
1	0	1	1	20 µs
1	1	0	0	21 µs
1	1	0	1	22 µs
1	1	1	0	23 µs
1	1	1	1	24 µs

4.2 BDLC Initialization

To initialize the BDLC, write the desired data to the configuration bits, then take the BDLC out of digital and analog loopback mode. Exiting from loopback mode entails dealing with change of state indications in the BSVR. Once this is complete, CPU interrupts can be enabled (if desired), and then the BDLC module is capable of SAE J1850 serial network communication. **Figure 4-1** shows the sequence necessary for initializing the BDLC.

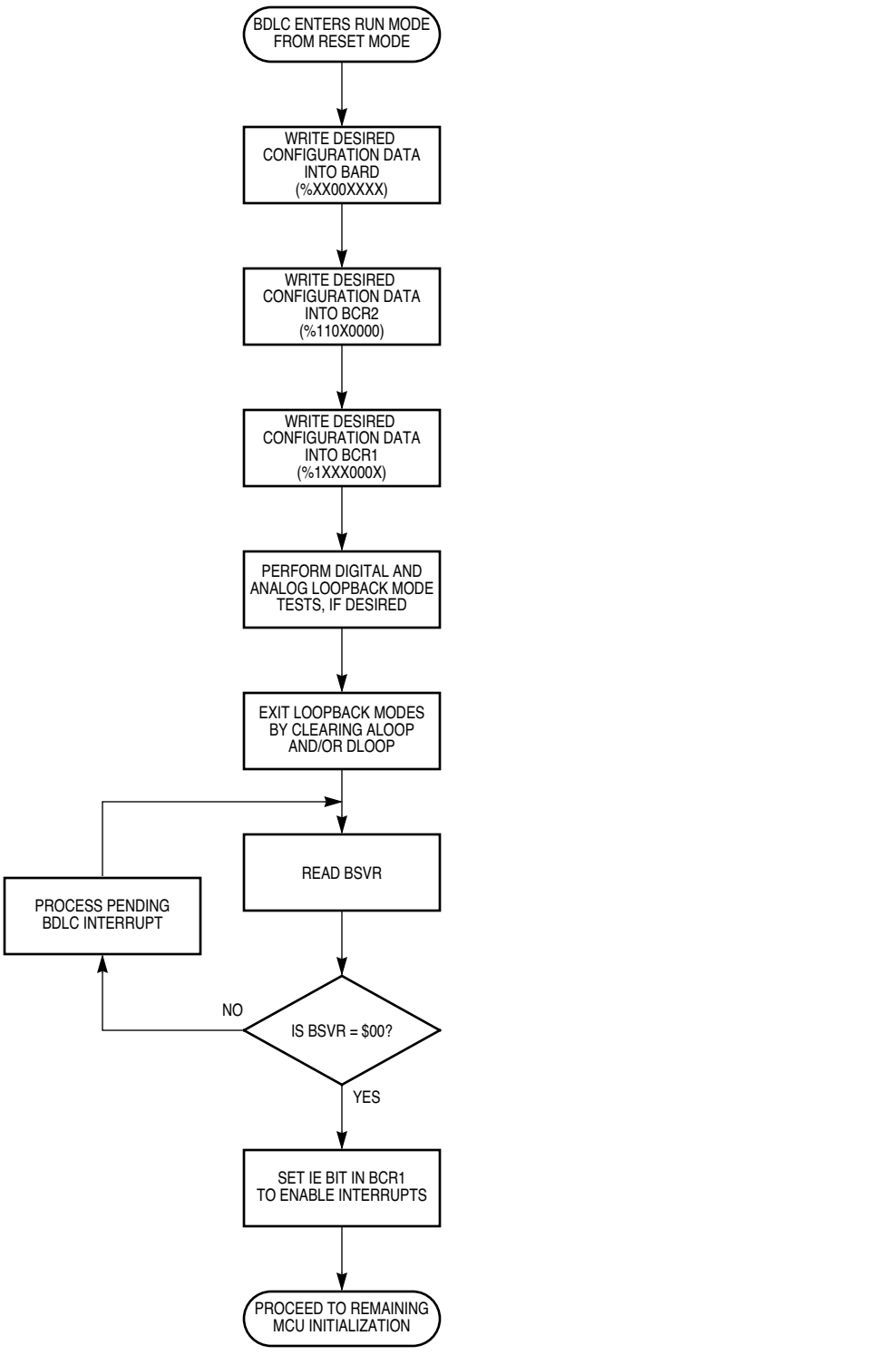

Figure 4-1 Basic BDLC Initialization Flowchart

4.2.1 Configuration Bit Initialization

The first step necessary for initializing the BDLC following an MCU reset is to write the desired values to each of the BDLC control registers. This is best done by storing predetermined initialization values directly into these registers. Since BCR1 and BCR2 each contain control bits which are not used for configuring the BDLC, care should be taken that these bits are written to their reset values to prevent undesired operation.

The following paragraphs outline a basic flow for initializing the BDLC. This basic flow does not detail more elaborate initialization routines, such as performing digital and analog loopback tests before enabling the BDLC for SAE J1850 communication. However, from the following descriptions and the BDLC specification, the user should be able to develop routines for performing various diagnostic procedures such as loopback tests.

4.2.1.1 Step 1 - Initialize BARD

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

4.2.1.2 Step 2 - Initialize BCR2

The next step in BDLC initialization should be writing the configuration bits into the BCR2 register. This initialization description assumes that the BDLC will be put into normal mode (not 4X mode), and that the BDLC should not yet exit either digital or analog loopback mode. Therefore, this step should write ALOOP 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.

NOTE

These last four bits must be written as logic zeros in order to prevent undesired operation of the BDLC.

4.2.1.3 Step 3 - Initialize BCR1

The next step in BDLC initialization is to write the configuration bits in BCR1. CLKS, RS[1:0], and WCM should be written to their desired values at this time. Afterwards, these bits are read-only. The IE bit should be written as a logic zero at this time so BDLC interrupts of the CPU remain masked temporarily. The IMSG bit should be written as a logic one to mask any receive interrupts until a valid SOF (or BREAK) symbol has been received by the BDLC.

NOTE

On some of the BDLC implementations, when using the MCU development system bits 2 and 3 of BCR1 must always be written as logic 1’s. This ensures that the BDLC operates normally in the emulator environment. Refer to the particular MCU emulator documentation for more information about operation of the BDLC in the emulation environment.

4.2.2 Exiting Loopback Mode

Once the configuration bits have been written to the desired values, the BDLC should be taken out of loopback mode and connected to the SAE J1850 bus. This is done by clearing the ALOOP and DLOOP bits, individually or at the same time.

4.2.2.1 Step 4 - Perform Loopback Tests (Optional)

Once the BDLC 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. Since this document is intended as a basic review on using the BDLC, development and implementation of loopback tests are left as an exercise for the user.

4.2.2.2 Step 5 - Exit Loopback Mode

Once loopback mode tests are complete (or skipped), the BDLC can be removed from analog loopback mode by clearing the ALOOP bit, and from digital loopback mode by clearing the DLOOP bit. These bits can be cleared individually if desired, or at the same time if both are set.

Once both ALOOP and DLOOP are cleared, the BDLC is ready for SAE J1850 communication. However, to ensure that the BDLC does not attempt to receive a message already in progress or to transmit a message while another device is transmitting, the BDLC must first observe an EOF symbol on the bus before the receiver is activated. To activate the transmitter, the BDLC needs to observe an inter-frame separator symbol.

4.2.3 Enabling BDLC Interrupts

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

4.2.3.1 Step 6 - Clear Pending BDLC Interrupts

In order to ensure that the BDLC does not immediately generate a CPU interrupt when interrupts are enabled, the user should read the BSVR to determine if any BDLC interrupt sources are pending before setting the IE bit in the BDLC. If the BSVR reads as a $00, no interrupts are pending and the user is free to enable BDLC interrupts, if desired.

If the BSVR 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 depends primarily upon how much time passes between the exit from loopback mode and the enabling of interrupts. It is a good practice to always clear any pending interrupt sources before enabling interrupts on any MCU subsystem.

If any interrupts are pending (BSVR $\neq$ $00), then each interrupt source should be dealt with accordingly. Once all of the interrupt sources have been dealt with, the BSVR should read $00, and the user is then free to enable BDLC interrupts.

4.2.3.2 Step 7 - Enable BDLC Interrupts

The last step in initializing the BDLC module is to enable interrupts to the CPU. This is done by setting the IE bit in the BCR1. Following this, the BDLC module is ready for operating in interrupt mode.

If the user chooses not to enable interrupts, the BSVR must be polled periodically to ensure that state changes in the BDLC are detected and dealt with in a timely manner.

SECTION 5 BDLC STATE VECTOR REGISTER

Each change in status of the BDLC is encoded into the BDLC state vector register (BSVR). The states reflected in the BSVR not only indicate to the user the current status of the BDLC, but each state can also generate a CPU interrupt, if BDLC interrupts are enabled.

BSVR — BDLC State Vector Register

7	6	5	4	3	2	1	0
0	0	ISF[3:0]				0	0
RESET							
0	0	0	0	0	0	0	0

The following explanation assumes that the IE bit in BCR1 has been set, enabling CPU interrupts by the BDLC. For polled operation, the BSVR operates the same, but the user needs to periodically read it to determine when a change of status has occurred in the BDLC.

5.1 BSVR Status Encoding

Unlike traditional encoding of status flags, where each flag has its own bit location in a status register, each BDLC status change which can (optionally) generate a CPU interrupt is encoded in the BSVR on a priority basis. When the BSVR is accessed by the user, the bit combination read reflects only the highest priority status change, rather than indicating every change in status which has occurred. This technique allows up to 256 status indicators to be encoded in a single eight bit register. However, in the BDLC there are only nine distinct states, so only four bit locations are utilized in the BSVR.

This technique of encoding status changes in the BSVR provides the user with a prioritized interrupt state which can be used as a jump index once the BDLC interrupt service routine is entered. By jumping directly to the appropriate location in the interrupt service routine, the amount of time needed for servicing BDLC interrupts can be greatly reduced.

Because of this state encoding, the user only deals with one BDLC interrupt source at a time. Once the highest priority BDLC 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.

In order to ease the CPU burden required to decode the BSVR value, the bits indicating the interrupt source are placed in the BSVR beginning in bit location 2. This placement results in the value of each interrupt source, when the entire byte is read, being a multiple of four. The value read from the BSVR can then be used to index into a jump table placed at the beginning of the interrupt service routine. This jump table enables the user to quickly move to the location in the interrupt service routine where that particular interrupt source is handled. **Table 5-1** shows the status encoding and priority assignments for all BDLC interrupt sources.

NOTE

Interrupt source 0 (no interrupts pending) does not generate an interrupt of the CPU, so an interrupt-based routine does not perform any tasks based on this state.

Table 5-1 BSVR Interrupt Sources

BSVR	ISF3	ISF2	ISF1	ISF0	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 (RXIFR)	2
$0C	0	0	1	1	Rx data register full (RDRF)	3
$10	0	1	0	0	Tx data register empty (TDRE)	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)

5.2 BSVR Interrupt Source Clearing Mechanisms

The mechanism for clearing each interrupt caused by the different states in the BSVR depends upon the source of the interrupt. For all states except for the received IFR byte (RXIFR), RX data register full (RDRF), and TX data register empty (TDRE), a read of the BSVR when that state is indicated clears the interrupt and the interrupt state in the BSVR.

A TDRE interrupt can only be cleared by a read of the BSVR when a TDRE state is indicated, followed by a write to the BDR. Likewise, RDRF and RXIFR interrupts can only be cleared by a read of the BSVR with the particular state indicated, followed by a read of the BDR. If the TDRE and RDRF clearing mechanisms seem familiar, that is because they operate very much like the serial communication interface (SCI) which is available on many Motorola MCUs.

As mentioned previously, when an interrupt state is cleared in the BSVR, if a lower priority interrupt event has occurred, that interrupt state is reflected in the BSVR. Since the interrupt sources are reflected in the BSVR one by one, there is no chance that an interrupt of lower priority will be inadvertently cleared while a higher priority interrupt is being serviced.

In order to limit the amount of CPU overhead required for servicing interrupts, the user should ensure that all pending BDLC interrupts have been serviced before exiting the BDLC interrupt service routine. This prevents direct reentry into the BDLC interrupt service routine should there be another BDLC interrupt pending when the CPU returns from dealing with the previous interrupt.

5.3 State Vector Handling Software

The bits in the BSVR have been positioned to allow use of this data as something more than simply status bits to be decoded. If used properly, the BSVR data can allow the user to jump directly to the appropriate location in the interrupt service routine. This can be done through the use of a jump table which is accessed upon initial entry into the BDLC service routine. The BSVR data can be used to index into a pointer table, which then allows the user to jump to the appropriate location. This typically requires fewer CPU cycles to reach the desired software routine than traditional bit decoding.

The following are three examples of how the BSVR data can be used in this manner to aid in interrupt processing. Multiple examples are described here, since there are a variety of ways to accomplish this, and because different CPUs have different indexed addressing modes, requiring slightly different techniques.

NOTE

Example 1 is written for M68HC05 and M68HC08 MCUs, but can be easily modified to run on the M68HC12 MCU. The other two examples are written for the M68HC11 and M68HC12, but can also be modified for use with the M68HC08 MCU.

5.3.1 Example 1 (M68HC05/M68HC08)

This example uses the BSVR data as the index value for moving into a jump table. Once in the table, the user then jumps to the appropriate location to begin servicing the interrupt source. There is one entry in the jump table for each of the nine possible BDLC states (including state 0). Since each entry in this table requires three bytes (one byte for the JMP opcode, two bytes for the destination address), a NOP instruction follows each JMP instruction to place each entry in the table at a 4-byte boundary. This aligns the offset of each jump table entry with the corresponding BSVR state value.

In this example, the user simply loads the BSVR data into the index register and then performs an indexed jump into a BSVR jump table starting from the label JMPTBL, located at the start of the jump table. Since this jump table has one JMP instruction for each state in the BSVR, the user can jump directly to the appropriate place in the service routine to deal with that particular state.

For CPUs with 16-bit constant offset indexed addressing, this solution provides the user with an easy and efficient way of vectoring into the desired place in the interrupt service routine in a fixed number of CPU bus cycles, regardless of which state is the source of the interrupt. This technique can be used with any M68HC05, M68HC08, or M68HC12 MCU.

For added security while servicing BDLC interrupts, the user could replace each NOP instruction in the jump table with an SWI instruction. Then, if an error occurs while indexing into the jump table and one of the SWI instructions is encountered, a trap placed at the SWI interrupt vector could prevent the CPU from entering a run away state.

```
SERVICE:    LDX     BSVR        ;Load State Vector value
                                ;into index register
            JMP     JMPTBL,X    ;Enter service routine
             •                  ;(must end in RTI)
             •
             •
JMPTBL:     JMP     SERV0       ;Service BSVR state #0
            NOP
            JMP     SERV1       ;Service BSVR state #1
            NOP
            JMP     SERV2       ;Service BSVR state #2
            NOP
             •
             •
             •
            JMP     SERV8       ;Service BSVR state #8
            END
```

5.3.2 Example 2 (M68HC11/M68HC12)

This example calculates the index value for moving into the jump table by adding the starting address of the jump table to the value in the BSVR. As in Example 1, once in the jump table the user then jumps to the appropriate location to begin servicing the interrupt source. There is again one entry in the jump table for each of the possible BDLC states (including state 0), with NOP instructions used to align the jump table entries on 4-byte boundaries.

In this example, the user loads the memory location of the label JMPTAB (the starting location of the jump table) into the index register X and loads the BSVR value into accumulator B. These two values are then added together to obtain the desired index value for entering the jump table. The indexed jump into the jump table is then made from the top of the MCU memory map (index offset = 0).

For CPUs which lack 16-bit constant offset indexed addressing, this solution provides the user with an efficient alternative to the previous example without restricting the jump table to the first page of memory. This technique can be used with most Motorola MCU families.

```
SERVICE:    LDX     #JMPTAB     ;Load location of jump
                                ;table into index reg.X
            LDAB    BSVR        ;Load the BSVR into
                                ;accumulator B
            ABX                 ;Add BSVR value to
                                ;contents of index reg. X
            JMP     0,X         ;Enter service routine
             •                  (must end in RTI)
             •
             •
JMPTAB:     JMP     SERV0       ;Service BSVR state #0
            NOP
            JMP     SERV1       ;Service BSVR state #1
            NOP
             •
             •
             •
            JMP     SERV8       ;Service BSVR state #8
            END
```

5.3.3 Example 3 (M68HC11/M68HC12)

This technique uses an address table rather than a jump table to index to the appropriate location in the interrupt service routine. With this technique, the starting address of the desired section of the interrupt service routine is loaded into the index register and then an indexed jump is made to that address. The advantage of this technique is that, although it takes several more CPU bus cycles to reach the desired starting address, the address table requires only half as much memory as the previously described jump table. As with the jump table, there is one entry in the address table for each of the nine possible BDLC states (including state 0), but only a 2-byte entry is required for each.

With this technique, the user loads the memory location of the label ADDTAB (the starting memory location of the address table) into index register X and loads the BSVR value into accumulator B. The value in accumulator B is then shifted right by one bit. This is necessary because the address table entries are on 2-byte boundaries, unlike the jump table entries. The value in accumulator B is then added to the value in index register X. The result is an index value which corresponds to the address table location holding the address of the service routine for the state indicated in the BSVR.

At this point, the desired address from the address table is loaded into index register X, using an indexed load from the top of the MCU memory map (index offset = 0). Once this is done, index register X contains the desired memory location within the BDLC interrupt service routine. The user then makes an indexed jump to this address, again with an index offset = 0.

This solution provides the user with a slightly slower but more memory efficient technique than using a jump table. As with example 2, this technique can be used with most Motorola MCU families.

```
SERVICE:    LDX     #ADDTAB     ;Load location of address
                                ;table into index reg. X
            LDAB    BSVR        ;Load State Vector value
                                ;into accumulator B
            LSRB                ;Shift the contents of
                                ;accumulator B to right
            ABX                 ;Add contents of acc. B
                                ;to index register X
            LDX     0,X         ;Load service routine
                                ;address into index reg.X


            JMP     0,X         ;Enter service routine
             •                  (must end in RTI)
             •
             •
ADDTAB:     FCB     SERV0       ;Address for BSVR state
                                ;#0 service routine
            FCB     SERV1       ;Address for BSVR state
                                ;#1 service routine
            FCB     SERV2       ;Address for BSVR state
                                ;#2 service routine
             •
             •
             •
            FCB     SERV8       ;Address for BSVR state
                                ;#8 service routine
            END
```

These examples are intended to illustrate how the BSVR state information can be used to speed up entry into the appropriate portion of the BDLC interrupt service routine. They are by no means the only ways in which the BSVR data can be used to accomplish this.

How the BSVR data can best be used depends upon the capabilities of the CPU chosen. For information on the memory map placement of the BSVR, as well as the indexing modes of the CPU being used, refer to the device specification of the particular MCU in question.

5.4 Polling the BSVR

In some applications it may be desirable to poll the BSVR rather than use the available BDLC CPU interrupts. In these instances, the BSVR state indications and clearing mechanisms are identical to those encountered in interrupt mode.

However, the user should be careful when selecting the rate at which the BSVR is polled. If the BSVR is polled at a rate slower than the length of the shortest byte transmitted on the SAE J1850 network (64µs X 8 bits = 512µs nominal), the user risks missing a byte of a message being received. This can occur if the BSVR is polled immediately before a byte is completely received, and then immediately after the next byte is completely received.

Conversely, if the BSVR is polled at a rate which is faster than the length of the longest BDLC service routine, the user runs the risk of polling the BSVR and reading a BDLC state which is already being serviced. Doing this can potentially result in the servicing of the same BSVR state twice.

To ensure that the BSVR data acted upon accurately reflects the state of the BDLC, the user should select a BSVR polling rate which falls between the transmission time of the shortest byte which can be transmitted on the SAE J1850 network and the execution time of the longest BDLC service routine.

SECTION 6 PROTOCOL HANDLER ARCHITECTURE

The protocol handler contains the protocol state machine, the receive (RX) and transmit (TX) shift registers, and the RX and TX shadow registers. **Figure 6-1** illustrates the BDLC protocol handler block diagram.

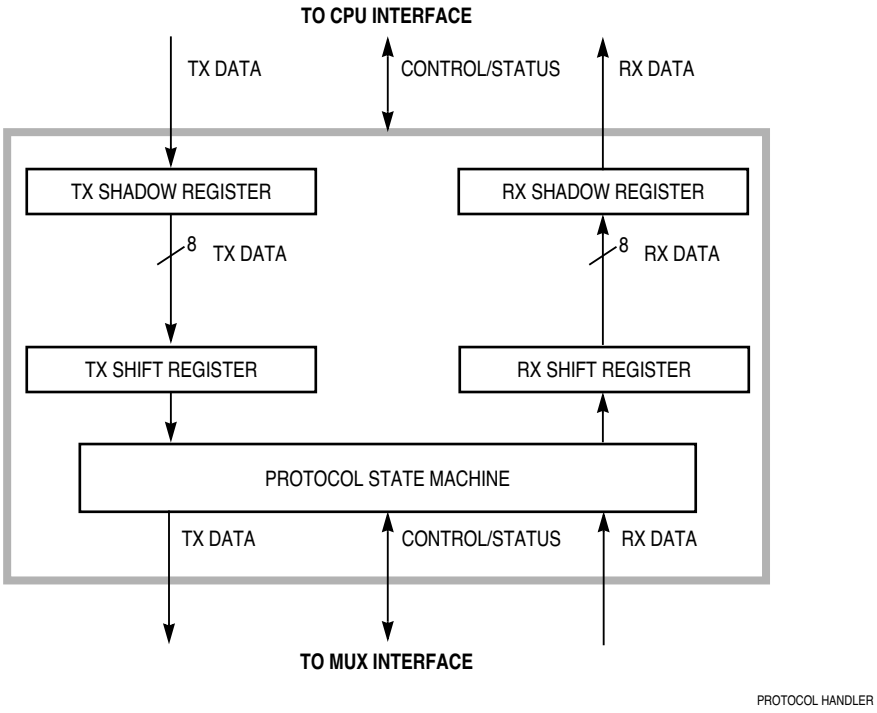

Figure 6-1 BDLC Protocol Handler Block Diagram

6.1 Protocol State Machine

All of the functions associated with performing the SAE J1850 protocol are executed or controlled by the protocol state machine. The state machine is responsible for bus access, message framing, collision detection, arbitration, CRC generation/checking, and error detection.

The protocol state machine also determines when the data in the TX shift register is to be shifted serially to the symbol encoder for conversion to VPW symbols to be transmitted, and controls the shifting of serial data from the symbol decoder into the RX shift register.

6.2 RX Shift Register

The RX shift register gathers received serial data bits from the SAE J1850 bus via the symbol decoder in the MUX interface, and makes them available in parallel form to the RX shadow register.

6.3 TX Shift Register

The TX shift register takes data (in parallel form) from the TX shadow register and presents it serially to the symbol encoder in the MUX interface so that it can be transmitted onto the SAE J1850 bus.

6.4 RX Shadow Register

The RX shadow register buffers bytes received from the SAE J1850 bus, which can then be read from the BDR. Once a complete byte is shifted into the RX shift register, it is transferred to the RX shadow register. A read of the BDR will always return the last byte written into the RX shadow register.

Immediately after the RX shift register has completed shifting in a byte of data, a parallel transfer of this data byte is made to the RX shadow register. Once this transfer occurs, the byte is available to be read by the user via the BDR.

6.5 TX Shadow Register

The TX shadow register is used to buffer bytes written to the BDR for transmission onto the SAE J1850 bus. Once a byte is written into the TX shadow register, it is then transferred to the TX shift register. Because of the double buffering of the BDR, bytes in the TX shadow register cannot be read by the user.

When the TX shift register has completed its shifting operation for the current byte, a parallel transfer of the data byte in the TX shadow register is made into the TX shift register. Once this transfer takes place, the TX shadow register is ready to accept new data from the user.

SECTION 7 MUX INTERFACE

The BDLC multiplexer (MUX) interface contains the symbol encoder/decoder, the RX digital filter and the digital loopback multiplexer. **Figure 7-1** illustrates the BDLC MUX interface block diagram.

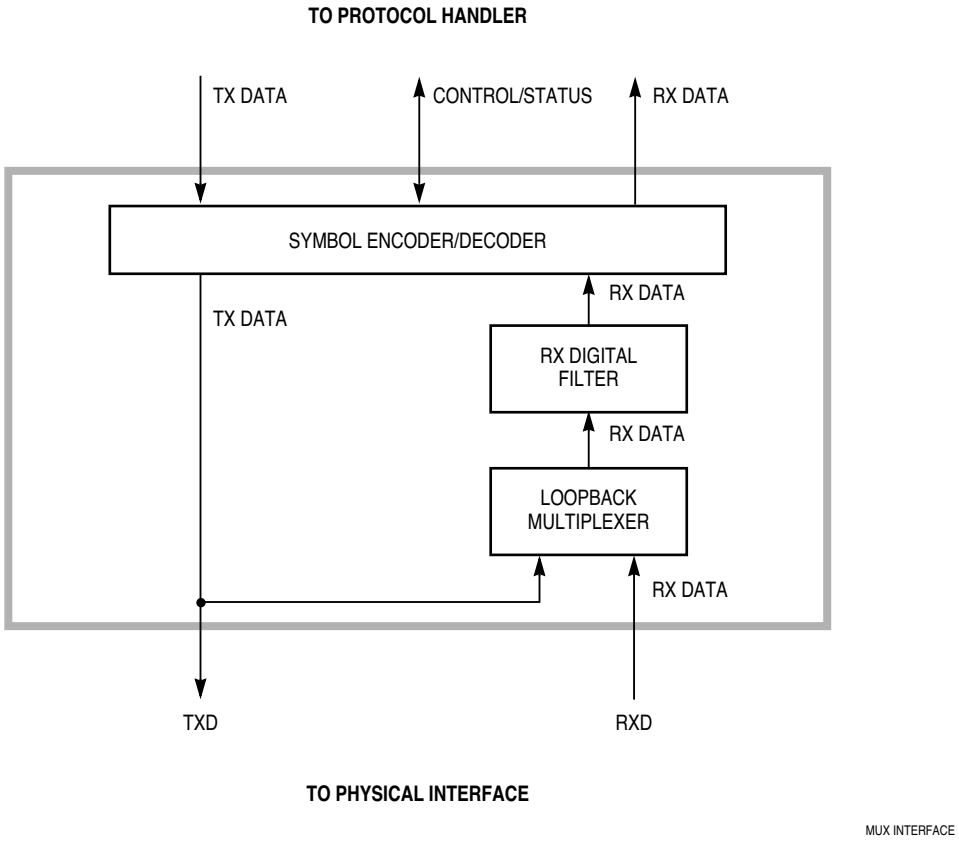

Figure 7-1 BDLC MUX Interface Block Diagram

7.1 Symbol Encoder/Decoder

The symbol encoder receives message framing information and serial transmit data from the BDLC protocol handler, and encodes this information into a logic-level VPW symbol and data bit stream. This stream is then transmitted to the physical interface, where the appropriate transmit drive levels and waveshaping are applied as it is transmitted onto the SAE J1850 bus. Likewise, the decoder receives a logic-level VPW data stream from the physical interface, and decodes the received symbols and data bits, which are then passed serially to the BDLC protocol handler.

7.2 RX Digital Filter Block

The receiver section of the BDLC MUX interface includes a digital low pass filter to remove narrow noise pulses which appear in the SAE J1850 bus. **Figure 7-2** illustrates the BDLC RX digital filter block diagram.

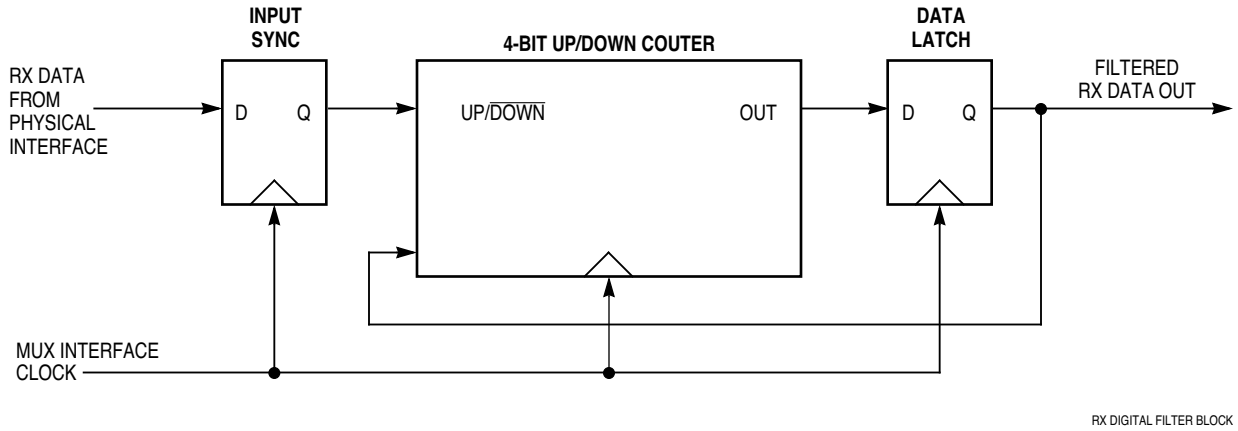

Figure 7-2 BDLC RX Digital Filter Block Diagram

7.2.1 Operation

The clock for the digital filter is provided by the BDLC operating clock (f_{bdlc}). At each positive edge of the clock signal, the current state of the physical interface receive signal is sampled. The receive signal state is used to determine whether the counter should increment or decrement at the next negative edge of the clock signal.

The counter increments if the input data sample is high, and decrements if the input sample is low. Therefore, the counter progresses either upward to 15 if, on average, the receive signal remains high or progresses downward to 0 if, on average, the receive signal remains low.

When the counter eventually reaches the value 15, the digital filter decides that the condition of the receive signal is at a stable logic level 1 and the data latch is set, causing the filtered RX data signal to become a logic level 1. 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 receive signal is at a stable logic level 0 and the data latch is reset, causing the filtered RX data signal to become a logic level 0. The counter is also prevented from underflowing, and can only be incremented from this state.

The data latch retains its value until the counter next reaches the opposite end point, signifying a definite transition of the signal.

7.2.2 Performance

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

If the receive signal transitions, there is a delay before that transition appears at the filtered RX data output signal. This delay is between 15 and 16 clock periods, depending on where the transition occurs with respect to the sampling points. This filter delay is taken into account when the BDLC performs message arbitration.

For example, if the frequency of the f_{bdlc} is 1.049 MHz, then the period (t_{bdlc}) is 954 ns, and the maximum filter delay in the absence of noise is 15.259 µs.

The effect of random noise on the receive signal depends on the characteristics of the noise itself. Narrow noise pulses on the receive signal are ignored completely 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 can 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, are detected by the next stage of the BDLC's symbol decoder as an invalid symbol.

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

7.3 Digital Loopback Multiplexer

The digital loopback multiplexer connects the BDLC digital transmit signal either to the transmit output, or directly to the digital receive input. If the digital transmit signal is connected directly to the digital receive input, the transmit output and receive input are isolated from the physical interface. Refer to 4.1.2 BDLC Control Register 2 (BCR2) for more information.

SECTION 8 TRANSMITTING A MESSAGE

This section describes the steps necessary for transmitting a message. Message reception is described in SECTION 9 RECEIVING A MESSAGE.

8.1 BDLC Transmission Control Bits

The transmit end of data (TEOD) control bit, located in BCR2, is used when transmitting a message onto the SAE J1850 bus. This bit is set by the user to indicate to the BDLC that the last byte of that part of the message frame has been loaded into the BDR. The TEOD bit is also used when transmitting an in-frame response (IFR). For more information, refer to SECTION 10 TRANSMITTING AN IN-FRAME RESPONSE.

Setting the TEOD bit indicates to the BDLC that the last byte written to the BDLC data register (BDR) is the final data byte to be transmitted, and that following this byte, a CRC byte and EOD symbol should be transmitted automatically. Setting the TEOD bit also inhibits any further TDRE interrupts until TEOD is cleared. The TEOD bit is cleared when the BDLC begins transmitting the first bit of the CRC byte, or if an error or loss of arbitration is detected on the bus. The user cannot clear the TEOD bit.

0 = The TEOD bit is cleared automatically when the first bit of the CRC byte is transmitted or if an error is detected. When TEOD is used to end an IFR transmission, TEOD is cleared when the BDLC begins to transmit the first bit of the CRC byte (if included) or the EOD symbol.
1 = Transmit end of data (EOD) symbol.

8.2 BDLC Data Register (BDR)

The BDR 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 BDR, and bytes received from the bus by the BDLC are read from the BDR. Since this register is double buffered, bytes written into it cannot be read by the user. If this is attempted, the byte which is read is the last byte placed in the BDR by the BDLC, not the last byte written to the BDR by the user.

BDR — BDLC Data Register

7	6	5	4	3	2	1	0
BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0
RESET							
U	U	U	U	U	U	U	U

8.3 Transmitting a Message with the BDLC

To transmit a message using the BDLC, the user writes the first byte of the message to be transmitted into the BDR, initiating the transmission process. When the TDRE

status appears in the BSVR, the user writes the next byte into the BDR. Once all of the bytes have been loaded into the BDR, the user sets the TEOD bit, and the BDLC completes the message transmission.

The following summarizes the basic steps required to transmit a message onto an SAE J1850 network using the BDLC. **Figure 8-1** displays a flowchart of the transmit sequence.

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.

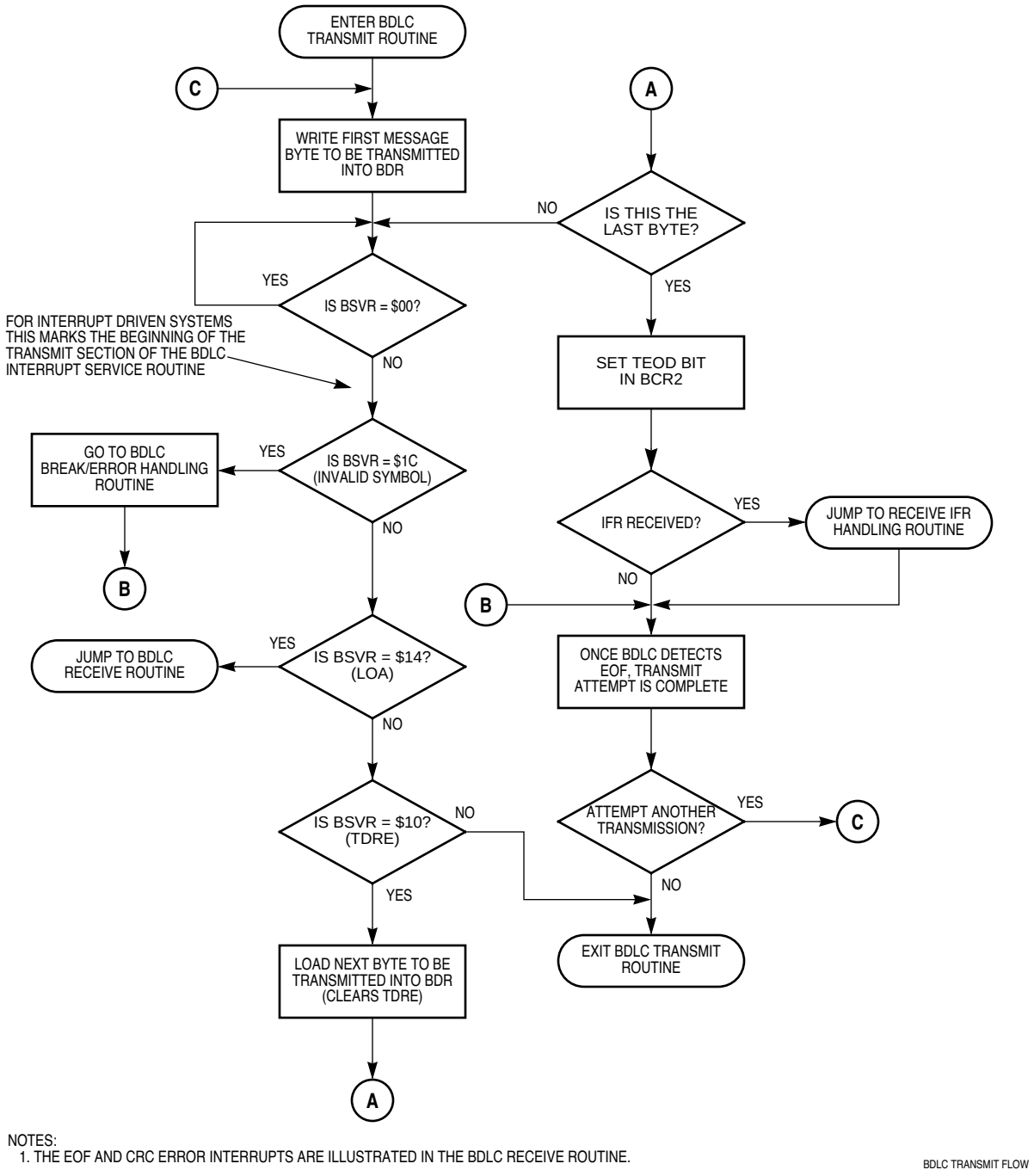

Figure 8-1 Basic BDLC Transmit Flowchart

8.3.1 Step 1: Write the First Byte into the BDR

To initiate a message transmission, the user simply loads the first byte of the message to be transmitted into the BDR. The BDLC performs the necessary bus acquisition duties to determine when the message transmission can begin.

Once the BDLC determines that the SAE J1850 bus is free, a start of frame (SOF) symbol is transmitted, followed by the byte written to the BDR. Once the BDLC begins transmitting this byte, the BSVR reflects that the next byte can be written to the BDR (TDRE interrupt).

NOTE

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

8.3.2 Step 2: When TDRE is Indicated, Write the Next Byte into the BDR

When a TDRE state is reflected in the BSVR, the user 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 BDR.

NOTE

Due to the design and operation of the BDLC, when transmitting a message, the user will write two of the bytes to be transmitted into the BDR 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.

8.3.3 Step 3: Write the Last Byte to the BDR and Set TEOD

Once the user has written the last byte to be transmitted into the BDR, the user then sets the TEOD bit in BCR2. When the TEOD bit is set, once the last byte written to the BDR is transmitted onto the bus, the BDLC begins transmitting the 8-bit CRC byte, as specified in SAE J1850. Following the CRC byte, the BDLC transmits 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 is recognized and the message is complete.

Setting the TEOD bit is the last step the user needs to take to complete the message transmission. No further transmission-related interrupts can occur. Receive interrupts continue until all bytes, including the CRC byte, are received. Once the message has been completely received by the BDLC, an EOF interrupt is 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 sequence in **Figure 8-1** shows the TEOD bit being set after the write to the BDR, either method is correct. If a TDRE interrupt is pending, it is cleared when the TEOD bit is set.

8.4 Transmitting Exceptions

While the above describes the basic transmit flow, the message transmit process can be interrupted under certain conditions. This can be caused by the SAE J1850 bus being busy, by a loss of arbitration to a higher priority message, or due to an error being detected on the network. For the transmit routine, any of these events, with the exception of the SAE J1850 bus being busy, can be dealt with in a similar manner.

8.4.1 Current SAE J1850 Bus Activity

Because there is no indication to the user that a message is being received from the SAE J1850 bus until the first byte is completely received and an RDRF interrupt occurs, it is possible for the user to write a byte to the BDR to initiate a transmission, but miss the timing window for that byte to participate in the arbitration process. If this happens, the BDLC does not immediately attempt to transmit the byte written to the BDR, but simply receives the message already being transmitted onto the SAE J1850 bus. The BDLC then waits until an idle bus condition is detected following the message being received, and then initiates the transmission.

The user can detect when this situation has occurred by monitoring the RDRF and TDRE interrupt sequence. Because the first TDRE interrupt for a message being transmitted precedes the first RDRF interrupt for that message, if an RDRF interrupt occurs first, and no loss of arbitration is detected, this indicates to the user that the transmission attempt was not initiated. When this is detected, the J1850 message handling software should immediately switch into the receive mode.

Once the message being received is completed, and an idle bus condition is detected, the BDLC initiates a transmission with the byte previously written to the BDR. The transmit sequence should resume, beginning from where it was interrupted.

If this situation occurs, and the user determines that an IFR must be transmitted in response to the message being received, the user simply writes the first byte of the IFR to the BDR, and sets the appropriate IFR control bit. The IFR byte overwrites the byte previously written to the BDR, and is transmitted at the appropriate time. The user must then reinitiate the original transmission from the beginning, since the byte originally written to the BDR has been discarded.

8.4.2 Loss of Arbitration

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

NOTE

If a loss of arbitration occurs on the 8th bit of a byte, the BDLC automatically appends up to two extra logic1 bits before halting transmission. This is done to ensure that a loss of arbitration to a transient on a byte boundary does not result in an abbreviated message being misinterpreted as a valid message.

These two extra bits are arbitrated normally and do not interfere with another message. The second logic 1 bit is not sent if the first loses arbitration. If the BDLC has lost arbitration to another valid message, the two extra logic 1's do not corrupt the current message. However, if the BDLC has lost arbitration due to noise on the bus, the two extra logic 1's ensure that the current message is detected and ignored as a noise-corrupted message.

When a loss of arbitration occurs, the BDLC message handling software should immediately switch into the receive mode. If the TEOD bit was set, it is 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.

8.4.3 Transmission Errors

Transmission errors include bit or symbol errors, message framing errors, CRC errors and bus faults. The transmission of a BREAK symbol, while not actually an error, can also be interpreted as an error by the BDLC. As with a loss of arbitration, if any error (except a CRC error) is detected on the SAE J1850 bus during a transmission, the BDLC stops transmitting immediately. Typically, a CRC error is not detected until the BDLC has completed transmitting all message bytes.

When any transmission error except a CRC error is detected, the byte which was being transmitted is discarded, and the "symbol invalid or out of range" status is reflected in the BSVR. As with a loss of arbitration, if the TEOD bit was set, it is cleared automatically, and any attempt to transmit the same message has to be reinitiated by the user. Unlike the other transmission errors, detection of a CRC error causes the "CRC error" status to be reflected in the BSVR.

8.4.3.1 Symbol Error

A symbol error occurs when an abnormal (invalid) bit or symbol is detected in a message being transmitted onto the SAE J1850 bus. This might include bits or symbols corrupted by transients appearing on the SAE J1850 bus during a transmission, or by greater than allowed ground offsets between modules on the network. Depending upon the nature of the symbol error, a CRC error may also be detected by the BDLC.

8.4.3.2 Framing Error

A framing error is detected if an EOD or EOF symbol is received on a non-byte boundary during a message transmission, or if an SOF or BREAK symbol is detected while a message is in progress. A framing error is also detected if the BDLC is transmitting an EOD symbol and instead receives an active symbol.

8.4.3.3 Bus Fault

A bus fault occurs when the SAE J1850 bus is shorted, either to ground or to a positive voltage source, typically vehicle battery. This can occur either though a mechanical failure of the network media or through the failure of a node on the network. In some instances, a transient appearing on the network might also appear to be a bus fault.

The response of the BDLC to the occurrence of a bus fault depends upon the type of bus fault. If the bus is shorted to a positive voltage source, following the initial error indication the BDLC waits for the bus to fall to a passive state before it attempts another message transmission, perceiving the bus to be active. As long as the short remains, the BDLC will never attempt to transmit a message onto the SAE J1850 bus.

NOTE

If the SAE J1850 bus is shorted to a positive voltage source while the BDLC is transmitting a message, the short may initially cause the BDLC to detect a loss of arbitration before the symbol or framing error is detected. How the bus fault is perceived is dependent upon at what point during the message the bus short occurs.

If the bus is shorted to ground, the BDLC may attempt to transmit a message and then detect an error because the short to ground prevents the bus from being driven to an active state. Unlike a bus short to a positive voltage source, a bus short to ground may be perceived by the BDLC as an idle bus condition. Therefore, the BDLC will always attempt a message transmission when a short to ground condition exists.

In either case, if the bus fault is temporary, once the fault is cleared the BDLC resumes normal operation. If the bus fault is permanent, it results in a loss of communication on the SAE J1850 bus.

8.4.3.4 BREAK

If a BREAK symbol is received while the BDLC is transmitting a message, it is perceived as an invalid symbol. Like a bus fault, it is also possible that the BREAK symbol may cause a loss of arbitration. This is again dependent upon the point in the message transmission at which the BREAK symbol appears.

8.4.3.5 CRC Error

A CRC error, detected following the reception of the EOD symbol, occurs when the output of the CRC shift register is not correct once all of the received bytes have been shifted through. Typically, a CRC error can only be detected after a transmission attempt in which another type of error was detected, unless a fault occurs in the BDLC CRC generating circuit.

8.4.4 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 BDR is completely transmitted onto the network before the next byte is loaded into the BDR, a transmitter underrun occurs.

If this happens, the BDLC transmits two additional logic ones to ensure that the partial message which was transmitted onto the bus does not end on a byte boundary. This is 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 is indicated in the BSVR. As with the other errors, it is up to the user’s software to determine if another transmission attempt should be made.

8.4.5 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 is indicated in the BSVR before the reception of the EOF symbol is indicated. SECTION 11 RECEIVING AN IN-FRAME RESPONSE describes how to handle the reception of IFR bytes.

8.5 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 BDR to initiate a message transmission and decides to send a different message, the byte in the BDR can be replaced, right up to the point that the message transmission begins.

If the user has loaded a byte into the BDR 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 causes 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 can also increase the probability of the transmitter losing arbitration if another node begins transmitting at the same time, also reducing the bus bandwidth needed.

SECTION 9 RECEIVING A MESSAGE

This section describes steps necessary for receiving a message from the SAE J1850 bus. Message transmission is described in SECTION 8 TRANSMITTING A MESSAGE.

When the first byte of a message comes in, the BSVR indicates to the CPU that a byte has been received. As each successive byte is received, that in turn is reflected in the BSVR. When the message is complete and the EOF has been detected on the bus, the BSVR reflects this, indicating that the message is complete.

For more information on receiving IFR bytes, refer to SECTION 11 RECEIVING AN IN-FRAME RESPONSE.

9.1 BDLC Reception Control Bits

The IMSG control bit enables message reception. Alternately, it can also prevent message reception. When the IMSG bit is set, BDLC interrupts of the CPU are inhibited until the next SOF symbol is received. This allows the BDLC to ignore the remainder of a message once the user 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 would require attention from the user for each byte broadcast on the network. The IMSG bit is cleared when the BDLC receives an SOF symbol, or it can also be cleared by the user, but this is not recommended. Refer to 4.1.1.1 Ignore Message Bit (IMSG) for more information.

NOTE

While the IMSG bit can be used to prevent the user from having to service the BDLC for every byte transmitted on the SAE J1850 bus, the IMSG bit should never be used to ignore the BDLC's own transmission. Because setting the IMSG bit causes all BSVR bits to be held in their reset state 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 user to complete the transmission correctly.

9.2 Receiving a Message with the BDLC

Receiving a message using the BDLC is extremely straight-forward. As each byte of a message is received and placed into the BDR, the BDLC indicates this to the user with an RX data register full (RDRF) status in the BSVR. EOF symbol reception, indicating to the user that the message is complete, is also reflected in the BSVR.

The following paragraphs describe the basic steps to be followed for receiving a message from the SAE J1850 bus with the BDLC. **Figure 9-1** displays a flowchart of a basic receive sequence.

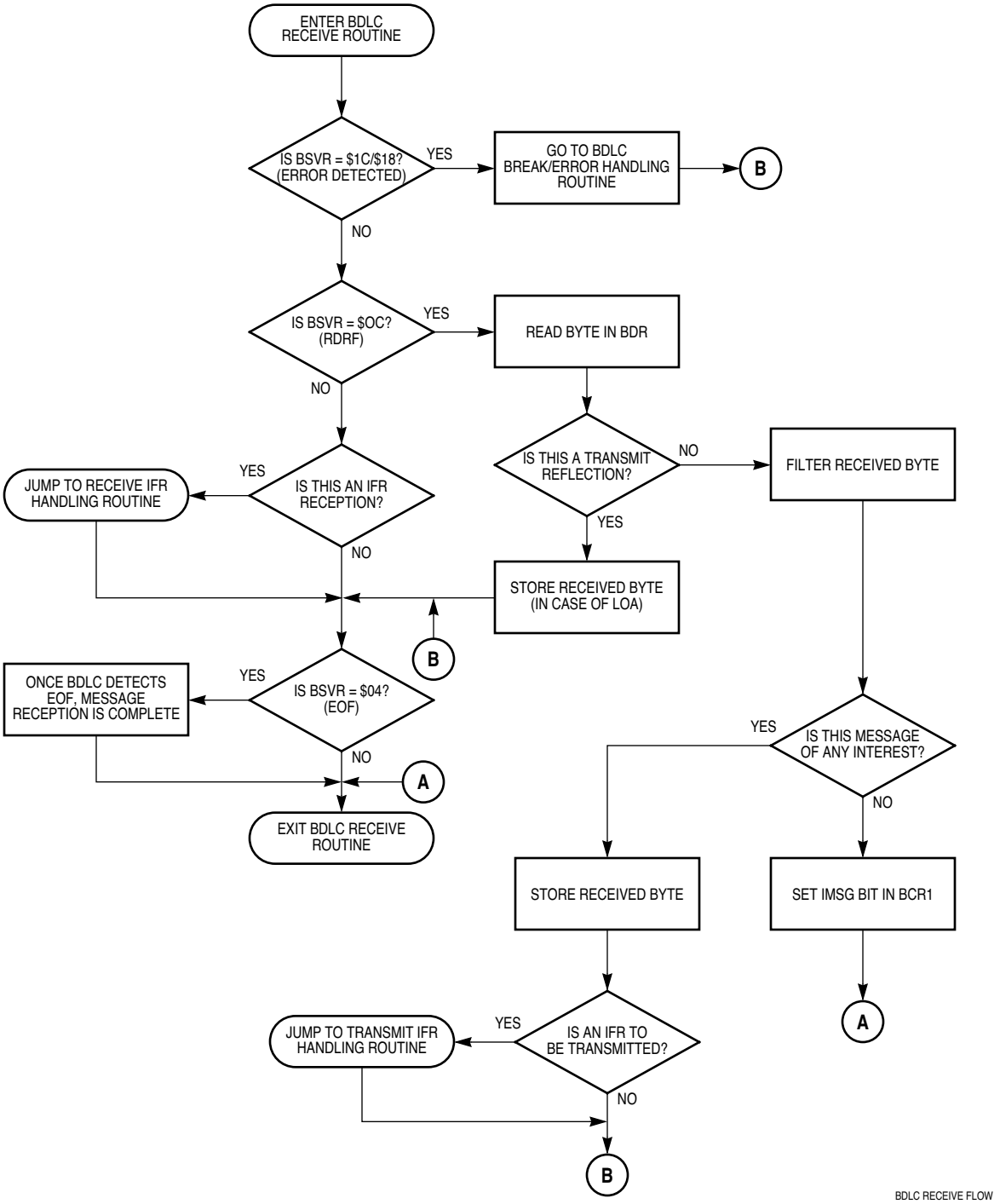

Figure 9-1 Basic BDLC Receive Flowchart

9.2.1 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 BDR, and an RDRF state is reflected in the BSVR. 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 be interpreted by the user as the beginning of a new message.

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

All bytes of the message, including the CRC byte, are placed into the BDR as they are received for the user to retrieve.

9.2.2 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 remains idle for a time period equal to an EOD symbol. Once the EOD symbol is received, the BDLC verifies that the CRC byte is correct. If the CRC byte is not correct, this error is reflected in the BSVR. There is no indication to the user that the EOD symbol has been observed.

If no in-frame response bytes are transmitted following the EOD symbol, the EOD transitions into an EOF symbol. When the EOF is received, it is reflected in the BSVR, indicating to the user that the message is complete. If IFR bytes do follow the first EOD symbol, once they are complete another EOD is transmitted, followed by an EOF.

Once the EOF state is reflected in the BSVR, 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.

9.3 Filtering Received Messages

No message filtering hardware is included on the BDLC, so all message filtering functions must be performed in software. Because the BDLC 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 user to decide while a message is still in progress whether or not that message is of any interest.

At any point during message reception, if the user determines that the message is of no interest, the IMSG bit can be set. Setting the IMSG bit commands the BDLC to ignore all further bus activity until the next valid SOF is received. This prevents the user from having to service the BDLC for each byte of every message sent over the network.

9.4 Receiving Exceptions

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

9.4.1 Reception Errors

Errors which can occur during a message reception, like transmission errors, include bit or symbol errors, message framing errors, CRC errors and bus faults. Likewise, the reception of a BREAK symbol, while not actually an error, is also interpreted as an error by the BDLC.

Detection of any of these errors (except for the CRC error) causes the BDLC to discard any partially received byte in the BDR, and wait until an EOF symbol is recognized on the SAE J1850 bus before any further messages are received. A CRC error is typically not detected until the BDLC has received all message bytes.

When any reception error except a CRC error is detected, the “symbol invalid or out of range” status is reflected in the BSVR. Detection of a CRC error causes the “CRC error” status to be reflected in the BSVR.

9.4.1.1 Symbol Error

A symbol error occurs when an abnormal (invalid) bit or symbol is detected in a message being received. This might include bits or symbols corrupted by transients appearing on the SAE J1850 bus, or by greater than allowed ground offsets between modules on the network. Depending upon the nature of the symbol error, a CRC error may also be detected by the BDLC.

9.4.1.2 Framing Error

A framing error is detected if an EOD or EOF symbol is received on a non-byte boundary, or if an SOF or BREAK symbol is detected while a message is in progress.

9.4.1.3 Bus Fault

A bus fault occurs when the SAE J1850 bus is shorted, either to ground or to a positive voltage source, typically a vehicle battery. This can occur either though a mechanical failure of the network media or through the failure of a node on the network. In some instances, a transient appearing on the network might also appear to be a bus fault.

The response of the BDLC to the occurrence of a bus fault depends upon the type of bus fault. If the bus is shorted to a positive voltage source, following the initial error indication the BDLC does not make any further indication that the bus short exists, since no additional messages appear on the SAE J1850 bus.

Likewise, if a short to ground occurs, no additional messages appear on the SAE J1850 bus as long as the bus short is present.

NOTE

As with the occurrence of a bus fault during a message transmission, if a bus fault occurs during a message reception, it may be perceived as an invalid symbol or framing error. In addition, a CRC error may also be indicated. How the bus fault is perceived is dependent upon at what point during the message the bus short occurs.

In either case, if the bus fault is temporary, once the fault is cleared the BDLC resumes normal operation. If the bus fault is permanent, it results in a loss of communication on the SAE J1850 bus.

9.4.1.4 BREAK

If a BREAK symbol is detected while the BDLC is receiving a message, it is perceived as an invalid symbol. This is again dependent upon the point in the message transmission at which the BREAK symbol appears.

9.4.2 Receiver Overrun

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

Once a receiver overrun occurs, there is no way for the user to recover the lost byte(s), so the entire message should be discarded. To prevent receiver overrun, the user should ensure that an RDRF interrupt is serviced before the next byte can be received. When polling the BSVR, the user should select a polling interval which can provide timely monitoring of the BDLC. For more information on selecting a polling interval, refer to 5.4 Polling the BSVR.

9.5 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) is indicated in the BSVR before reception of the EOF symbol is indicated. Refer to SECTION 11 RECEIVING AN IN-FRAME RESPONSE for a description of how to deal with the reception of IFR bytes.

SECTION 10 TRANSMITTING AN IN-FRAME RESPONSE

The BDLC 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 which follows assumes the user is familiar with the use of IFR 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.

10.1 IFR Types Supported by the BDLC

SAE J1850 defines four distinct types of IFR. **Figure 10-1** illustrates the different types of IFR. The first (and most basic) IFR is type 0, or no IFR. IFR types 1, 2 and 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 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. 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.

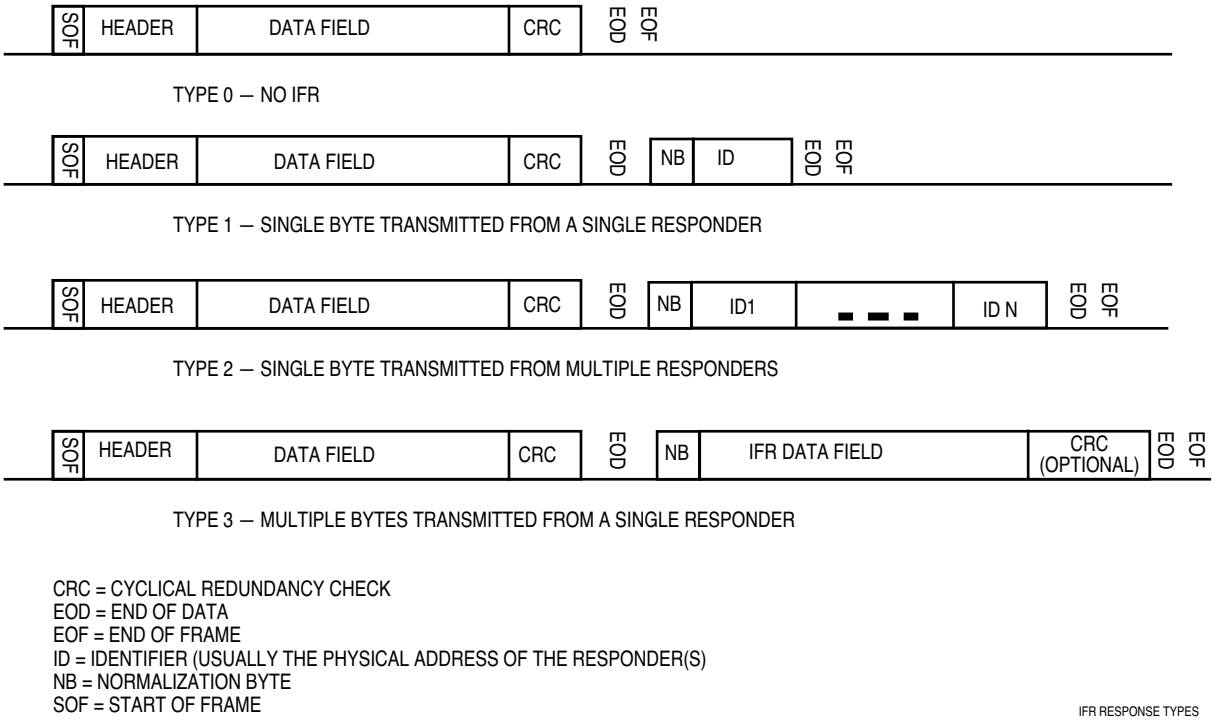

Figure 10-1 IFR Types

10.1.1 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 with no additional user intervention required.

10.1.2 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 begin transmitting their node ID as an IFR 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.

10.1.3 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.

The type 2 IFR 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 the type 2 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.

10.1.4 IFR Type 3 – Multiple Bytes from a Single Responder

The last type of IFR defined by SAE J1850 is the type 3 IFR. The type 3 IFR 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. The type 3 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.

10.2 BDLC IFR Transmit Control Bits

The BDLC has three bits which are used to control the transmission of an in-frame response. These bits, all located in BCR2, are TSIFR, TMIFR1, and TMIFR0. Each is used in conjunction with the TEOD bit to transmit one of the three IFR types defined in SAE J1850.

Because each of the bits used for transmitting an IFR with the BDLC is used to transmit a particular type of IFR, only one bit should be set by the user 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. **Table 10-1** indicates which IFR bit is actually acted upon by the BDLC should multiple IFR bits get set at the same time.

NOTE

As with transmitted messages, IFRs transmitted by the BDLC are also received by the BDLC. For a description of how IFR bytes received by the BDLC should be handled, refer to SECTION 11 RECEIVING AN IN-FRAME RESPONSE.

Table 10-1 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

10.2.1 Transmit Single Byte IFR Bit (TSIFR)

The TSIFR bit in BCR2 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 attempts to send that byte, preceded by the appropriate normalization bit, as a single byte IFR without a CRC. If arbitration is lost, the BDLC automatically attempts to transmit the byte again (without a normalization bit) as soon as the byte winning arbitration completes transmission. Attempts to transmit the byte 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 the bit remains cleared and no IFR transmit attempts are made. The TSIFR bit is cleared by a successful transmission of the byte as an IFR, when the TEOD bit is set, or if a symbol invalid error is detected. The TSIFR bit is also cleared and no IFR attempt is made if an error is detected before the first EOD is received.

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

0 = The TSIFR bit is cleared automatically once the BDLC has successfully transmitted the byte in the BDR onto the bus, or TEOD is set, or an error is detected on the bus.
1 = If the TSIFR bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received, the BDLC attempts to transmit the appropriate normalization bit followed by the byte in the BDR.

10.2.2 Transmit Multi-Byte IFR 1 Bit (TMIFR1)

The 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 BDR, the BDLC begins transmitting that byte, preceded by the appropriate normalization bit, onto the SAE J1850 bus. Once this happens, a TDRE interrupt occurs, indicating to the user that the next IFR byte should be loaded into the BDR. When the last byte to be transmitted is written to the BDR, the user sets the TEOD bit. This causes 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 remains cleared and no attempt is made to transmit the IFR byte. The TMIFR1 bit is 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 halts immediately and the loss of arbitration is indicated in the BSVR.

0 = The TMIFR1 bit is cleared automatically once the BDLC has successfully transmitted the CRC byte and EOD symbol, or by the detection of an error on the multiplex bus, or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.
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 attempts to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into the BDR register. After TEOD has been set and the last IFR byte has been transmitted, the CRC byte is transmitted.

10.2.3 Transmit Multi-Byte IFR 0 Bit (TMIFR0)

The 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 BDR, the BDLC begins transmitting that byte, preceded by the appropriate normalization bit, onto the SAE J1850 bus. Once this happens a TDRE interrupt occurs, indicating to the user that the next IFR byte should be loaded into the BDR. When the last byte to be transmitted is written to the BDR, the user sets the TEOD bit. This causes 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 remains cleared and no attempt is made to transmit the IFR byte. The TMIFR0 bit is 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 halts immediately and the loss of arbitration is indicated in the BSVR.

0 = The TMIFR0 bit is cleared automatically once the BDLC has successfully transmitted the EOD symbol, or by the detection of an error on the multiplex bus, or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.

1 = If the TMIFR0 bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received, the BDLC attempts to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte is written into the BDR register. After TEOD has been set, the last IFR byte to be transmitted will be the last byte which was written into the BDR register.

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 are transmitted to ensure that the IFR does not end on a byte boundary. This can cause an error in a type 1 IFR.

10.3 Transmitting an IFR

While the design of the BDLC makes the transmission of each type of IFR similar, the steps necessary for transmitting each are 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.

10.3.1 Transmitting a Type 1 IFR

To transmit a type 1 IFR, the user loads the byte to be transmitted into the BDR and sets both the TSIFR bit and the TEOD bit. This directs the BDLC to attempt transmitting the byte written to the BDR one time, preceded by the appropriate normalization bit. If the transmission is not successful, the byte is discarded and no further transmission attempts are made. **Figure 10-2** displays a flowchart of the steps involved in transmitting a type 1 IFR.

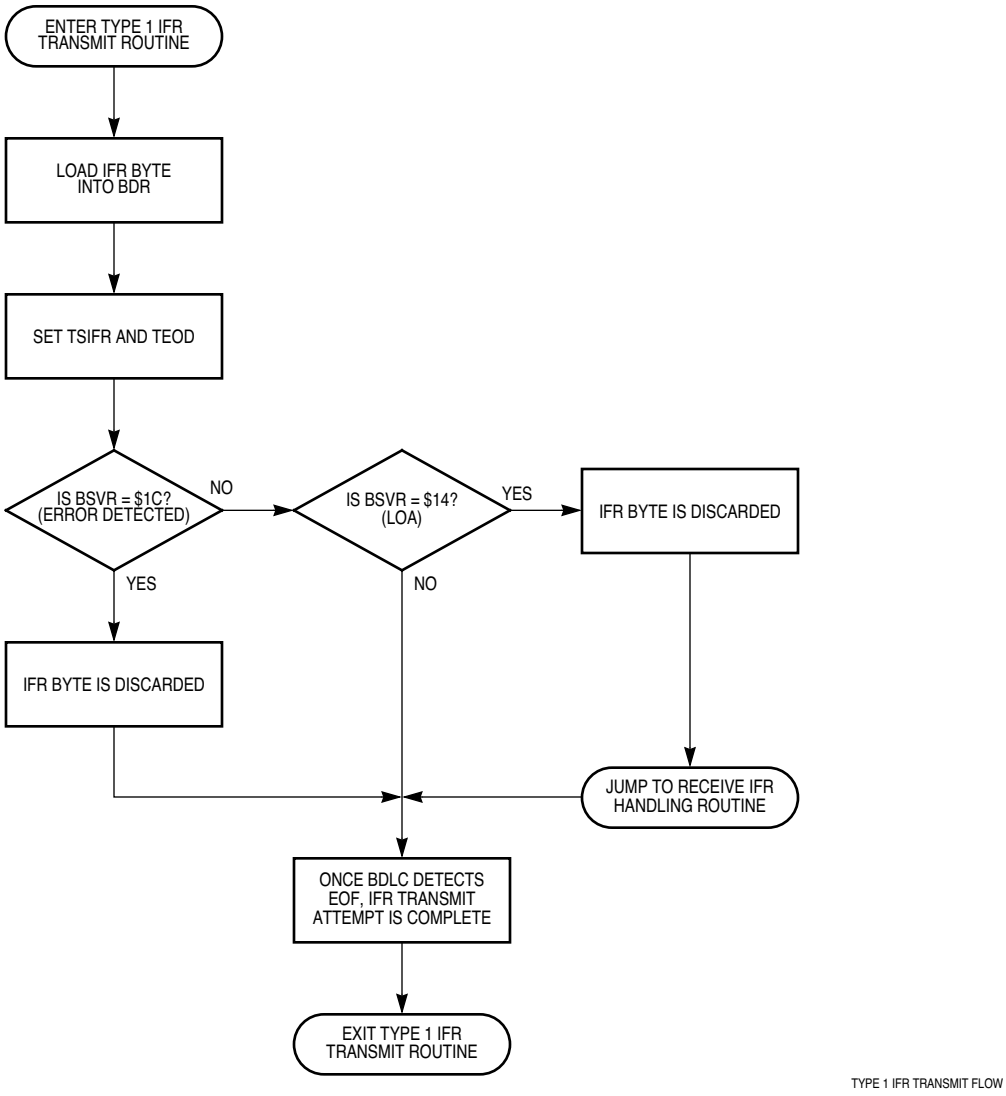

Figure 10-2 Transmitting a Type 1 IFR

10.3.1.1 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 BDR. If a byte has already been written into the BDR for transmission as a new message, the user can simply write the IFR byte to the BDR, replacing the previously written byte. This must be done before the EOD symbol following the CRC byte is received, indicating that the main part of the message is complete.

10.3.1.2 Step 2 – Set the TSIFR and TEOD Bits

The final step in transmitting a type 1 IFR with the BDLC is to set the TSIFR and TEOD bits in BCR2. Setting both bits directs the BDLC to make one attempt at transmitting the byte in the BDR as an IFR. If the byte is transmitted successfully, or if an error or loss of arbitration occurs, TEOD and TSIFR are cleared and no further transmit attempts are made.

10.3.2 Transmitting a Type 2 IFR

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

Figure 10-3 displays a flowchart of the steps involved in transmitting a type 2 IFR.

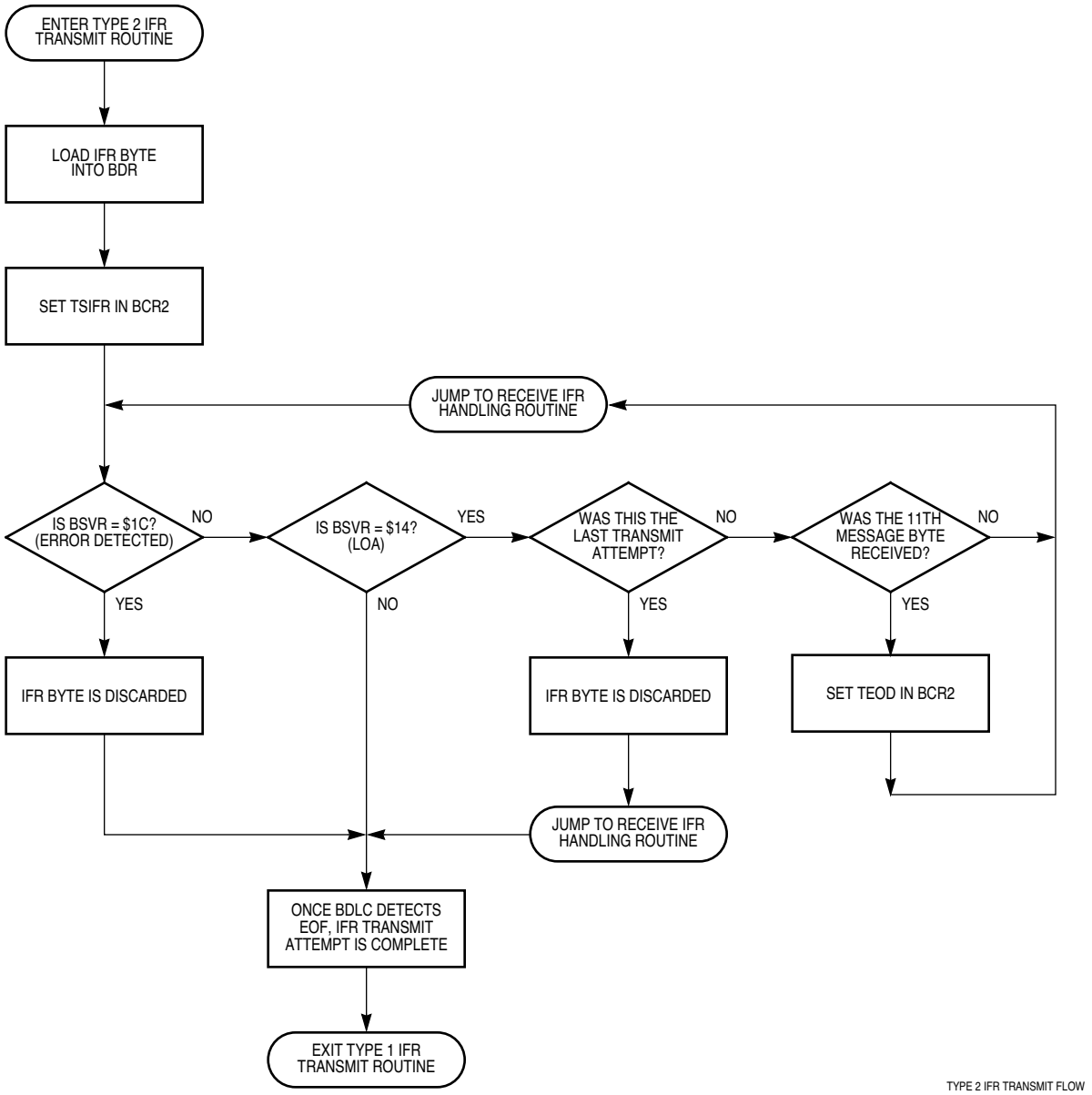

Figure 10-3 Transmitting a Type 2 IFR

10.3.2.1 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 BDR. If a byte has already been written into the BDR for transmission as a new message, the user can simply write the IFR byte to the BDR, replacing the previously written byte. This must be done before the EOD symbol following the CRC byte is received, indicating that the main part of the message is complete.

10.3.2.2 Step 2 – Set the TSIFR Bit

The second step necessary for transmitting a type 2 IFR is to set the TSIFR bit in BCR2. Setting this bit directs the BDLC to attempt to transmit the byte in the BDR as an IFR until it is successful. If the byte is transmitted successfully, or if an error or loss of arbitration occurs, TSIFR is cleared and no further transmit attempts are made.

10.3.2.3 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's attempt to transmit the byte loaded into the BDR 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 is between transmission attempts, it makes one more attempt to transmit the IFR byte. If it is transmitting the byte when TEOD is set, the BDLC continues the transmission attempt until it is successful or it loses arbitration to another transmitter. At this point, it discards the byte and makes 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 prevents the 12-byte limit from being exceeded.

10.3.3 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 BDR and 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 discusses both.

Figure 10-4 displays a flowchart of the steps involved in transmitting a type 3 IFR.

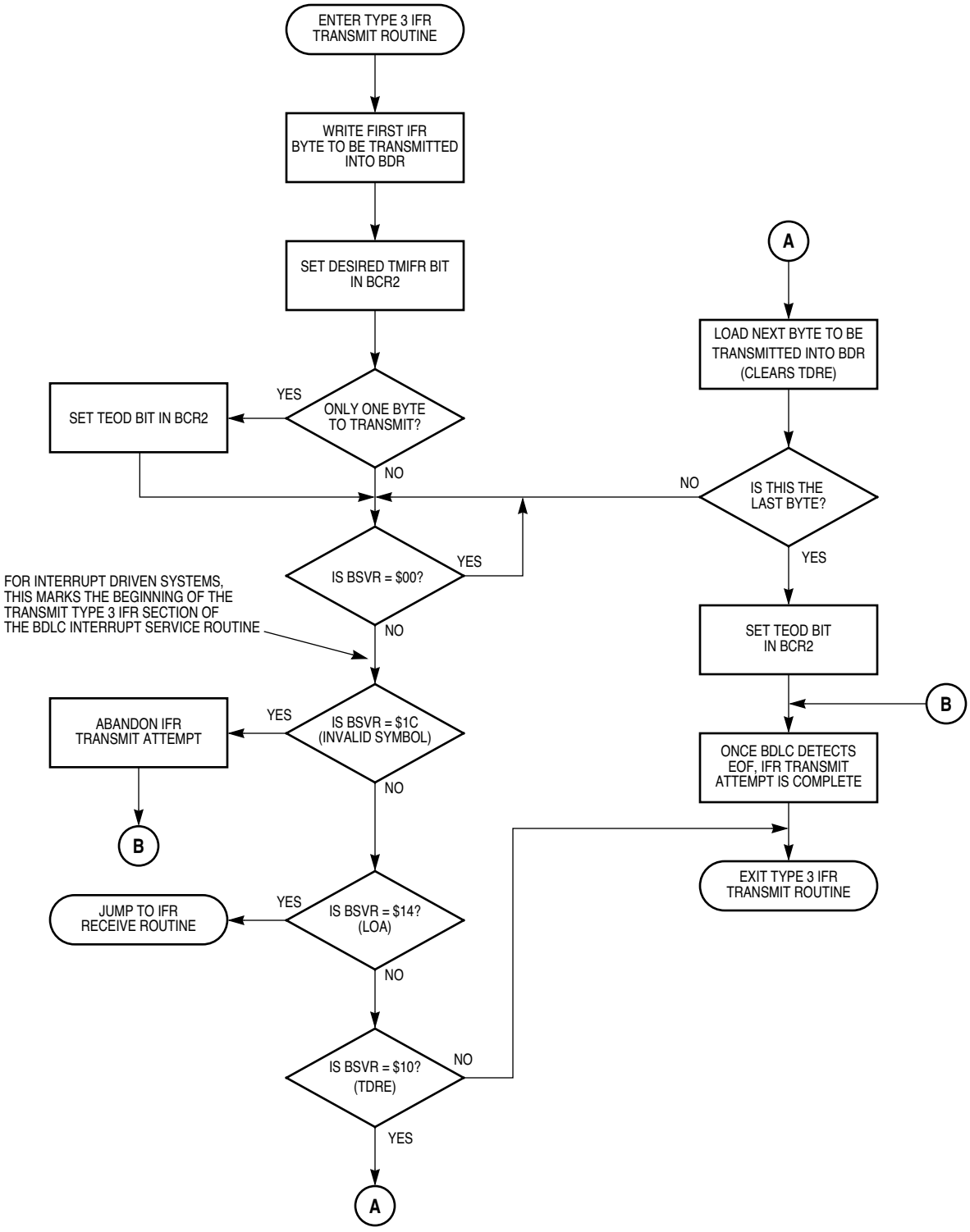

Figure 10-4 Transmitting a Type 3 IFR

10.3.3.1 Step 1: Load the First IFR Byte into the BDR

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 BDR. If a byte has already been written into the BDR for transmission as a new message, the user can simply write the first IFR byte to the BDR, replacing the previously written byte. This must be done before the first EOD symbol is received.

10.3.3.2 Step 2: Set the TMIFR Bit

The second step necessary for transmitting a type 3 IFR is to set the desired TMIFR bit in BCR2, depending upon whether or not a CRC is desired. 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. Refer to 10.2 BDLC IFR Transmit Control Bits for more information on the TMIFR1 and TMIFR0 bits.

Setting the TMIFR1 or TMIFR0 bit directs the BDLC 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 BSVR reflects that the next byte of the IFR can be written to the BDR (TDRE interrupt).

The TMIFR1 or TMIFR0 bit must be set before the EOD has been received by the BDLC or the bit remains cleared and the IFR byte is discarded.

10.3.3.3 Step 3: When TDRE is Indicated, Write the Next IFR Byte into the BDR

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

NOTE

As when transmitting a message, when transmitting an type 3 IFR the user writes two bytes to be transmitted into the BDR before the first RXIFR interrupt occurs. For this reason, the user should avoid using the receive IFR byte interrupts to control the sequencing of IFR bytes to be transmitted.

10.3.3.4 Step 4: Write the Last IFR Byte into the BDR and Set TEOD

Once the last IFR byte to be transmitted is written to the BDR, the user sets the TEOD bit in BCR2. Once the TEOD bit is set, after the last IFR byte written to the BDR is transmitted onto the bus, if the TMIFR1 bit has been set, the BDLC begins transmitting the CRC byte, followed by an EOD. If the TMIFR0 bit has been set, the last IFR byte is immediately followed by the transmission of an EOD. Following the EOD, and EOF is recognized and the message is complete.

If at any time during the transmission of a type 3 IFR a loss of arbitration occurs, the TMIFR bit is cleared and the TEOD bit (if set) is cleared. Any IFR byte being transmitted is discarded and the loss of arbitration state is reflected in the BSVR. Likewise, if an error is detected during the transmission of a type 3 IFR, the IFR control bits are cleared, the byte being transmitted is discarded, and the BSVR reflects 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 treats that byte as both the first and last IFR byte to be sent.

10.4 Transmitting IFR Exceptions

This basic IFR transmitting flow can be interrupted for most of 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 user failing to service a TDRE interrupt in a timely fashion. Refer to 8.4 Transmitting Exceptions for more information on how these exceptions can affect the IFR transmit process.

SECTION 11 RECEIVING AN IN-FRAME RESPONSE

Receiving an in-frame response with the BDLC is very similar to receiving a message frame. As each byte of an IFR is received, the BSVR indicates this to the user. An EOF indication in the BSVR indicates that the IFR (and message) is complete. Also, the IMSG bit can also be used to command the BDLC to ignore any further network activity, including IFR bytes being received, until the next valid SOF is received.

NOTE

As with a message transmission, the IMSG bit should never be used to ignore the BDLC's own IFR transmissions. This is again due to the BSVR bits being held in their reset state until IMSG is cleared, preventing the user from detecting any IFR-related state changes which may be of interest.

11.1 Receiving an IFR with the BDLC

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 BSVR. All other actions are the same. **Figure 11-1** displays a flowchart of the steps involved in receiving an IFR with the BDLC.

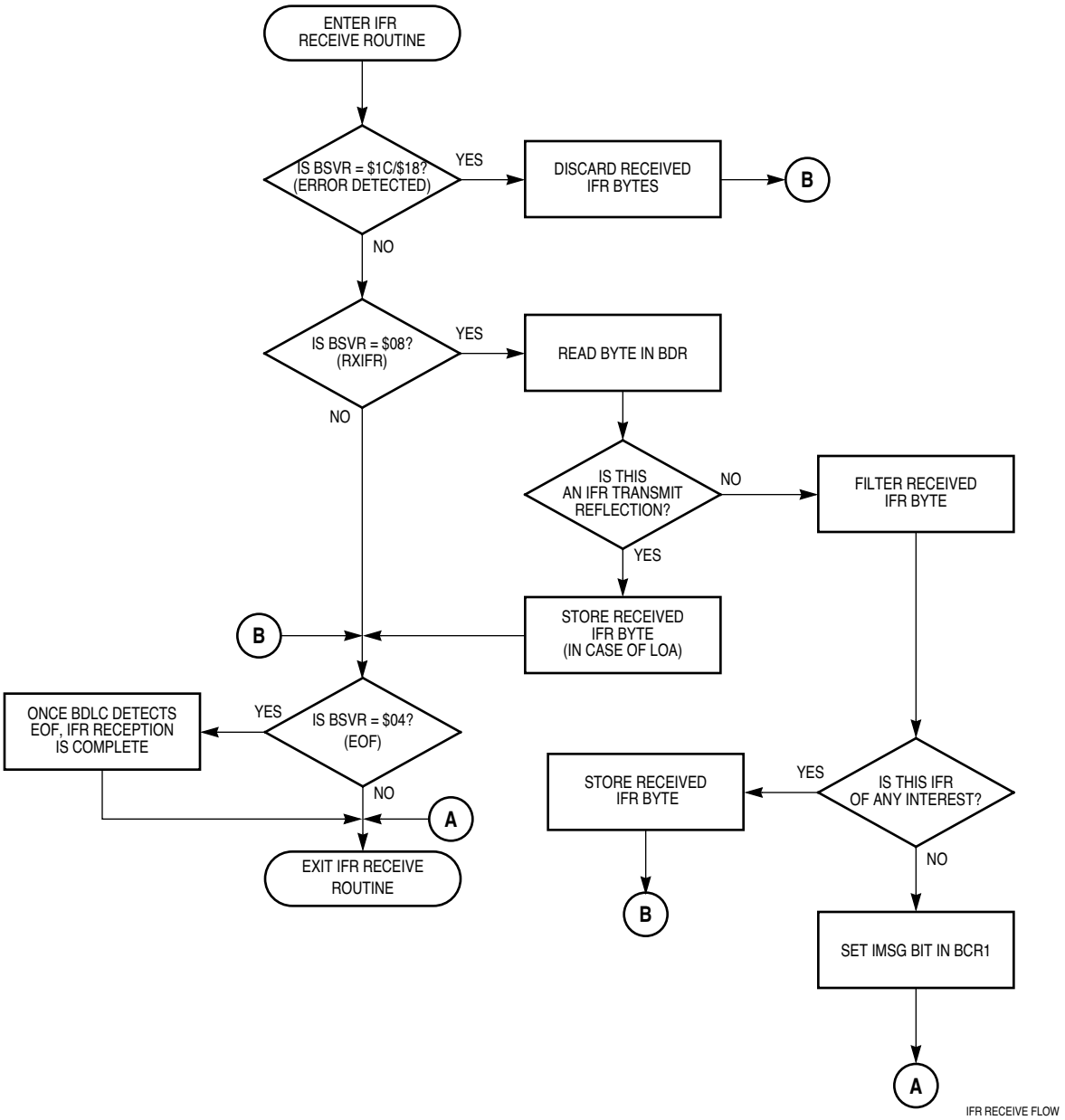

Figure 11-1 Receiving an IFR with the BDLC

11.1.1 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 BDR, and an RXIFR state is reflected in the BSVR. No indication of the EOD reception in made, since the RXIFR state indicates 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 BDR. Once this is done, no further user 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, are placed into the BDR as they are received for the user to retrieve.

11.1.2 Step 2: 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 remains idle for a time period equal to an EOD symbol. Following this, the BDLC determines whether or not the last byte of the IFR is a CRC byte, and if so verifies that the CRC byte is correct. Detection of an incorrect CRC byte is reflected in the BSVR.

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

11.2 Receiving IFR Exceptions

The 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 a receiver overrun caused by the user failing to service an RXIFR interrupt. For information on how these exceptions can affect the IFR receiving process, refer to 9.4 Receiving Exceptions.

SECTION 12SPECIAL OPERATIONS

There are a few special operations which the BDLC can perform, some of which fall outside the definitions of the SAE J1850 protocol. What follows is a brief description of each of these functions and how they might be used.

12.1 Transmitting or Receiving a Block Mode Message

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

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

12.2 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 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 41.6 kbps, allows these large blocks to be transmitted much more quickly.

The BDLC is designed to receive (but not transmit) messages transmitted at this higher speed. By setting the RX4XE bit in BCR2, the user can command the BDLC to receive any message sent over the network at a 4X rate.

If the BDLC is placed in this 4X mode, messages transmitted at the normal bit rate are not received correctly. Likewise, 4X messages transmitted on the SAE J1850 bus when the BDLC is in normal mode are interpreted as noise on the network by the BDLC. In either case, a symbol error may be indicated by the BDLC. The RX4XE bit is not affected by entry or exit from BDLC stop or wait modes. Refer to 4.1.2.3 Receive 4X Mode Enable Bit (RX4XE) for more information on the RX4XE bit.

12.3 Wakeup on Network Activity

The BDLC has two different power-conserving modes, BDLC stop and BDLC wait. Depending upon the logic level of the WCM bit in BCR1, the BDLC enters one of these modes whenever the CPU executes the STOP or WAIT instruction. Refer to **4.1.1.5 Wait Clock Mode Bit (WCM)** for more information on the function of the WCM bit.

When one of these two power-conserving modes has been entered, any activity on the network causes the BDLC to exit low-power mode. When exiting from the BDLC stop mode, the BDLC generates an unmaskable interrupt of the CPU. This wakeup interrupt state is reflected in the BSVR, encoded as the highest priority interrupt. This interrupt can be cleared by the user with a read of the BSVR.

Depending upon which low-power mode instruction the CPU executes and which mode the BDLC enters, the message which wakes up the BDLC (and the CPU) may or may not be received. There are three different possibilities, each of which is described below. These descriptions apply regardless of whether the BDLC is in normal or 4X mode when the STOP or WAIT instruction is executed.

12.3.1 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, are turned off. Therefore, the message which wakes up the BDLC and the CPU from stop mode will not be received by the BDLC. 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.

12.3.2 Wakeup from BDLC Stop with CPU in Wait

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

To ensure that the message causing the BDLC to exit the BDLC stop mode while the CPU is in wait mode is received correctly, the user must wait until the EOF following the last message appearing on the SAE J1850 bus has been received. If the EOF symbol has not been received and the CPU executes a WAIT instruction, placing the BDLC into stop mode, the next message activity on the SAE J1850 bus generates a Wake-Up interrupt of the CPU, but the message will not be received correctly.

Likewise, if the user desires to put the BDLC into this mode immediately following a reset of the MCU, the user must first wait until the BDLC has observed an EOF time on the bus before executing the WAIT instruction. An EOF interrupt is generated by the BDLC once this has occurred, if CPU interrupts are enabled and the IMSG bit in BCR1 is cleared.

NOTE

While the BDLC is designed to correctly receive a message which arrives when the BDLC is in BDLC stop 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 becomes corrupted. This is due to the steps required for the BDLC to resume operation upon exiting BDLC stop mode, and its subsequent resynchronization with the SAE J1850 bus.

12.3.3 Wakeup from BDLC Wait with CPU in Wait

If the CPU executes a WAIT instruction and the BDLC enters the BDLC wait mode (WCM = 0), the clocks to the BDLC as well as the clocks in the MCU continue to run. Therefore, the message which wakes up the BDLC from BDLC wait mode and the CPU from wait mode will also be received correctly by the BDLC. The is because all of the required clocks continue to run in the BDLC in BDLC wait mode.

12.4 Digital Loopback Mode

When a bus fault has been detected, the digital loopback mode can be used to determine if the fault condition is caused by a failure in the BDLC's internal circuits or elsewhere in the network, including the node's analog physical interface. In digital loopback mode, the transmit digital output and the receive digital input signals are isolated from the analog physical interface and connected together to allow the digital portion of the BDLC to transmit and receive its own messages without driving the physical layer transceiver.

Digital loopback mode is intended to provide the user with the capability of verifying that the BDLC module is operating properly. Entry into digital loopback mode is controlled by the DLOOP bit in BCR2. For a description of the operation of the DLOOP bit, refer to 4.1.2.2 Digital Loopback Mode Bit (DLOOP).

12.5 Analog Loopback Mode

Analog loopback mode is used to determine if a bus fault has been caused by a failure in a node's analog transceiver (integrated or external), or elsewhere in the network. The BCLD analog loopback mode does not modify the digital transmit or receive functions of the BDLC. It does, however, ensure that once analog loopback mode is exited, the BDLC will wait for an idle bus condition before participation in network communication resumes.

If an analog transceiver has a loopback mode, it usually causes the input to the output drive stage to be looped back into the receiver, allowing the node to receive messages it has transmitted without driving the SAE J1850 bus. In this mode, the output to the SAE J1850 bus is typically high impedance. This allows the communication path through the analog transceiver to be tested without interfering with network activity.

Using the BDLC analog loopback mode in conjunction with the analog transceiver's loopback mode ensures that, once the analog transceiver has exited loopback mode, the BDLC does not begin communicating before a known condition exists on the SAE J1850 bus. Entry into analog loopback mode is controlled by the ALOOP bit in BCR2. For a description of the operation of the ALOOP bit, refer to 4.1.2.1 Analog Loopback Mode Bit (ALOOP).

APPENDIX ASAE J1850 SYMBOL TIMINGS

This appendix contains SAE J1850 transmit and receive symbol timing specification tables for the BDLC. The values specified are for the symbols appearing on the SAE J1850 bus. These values assume that the BDLC is communicating on the SAE J1850 bus using an analog transceiver (external or integrated), and that the BDLC analog roundtrip delay value programmed into the BARD register is the appropriate value for the transceiver being used. If these conditions are not being 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 A-1 BDLC Transmitter VPW Symbol Timing

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	t_{TVP1}	62	64	66	µs
2	Passive Logic1	t_{TVP2}	126	128	130	µs
3	Active Logic 0	t_{TVA1}	126	128	130	µs
4	Active Logic 1	t_{TVA2}	62	64	66	µs
5	Start of Frame (SOF)	t_{TVA3}	198	200	202	µs
6	End of Data (EOD)	t_{TVP3}	198	200	202	µs
7	End of Frame (EOF)	t_{TV4}	278	280	282	µs
8	Inter-Frame Separator (IFS)	t_{TV6}	298	300	302	µs

Table A-2 BDLC Receiver VPW Symbol Timing

Number	Characteristic	Symbol[1]	Min	Typ	Max	Unit
1	Passive Logic 0	t_{RVP1}	34	64	96	µs
2	Passive Logic1	t_{RVP2}	96	128	163	µs
3	Active Logic 0	t_{RVA1}	96	128	163	µs
4	Active Logic 1	t_{RVA2}	34	64	96	µs
5	Start of Frame (SOF)	t_{RVA3}	163	200	239	µs
6	End of Data (EOD)	t_{RVP3}	163	200	239	µs
7	End of Frame (EOF)	t_{RV4}	239	280	320	µs
9	Inter-Frame Separator (IFS)	t_{RV6}	280	—	—	µs

NOTES:

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

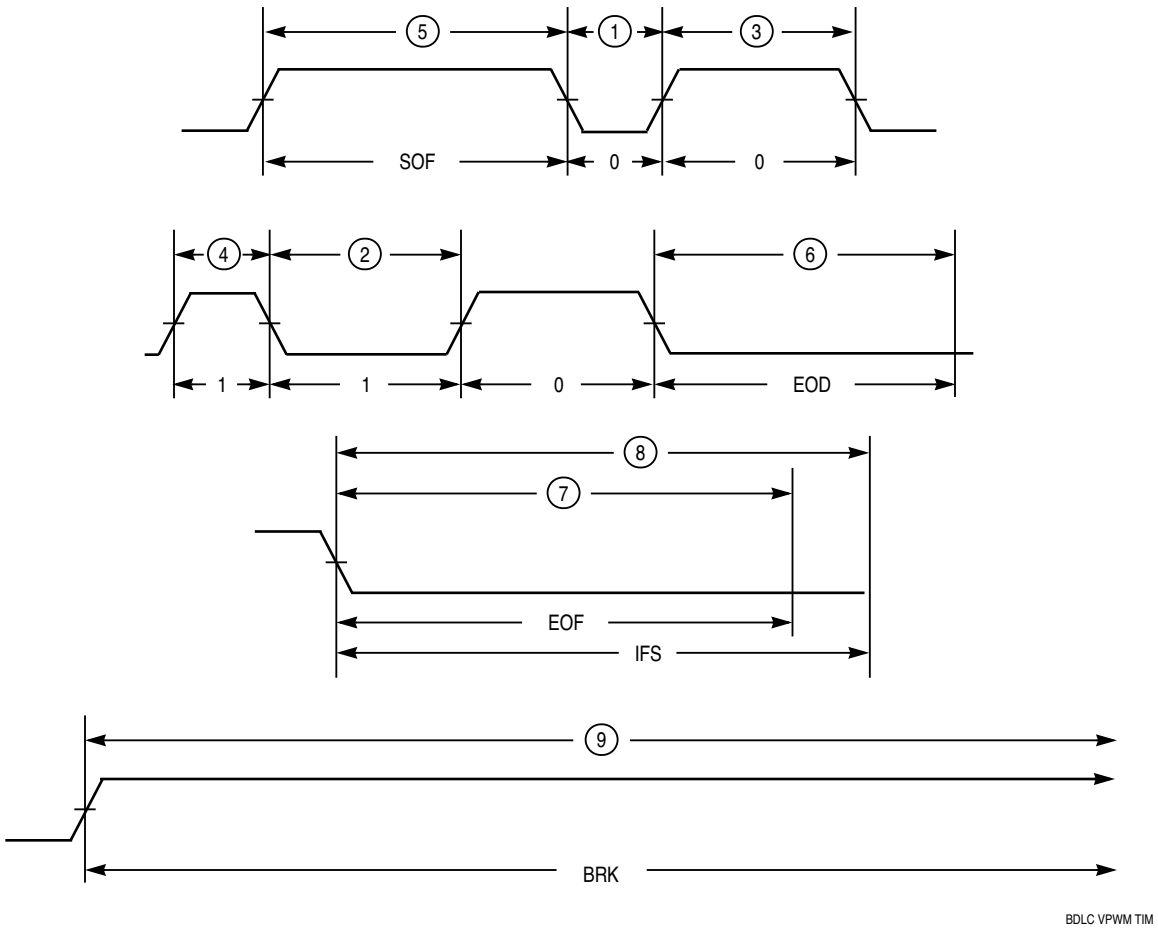

Figure A-1 BDLC Variable Pulse Width Modulation (VPW) Symbol Timing

APPENDIX BSAE J1850 PROTOCOL

This appendix explains the frame format, symbol format and arbitration techniques used to transmit and receive information on the SAE J1850 bus. Variable pulse width modulation (VPW) valid/invalid bits and symbols are also discussed. For a detailed description of the SAE J1850 protocol, refer to the document *SAE Standard J1850 Class B Data Communication Network Interface*, available from the Society of Automotive Engineers.

B.1 J1850 Frame Format

All messages transmitted on the SAE J1850 bus are structured using the format shown in **Figure B-1**.

The SAE J1850 protocol limits each VPW message to a maximum length of 12 bytes, excluding the SOF, EOD, NB, and EOF symbols. Each byte is transmitted MSB first.

All VPW symbol lengths in the following descriptions are nominal values at a 10.4 kbps bit rate.

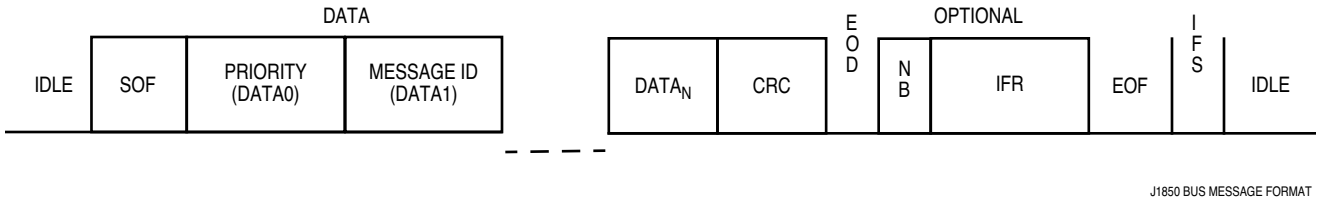

Figure B-1 J1850 Bus Message Format (VPW)

SOF — Start of Frame Symbol

All messages transmitted onto the J1850 bus must begin with an active 200 µs period SOF symbol. This indicates the start of a new message transmission. The SOF symbol is not used in the CRC calculation.

Data — In Message Data Bytes

Messages transmitted by the BDLC 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 eight bits in length and is transmitted most significant byte (MSB) to least significant byte (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. It 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 initially is set to all ones. Each byte in the message after the SOF symbol is processed serially 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 end of data symbols (EOD) but including the CRC byte, are used to check the CRC. If the message is error free, the remainder polynomial equals $X^7 + X^6 + X^2$ ($C4), regardless of the data contained in the message. If the calculated CRC does not equal $C4, the BDLC recognizes this as a CRC error and sets the CRC error flag in the BSVR register.

EOD — End of Data Symbol

The EOD symbol is a 200 μs 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 more information about the IFR option should review the *SAE Standard J1850 Class B Data Communication Network Interface* specification.

EOF — End of Frame Symbol

This symbol is a 280 μs passive period on the J1850 bus which signifies the end of a message. Since an EOF symbol is longer than a 200 μs 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 20 μs passive period on the J1850 bus which allows proper synchronization between nodes during continuous message transmission. The IFS symbol is transmitted by a node after the completion of the EOF period, and together they are cumulatively seen as a 300 μs passive 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 wait for the IFS symbol time to expire before transmitting an SOF symbol, marking the beginning of another message.

However, if the BDLC 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 synchronizes internally to that edge. If a write to the BDR register (for instance, to initiate transmission) occurs on or before $104 \bullet (1 / f_{bdlc})$ from the received rising edge, the BDLC transmits and arbitrates for the bus. If a write to the BDR register occurs after $104 \bullet (1 / f_{bdlc})$ from the detection of the rising edge, the BDLC does not attempt to transmit, but waits for the next IFS period to expire before attempting to transmit the byte.

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. To allow for individual clock tolerances, receivers must synchronize to any SOF occurring during an IFS period.

BREAK — Break

A BREAK symbol is typically transmitted onto the SAE J1850 bus to reset to normal communication mode any modules which have been previously placed into 4X mode. The BDLC cannot transmit a BREAK symbol.

If the BDLC is transmitting at the time a BREAK is detected, it treats the BREAK as a transmission error and immediately halts transmission.

If the BDLC detects a BREAK symbol while receiving a message, it treats the BREAK as a reception error and sets the invalid symbol flag in the BSVR, also ignoring the frame it was receiving. If while receiving a message in 4X mode, the BDLC detects a BREAK symbol, it treats the BREAK as a reception error, sets the invalid symbol flag, and exits 4X mode (for example, the RX4XE bit in BCR2 is cleared automatically).

IDLE — Idle Bus

An idle condition exists on the bus after expiration of the IFS period (≥ 300 μs). The idle condition lasts until a node begins transmitting onto the SAE J1850 bus. Any node sensing an idle bus condition can begin transmission immediately.

B.1.1 J1850 VPW Symbols

VPW modulation 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 reduce the number of bus transitions during a message transmission.

Each logic 1 or logic 0 contains a single transition and can be at either the active or passive level and one of two lengths, either 64 μs or 128 μs (nominal value at 10.4 kbps bit rate). The SOF, EOD, EOF, and IFS symbols are always encoded at an assigned level and length. Refer to **Figure B-2**.

Each message begins with an SOF symbol, an active symbol, and therefore each data byte (including the CRC byte and all IFR bytes) begins with a passive bit, regardless of whether it is a logic 1 or a logic 0.

All VPW bit lengths stated in the following descriptions are typical values at a 10.4 kbps bit rate.

B.1.1.1 Logic 0

A logic 0 is defined as either:

- An active-to-passive transition followed by a passive period 64 μs in length.
- A passive-to-active transition followed by an active period 128 μs in length.

Refer to **Figure B-2(a)**.

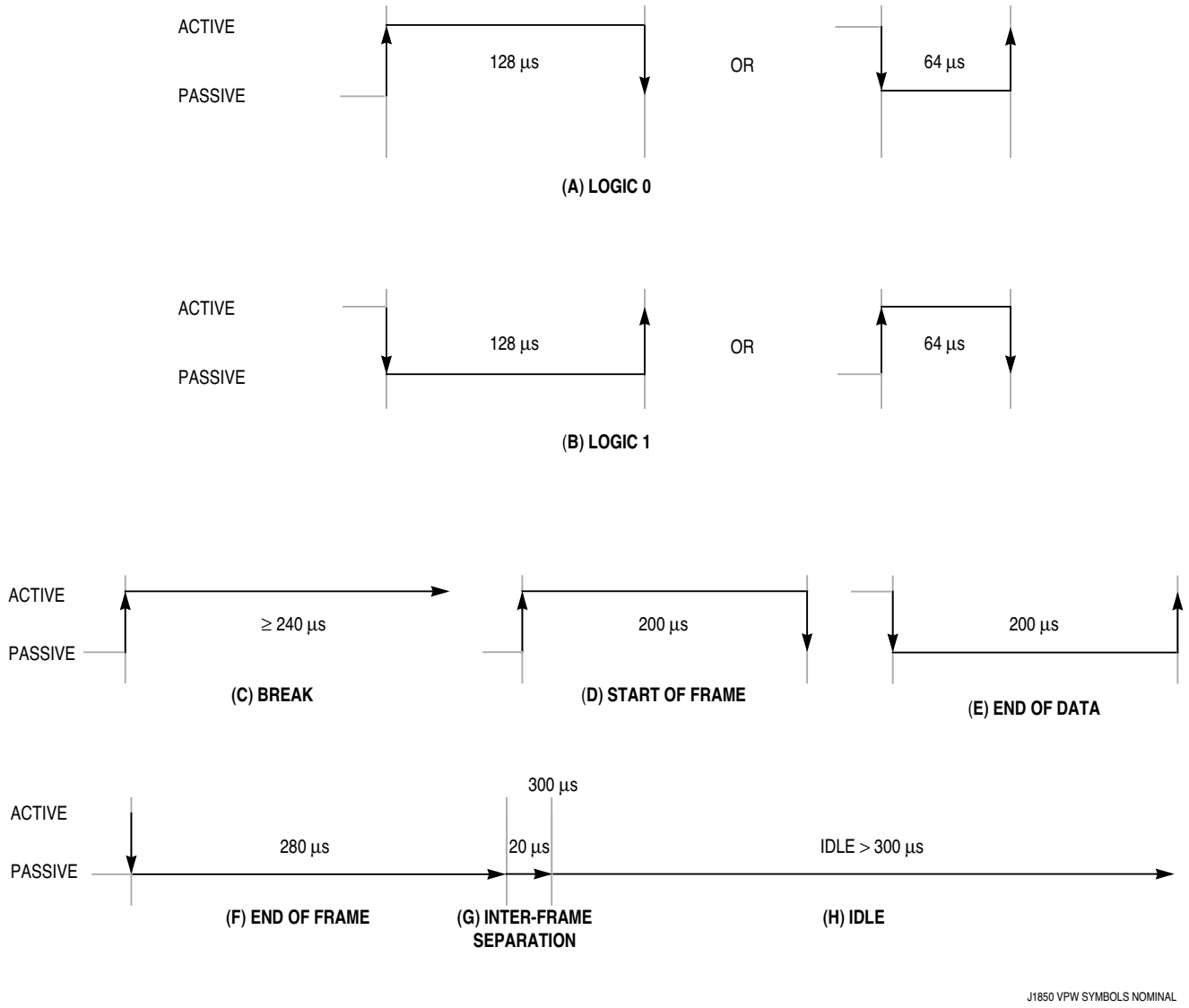

Figure B-2 J1850 VPW Symbols with Nominal Symbol Times

B.1.1.2 Logic 1

A logic 1 is defined as either:

- An active-to-passive transition followed by a passive period 128 µs in length
- A passive-to-active transition followed by an active period 64 µs in length

Refer to **Figure B-2(b)**.

B.1.1.3 Normalization Bit (NB)

The NB symbol has the same property as a logic 1 or a logic 0. It is only used during the optional in-frame response.

B.1.1.4 Break Signal (BREAK)

The BREAK signal is defined as a passive-to-active transition followed by an active period of at least 240 μs. Refer to **Figure B-2(c)**.

B.1.1.5 Start of Frame Symbol (SOF)

The SOF symbol is defined as passive-to-active transition followed by an active period 200 μs in length. Refer to **Figure B-2(d)**. This allows the data bytes which follow the SOF symbol to begin with a passive bit, regardless of whether it is a logic 1 or a logic 0.

B.1.1.6 End of Data Symbol (EOD)

The EOD symbol is defined as an active-to-passive transition followed by a passive period 200 μs in length. Refer to **Figure B-2(e)**.

B.1.1.7 End of Frame Symbol (EOF)

The EOF symbol is defined as an active-to-passive transition followed by a passive period 280 μs in length. Refer to **Figure B-2(f)**. If no IFR byte is transmitted after an EOD symbol is transmitted, after another 80 μs the EOD becomes an EOF, indicating completion of the message.

B.1.1.8 Inter-Frame Separation Symbol (IFS)

The IFS symbol is defined as a passive period 20 μs in length. The 20 μs IFS symbol contains no transition, since when used it is always appended to an EOF symbol . Refer to **Figure B-2(g)**.

B.1.1.9 Idle Bus

An idle bus is defined as a passive period greater than 300 μs in length.

B.1.2 J1850 VPW Valid/Invalid Bits and Symbols

The timing tolerances for receiving data bits and symbols from the J1850 bus have been defined to allow for variations in oscillator frequencies and ground offsets between modules. In many cases the maximum specified length of a data bit or symbol is equal to the minimum specified length of another data bit or symbol. For example, the maximum length of a passive logic 0 is equal to the minimum length of a passive logic 1, and the maximum length of an active logic 0 is equal to the minimum length of a valid SOF symbol

Since the minimum timing resolution the BDLC uses to determine what symbol is being received is equal to a single period of the MUX interface clock ($1 / f_{bdlc}$), an apparent separation in these maximum time/minimum time concurrences equal to one cycle of $1 / f_{bdlc}$ will occur.

This one clock resolution allows the BDLC to differentiate properly between the different bits and symbols. This is done without reducing the valid window for receiving bits and symbols transmitted onto the J1850 bus from transmitters which have varying oscillator frequencies.

B.1.2.1 Invalid Passive Bit

Refer to **Figure B-3(1)**. If the passive-to-active received transition beginning the next data bit or symbol occurs between the active-to-passive transition beginning the current data bit (or symbol) and **a**, the current bit is invalid.

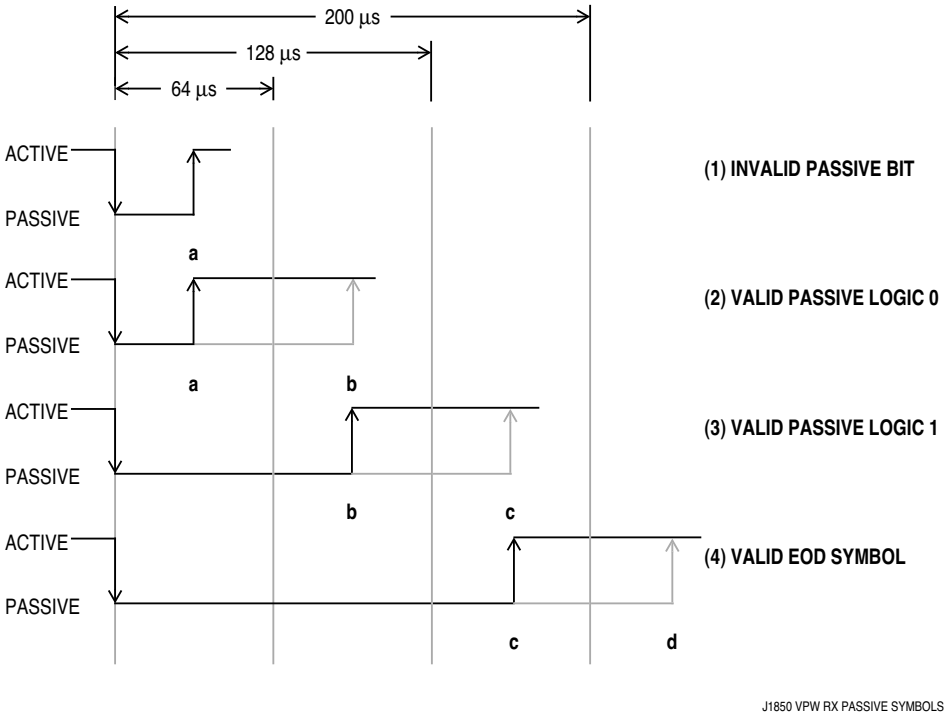

Figure B-3 J1850 VPW Received Passive Symbol Times

B.1.2.2 Valid Passive Logic 0

Refer to **Figure B-3(2)**. If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **a** and **b**, the current bit is considered a logic 0.

B.1.2.3 Valid Passive Logic 1

Refer to **Figure B-3(3)**. If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **b** and **c**, the current bit would be considered a logic 1.

B.1.2.4 Valid EOD Symbol

Refer to **Figure B-3(4).** If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **c** and **d**, the current symbol is considered a valid EOD symbol.

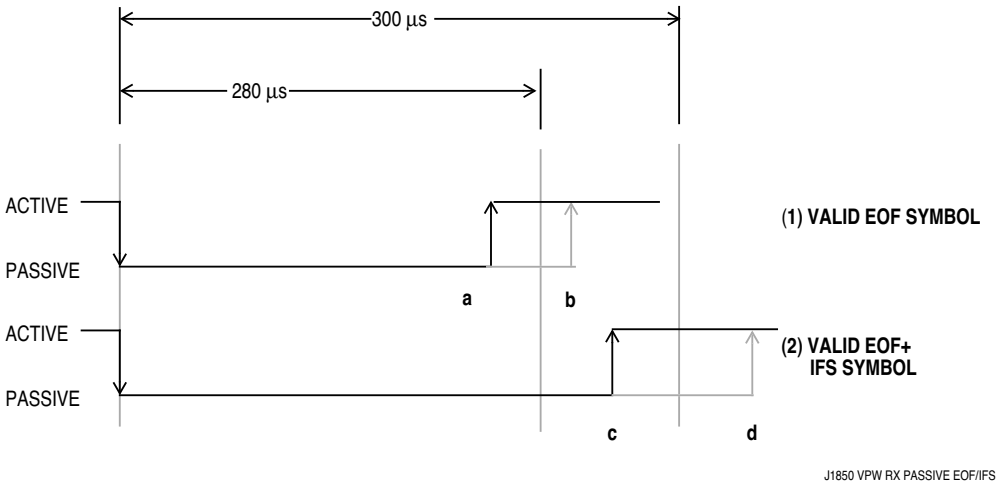

Figure B-4 J1850 VPW Received Passive EOF and IFS Symbol Times

B.1.2.5 Valid EOF and IFS Symbol

In **Figure B-4(1)**, if the passive-to-active received transition beginning the SOF symbol of the next message occurs between **a** and **b**, the current symbol is considered a valid EOF symbol.

Refer to **Figure B-4(2)**. If the passive-to-active received transition beginning the SOF symbol of the next message occurs between **c** and **d**, the current symbol is considered a valid EOF symbol followed by a valid IFS symbol. 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, any time 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.

B.1.2.6 Idle Bus

In **Figure B-4(2)**, if the passive-to-active received transition beginning the SOF symbol of the next message does not occur before **d,** the bus is considered to be idle, and any node wishing to transmit a message may do so immediately.

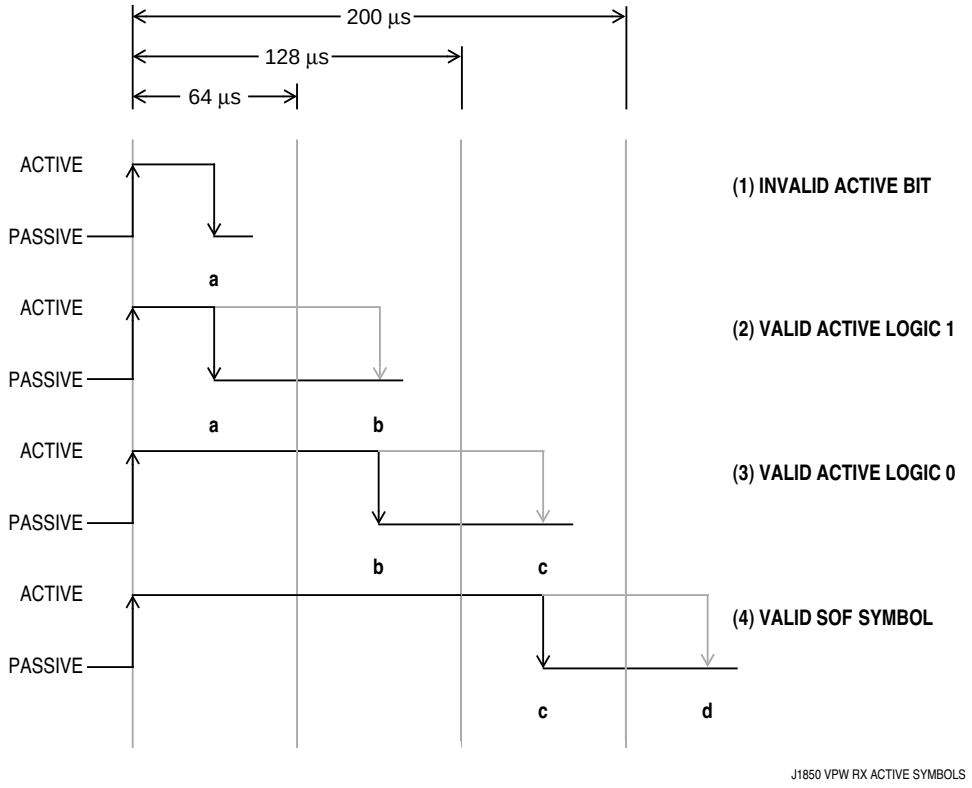

Figure B-5 J1850 VPW Received Active Symbol Times

B.1.2.7 Invalid Active Bit

In **Figure B-5(1)**, if the active-to-passive received transition beginning the next data bit (or symbol) occurs between the passive-to-active transition beginning the current data bit (or symbol) and **a**, the current bit is invalid.

B.1.2.8 Valid Active Logic 1

In **Figure B-5(2)**, if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **a** and **b**, the current bit is considered a logic 1.

B.1.2.9 Valid Active Logic 0

In **Figure B-5(3)**, if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **b** and **c**, the current bit is considered a logic 0.

B.1.2.10 Valid SOF Symbol

In **Figure B-5(4)**, if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **c** and **d**, the current symbol is considered a valid SOF symbol.

B.1.2.11 Valid BREAK Symbol

In **Figure B-6**, if the next active-to-passive received transition does not occur until after **e**, the current symbol is considered a valid BREAK symbol. Refer to B.1 J1850 Frame Format for BDLC handling of BREAK symbols.

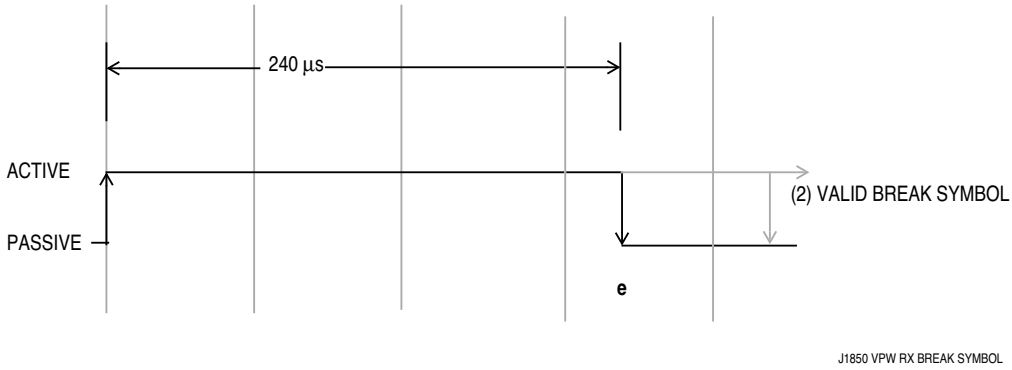

Figure B-6 J1850 VPW Received BREAK Symbol Times

B.1.3 Message Arbitration

Message arbitration on the J1850 bus is accomplished in a nondestructive 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 wants to transmit onto the J1850 bus, but detects that another message is in progress, it waits until the bus is idle. However, if multiple nodes begin to transmit in the same synchronization window, the arbitration process begins with the first bit after the SOF symbol and continues with each bit thereafter.

The VPW symbols and J1850 bus electrical characteristics have been chosen carefully so that a logic 0 (active or passive) always dominates over a logic 1 (active or passive) simultaneously transmitted. Therefore, logic 0’s are said to be dominant and logic 1’s are said to be recessive.

Whenever a node detects a dominant bit when it has transmitted a recessive bit, it loses arbitration and immediately stops transmitting. This bitwise arbitration continues throughout the entire message, if necessary. Refer to **Figure B-7**.

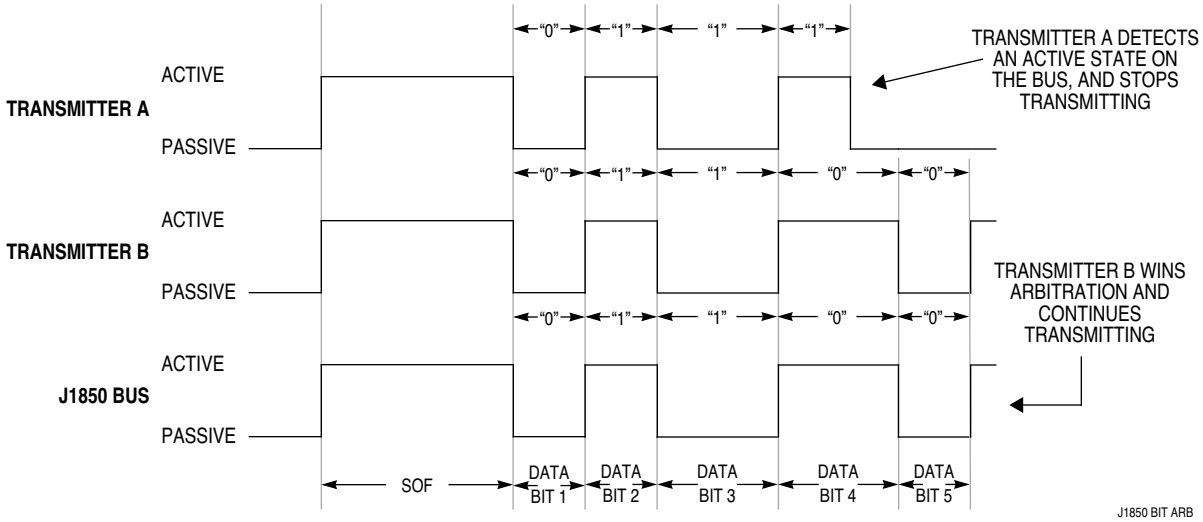

Figure B-7 J1850 VPW Bitwise Arbitrations

Since a logic 0 dominates a logic1, the message with the lowest value has the highest priority, and always wins arbitration. For instance, a message with priority %000 wins arbitration over a message with priority %011. This is true regardless of how many bits of priority encoding are contained in a message.

APPENDIX CREGISTER SUMMARY

This appendix contains the register map, register diagrams, and bit/field definitions for the BDLC. More detailed information about register function is provided in the appropriate sections of the manual.

C.1 BDLC Register Map

Table C-1 shows the BDLC register map. These registers are common to all implementations of the BDLC. Certain MCU implementations contain additional registers which may be used to enable the BDLC or modify its outputs. For information on these device-specific registers, refer to the appropriate device specification.

Table C-1 BDLC Register Map

Address	7 0
$######	BDLC Control Register 1 (BCR1)
$######	BDLC State Vector Register (BSVR)
$######	BDLC Control Register 2 (BCR2)
$######	BDLC Data Register (BDR)
$######	BDLC Analog Roundtrip Delay Register (BARD)

NOTE

Depending upon the MCU implementation, the BDLC registers may not be implemented in this order, or may not even be contiguous in the MCU register map. Refer to the specific device register map for BDLC register placement.

C.1.1 BDLC Control Register 1

BCR1 — BDLC Control Register 1

7	6	5	4	3	2	1	0
IMSG	CLKS	RS[1:0]		0	0	IE	WCM
RESET							
1	1	1	0	0	0	0	0

IMSG — Ignore Message

IMSG is used when the CPU has determined that a message being received is of no interest.

0 = BDLC interrupts of the CPU are enabled.

1 = BDLC interrupts of the CPU are inhibited until the next SOF symbol is received.

CLKS — Clock Select

CLKS is used in conjunction with RS[1:0] to prescale the MCU system clock frequency to the necessary f_{bdlc} frequency. This bit can be written only once following a reset of the MCU.

0 = f_{bdlc} is 1.00 MHz.
1 = f_{bdlc} is 1.049 MHz.

RS[1:0] — Rate Select Field

RS[1:0] are used with CLKS to properly prescale the MCU system clock frequency (f_{sys}) to produce the BDLC operating clock (f_{bdlc}). While CLKS indicates to the prescaler circuitry whether a binary or integer base frequency is used, R[1:0] provide the necessary prescale to divide the system clock frequency down to the required BDLC operating frequency.

The f_{sys} used to produce the BDLC operating clock is either based on the oscillator frequency (f_{osc}) or on the oscillator frequency ÷ 2 ($f_{osc}/2$), depending upon the MCU family.

The BDLC operating clock is derived using the equation:

$$f_{bdlc} = f_{sys} \div \text{PRESCALER VALUE}$$

with f_{sys} derived as follows, depending upon the MCU family:

$$\text{HC05: } f_{sys} = f_{osc} \div 2$$

$$\text{HC08: } f_{sys} = f_{osc}$$

$$\text{HC12: } f_{sys} = f_{osc} \div 2$$

Refer to the particular device specification for more information on the system clock frequency used to arrive at the BDLC operating frequency.

Tables C-2 and **C-3** give the appropriate RS[1:0] values for selecting the correct prescaler value necessary to achieve f_{bdlc}, depending upon the f_{sys} as derived above.

NOTE

On MCUs which contain a PLL circuit to generate the internal operating frequency of the MCU, the frequency input into the BDLC prescaling circuit should not be derived from the PLL.

Table C-2 BDLC Rate Selection for Integer Frequencies (CLKS = 0)

f_{sys}	R1	R0	Prescale Value	f_{bdlc}
1.00 MHz	0	0	1	1.00 MHz
2.00 MHz	0	1	2	1.00 MHz
4.00 MHz	1	0	4	1.00 MHz
8.00 MHz	1	1	8	1.00 MHz

Table C-3 BDLC Rate Selection for Binary Frequencies (CLKS = 1)

f_{sys}	R1	R0	Prescale Value	f_{bdlc}
1.049 MHz	0	0	1	1.049 MHz
2.097 MHz	0	1	2	1.049 MHz
4.194 MHz	1	0	4	1.049 MHz
8.389 MHz	1	1	8	1.049 MHz

These bits can be written only once following a reset of the MCU.

IE — Interrupt Enable

IE enables BDLC interrupts of the CPU. It can be written at any time and may be set just once or set and cleared repeatedly, depending upon the application.

0 = Only a wakeup interrupt from wait (with WCM = 1) or stop mode causes an interrupt of the CPU. All other interrupts are masked, and the user needs to poll the BSVR to determine the status of the BDLC.

1 = All BDLC interrupt sources cause an interrupt of the CPU.

WCM —Wait Clock Mode

WCM determines the operation of the BDLC internal clocks during the CPU wait mode. This bit can be written only once following a reset of the MCU.

0 = The BDLC internal clocks continue to run when the CPU is in WAIT mode.

1 = The BDLC internal clocks stop when a WAIT instruction is executed by the CPU.

C.1.2 BDLC State Vector Register

BSVR — BDLC State Vector Register

7	6	5	4	3	2	1	0
0	0	ISF[3:0]				0	0
RESET							
0	0	0	0	0	0	0	0

ISF[3:0] — Interrupt Sources Field

ISF[3:0] indicate BSVR interrupt sources. **Table C-4** shows the status encoding and priority assignments for all BDLC interrupt sources.

NOTE

Interrupt source 0 (no interrupts pending) cannot generate an interrupt of the CPU, so an interrupt-based routine does not perform any tasks based on this state.

Table C-4 BSVR Interrupt Sources

BSVR	ISF3	ISF2	ISF1	ISF0	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 (RXIFR)	2
$0C	0	0	1	1	Rx data register full (RDRF)	3
$10	0	1	0	0	Tx data register empty (TDRE)	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)

C.1.3 BDLC Control Register 2

BCR2 — BDLC Control Register 2

7	6	5	4	3	2	1	0
ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
RESET							
1	1	0	0	0	0	0	0

ALOOP — Analog Loopback Mode

ALOOP indicates to the BDLC that the analog transceiver has been placed into an analog loopback mode, where its output driver is bypassed internally and the transmitted signal is looped back to the receive comparator.

0 = The BDLC waits for the bus to be idle for at least one EOF time period before it begins to receive messages, and waits for the bus to be idle for at least one inter-frame separation time period before it begins to transmit messages. An EOF interrupt is generated (if interrupts are enabled), indicating to the CPU that the BDLC is ready to receive valid SAE J1850 messages.

1 = The BDLC continues to operate normally.

DLOOP — Digital Loopback Mode

DLOOP isolates the analog transceiver in order to determine whether a nodes' transceiver is causing a disruption on the bus.

0 = The BDLC waits for the bus to be idle for at least one EOF time period before it begins to receive messages and waits for the bus to be idle for at least one inter-frame separation time period before it begins to transmit messages. An EOF interrupt is generated (if interrupts are enabled), indicating to the CPU that the BDLC is ready to receive valid SAE J1850 messages.

1 = The analog transceiver is by-passed, with the transmitted digital signal being passed directly back to the digital receive circuitry.

RX4XE — Receive 4X Mode Enable

RX4XE allows the BDLC to receive SAE J1850 - VPW messages transmitted at four times the normal rate (41.6 kbps).

0 = The BDLC operates in its normal communicating mode.

1 = The BDLC is placed into 4X receive mode, and only receives messages transmitted in 4X mode.

NBFS — Normalization Bit Format Select

NBFS determines the format of the normalization bit used by the BDLC while transmitting or receiving in-frame responses on the SAE J1850 network.

0 = An IFR with a CRC byte is preceded by an active logic “1” symbol, and an IFR with no CRC begins with an active logic “0” symbol.

1 = The BDLC transmits (and expects to receive) an active logic “0” symbol preceding an IFR which contains a CRC byte, and an active logic “1” symbol preceding an IFR which contains no CRC byte.

TEOD — Transmit End of Data

TEOD indicates to the BDLC that the last byte of that part of the message frame has been loaded into the BDR.

0 = Indicates the first bit of the CRC byte is being transmitted, or that an error or loss of arbitration has been detected on the bus. The user cannot clear this bit.

1 = Indicates to the BDLC 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 this bit also inhibits any further TDRE interrupts until TEOD is cleared. This bit is set by the user.

TSIFR — Transmit Single Byte In-Frame Response

TSIFR is used to transmit type 1 and type 2 IFRs onto the SAE J1850 bus.

0 = Indicates a successful transmission of the byte as an IFR when the TEOD bit is set, or that a transmission error was detected.

1 = Indicates to the BDLC that the byte in the BDR should be transmitted, preceded by the appropriate normalization bit, as a type 1 or type 2 IFR. This bit is set by the user.

TMIFR1 — Transmit Multi-Byte In-Frame Response 1

TMIFR1 is used to transmit an SAE J1850 type 3 IFR with a CRC byte appended.

0 = Indicates that an IFR, with a CRC byte appended, has been transmitted, or that an error has been detected on the bus, or that a loss of arbitration occurred.

1 = Indicates to the BDLC that a multi-byte IFR with a CRC byte appended, preceded by the appropriate normalization bit, is to be transmitted onto the SAE J1850 bus.

TMIFR0 — Transmit Multi-Byte In-Frame Response 0

TMIFR0 is used to transmit an SAE J1850 type 3 IFR without a CRC byte appended.

0 = Indicates that an IFR, without a CRC byte appended, has been transmitted, or that an error has been detected on the bus, or that a loss of arbitration occurred.

1 = Indicates to the BDLC that a multi-byte IFR without a CRC byte appended, preceded by the appropriate normalization bit, is to be transmitted onto the SAE J1850 bus.

C.1.4 BDLC Data Register

BDR — BDLC Data Register

7	6	5	4	3	2	1	0
BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0
RESET							
U	U	U	U	U	U	U	U

BDR 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 BDR, and bytes received from the bus by the BDLC are read from the BDR. 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 is the last byte placed in the BDR by the BDLC, not the last byte written to the BDR by the CPU.

C.1.5 BDLC Analog Roundtrip Delay Register

BARD — BDLC Analog Roundtrip Delay Register

7	6	5	4	3	2	1	0
ATE	RXPOL	0	0	BO[3:0]			
RESET							
1	1	0	0	0	1	1	1

ATE — Analog Transceiver Enable

ATE selects either an integrated or stand-alone SAE J1850 analog transceiver.

0 = An external analog transceiver is selected.

1 = An integrated analog transceiver is selected.

RXPOL — Receive Polarity

RXPOL selects the polarity of the digital receive data coming from an external SAE J1850 transceiver. When using an integrated analog transceiver, this bit has no affect on BDLC operation.

0 = Data coming from the analog transceiver is expected to be inverted from a true polarity.

1 =Receive data from the external transceiver is expected to be a true, non-inverted signal.

BO[3:0] — BDLC Analog Roundtrip Delay Offset

BO[3:0] adjust the transmitted bit and symbol timings to account for the differing roundtrip delays found in different SAE J1850 analog transceivers. The allowable delay range is from 9µs to 24µs, with a nominal target of 16 µs (reset value). Refer to **Table C-5** for the BO[3:0] values corresponding to the expected transceiver delays. Refer to the analog transceiver device specification for the expected roundtrip delay though both the transmitter and the receiver. The sum of these two delays makes up the total roundtrip delay value.

Table C-5 BO[3:0] Offset Values

BO3	BO2	BO1	BO0	Delay
0	0	0	0	9 µs
0	0	0	1	10 µs
0	0	1	0	11 µs
0	0	1	1	12 µs
0	1	0	0	13 µs
0	1	0	1	14 µs
0	1	1	0	15 µs
0	1	1	1	16 µs
1	0	0	0	17 µs
1	0	0	1	18 µs
1	0	1	0	19 µs
1	0	1	1	20 µs
1	1	0	0	21 µs
1	1	0	1	22 µs
1	1	1	0	23 µs
1	1	1	1	24 µs

–A–

Abnormal bit 8-6

ALOOP 4-4, 12-3, C-4
Analog
    loopback mode (ALOOP) 4-4, 12-3, C-4
    transceiver enable (ATE) 4-6, C-6
ATE 4-6, C-6

–B–

BARD 3-2, 4-6, C-6
BCR1 3-1, 4-1, C-1
BCR2 3-2, 4-4, 8-1, C-4
BDLC 1-1
    analog roundtrip delay
        offset field (BO) 4-7, C-6
        register (BARD) 3-2, 4-6, C-6
    block diagram 1-1
    control register 1 (BCR1) 3-1, 4-1, C-1
    control register 2 (BCR2) 3-2, 4-4, C-4
    data register (BDR) 3-2, 8-1, C-6
    features 1-1
    operating clock frequency (f_{bdlc}) 4-2, 7-2
    register map 3-1, C-1
    state vector register (BSVR) 3-1, 5-1, C-3
    stop 12-2
    wait 12-2
BDR 3-2, 8-1, C-6
Block mode message 12-1
BO 4-7, C-6
BREAK symbol 8-7, 9-5, B-3
BSVR 3-1, 5-1, C-3
    interrupt sources 5-2, C-4
    polling 5-7
    status encoding 5-1
Bus fault 8-7, 9-4
Byte data link controller (BDLC) 1-1

–C–

Clearing mechanisms 5-2
CLKS 4-2, C-2
Clock select (CLKS) 4-2, C-2
Configuration bits 4-1
CPU interface
    architecture 3-1
    block 1-3
CRC 8-1, 8-8, 12-1, B-2
Cyclical redundancy check (CRC) 8-8

–D–

Data B-1
Digital loopback
    mode (DLOOP) 4-5, 12-3, C-4
    multiplexer 7-3
DLOOP 4-5, 12-3, C-4

–E–

End
    of data (EOD) 8-1, B-2
    of frame (EOF) B-2
EOD 8-1, 9-3, B-2
EOF B-2

–F–

f_{bdlc} 4-2, 7-2, 7-3, C-2
f_{OSC} 4-2, C-2
Framing error 8-7, 9-4
f_{sys} 4-2, C-2

–I–

IDLE B-3
Idle bus condition B-3
IE 4-4, C-3
IFR 8-1, 8-8, 10-1
IFS B-2
Ignore message (IMSG) 4-2, C-1
IMSG 4-2, 9-1, C-1
In-frame response (IFR) 8-1, B-2
    response to a received message 9-5
    transmit control bits 10-3
    transmitting 10-1
    types 10-1–10-3
Initialization 4-8–4-12
Inter-frame separation (IFS) B-2
Interrupt
    enable (IE) 4-4, C-3
    sources field (IFS) C-3
Invalid bit 8-6
ISF C-3

–L–

Least significant byte (LSB) B-1
LOA 8-6

Loss of arbitration (LOA) 8-6
LSB B-1

–M–

M68HC05 5-3
M68HC08 5-3
M68HC11 5-4
M68HC12 5-4
Message
    arbitration B-9
    data bytes B-1
Most significant byte (MSB) B-1
MSB B-1
MUX interface 1-3, 7-1
    block diagram 7-1
    operation 7-2

–N–

NBFS 4-6, C-5
Normalization bit format select (NBFS) 4-6, C-5

–O–

Operating modes
    BDLC stop mode 2-2
    BDLC wait mode 2-2
    power off mode 2-1
    reset mode 2-1
    run mode 2-2

–P–

Power-conserving modes 12-2
Protocol
    handler
        architecture 6-1
        block 1-3
        block diagram 6-1
    state machine 6-1

–R–

Rate select field (RS) 4-2, C-2
RDRF 5-2, 8-5, 9-3
Receive
    4X mode enable (RX4XE) 4-5, C-5
    polarity (RXPOL) 4-7, C-6
    sequence 9-1–9-3
        flowchart 9-2
Received IFR byte (RXIFR) 5-2, 11-1
Receiver overrun 9-5
Receiving
    a block mode message 12-1
    a message in 4X mode 12-1
    an in-frame response
        flowchart 11-2
        sequence 11-1–11-3
    exceptions 9-4
    IFR exceptions 11-3
    messages 9-1
        filtering 9-3
Reception
    control 9-1
    errors
        BREAK symbol 9-5
        bus fault 9-4
        framing 9-4
        symbol 9-4
RS 4-2, C-2
RX
    data register full (RDRF) 5-2
    digital filter
        block 7-2
        block diagram 7-2
    shadow register 6-2
    shift register 6-2
RX4XE 4-5, 12-1, C-5
RXIFR 5-2, 11-1, 11-3
RXPOL 4-7, C-6

–S–

SAE J1850
    bus 7-2, 8-1, 8-4, 9-1
    frame format B-1
    message limits 12-1
    protocol B-1
    symbol timings A-1
SAE STANDARD J1850 CLASS B DATA COMMUNICATION NETWORK INTERFACE 1-1, B-2
Society of Automotive Engineers (SAE) 1-1
SOF 8-4, 9-2, B-1
Special operations 12-1
Start of frame (SOF) 8-4, B-1
State vector handling software 5-3–5-7
STOP instruction 12-2
Symbol
    encoder/decoder 7-1
    error 8-6, 9-4
System clock frequency (f_{sys}) 4-2

–T–

t_{bdlc} 7-3
TDRE 5-2, 8-1, 8-4
TEOD 8-1, 8-4, 10-3, 10-6, C-5
TMIFR0 10-3, 10-5, C-5
TMIFR1 10-3, 10-4, C-5
Transmission
    aborting 8-8
    control 8-1
    errors 8-6
Transmit
    end of data (TEOD) 8-1, C-5
    flowchart 8-3

multi-byte IFR 0 (TMIFR0) 10-5, C-5
multi-byte IFR 1 (TMIFR1) 10-4, C-5
sequence 8-3–8-4
single byte IFR (TSIFR) 10-3, C-5
Transmitter underrun 8-8
Transmitting
a block mode message 12-1
a type 1 IFR 10-6–10-7
flowchart 10-6
a type 2 IFR 10-7–10-9
flowchart 10-8
a type 3 IFR 10-9–10-12
flowchart 10-10
exceptions 8-5
IFR exceptions 10-12
TSIFR 10-3, 10-6, C-5
TX
data register empty (TDRE) 5-2, 8-1
shadow register 6-2
shift register 6-2

–V–

Variable pulse width (VPW) B-1
maximum message length B-1
V_{DD} 2-1
VPW B-1
bit lengths B-3
bitwise arbitration B-10
invalid bits and symbols B-5–B-9
received
active symbol times B-8
BREAK symbol times B-9
passive EOF and IFS symbol times B-7
passive symbol times B-6
symbols B-3–B-5

–W–

Wait clock mode (WCM) 4-4, C-3
WAIT instruction 12-2
Wakeup
from BDLC stop with CPU in stop 12-2
from BDLC stop with CPU in wait 12-2
from BDLC wait with CPU in wait 12-3
WCM 4-4, 12-2, C-3