

---

## J1850 BDLC PWM Core

---

June 12, 2009

Product Specification

---



### Drivven, Inc.

14027 N. Hills Village Dr.  
San Antonio, Texas  
USA, 78249  
Phone: +1 210 269 4667  
Fax: +1 761 661 6542  
E-mail: info@drivven.com  
URL: www.drivven.com

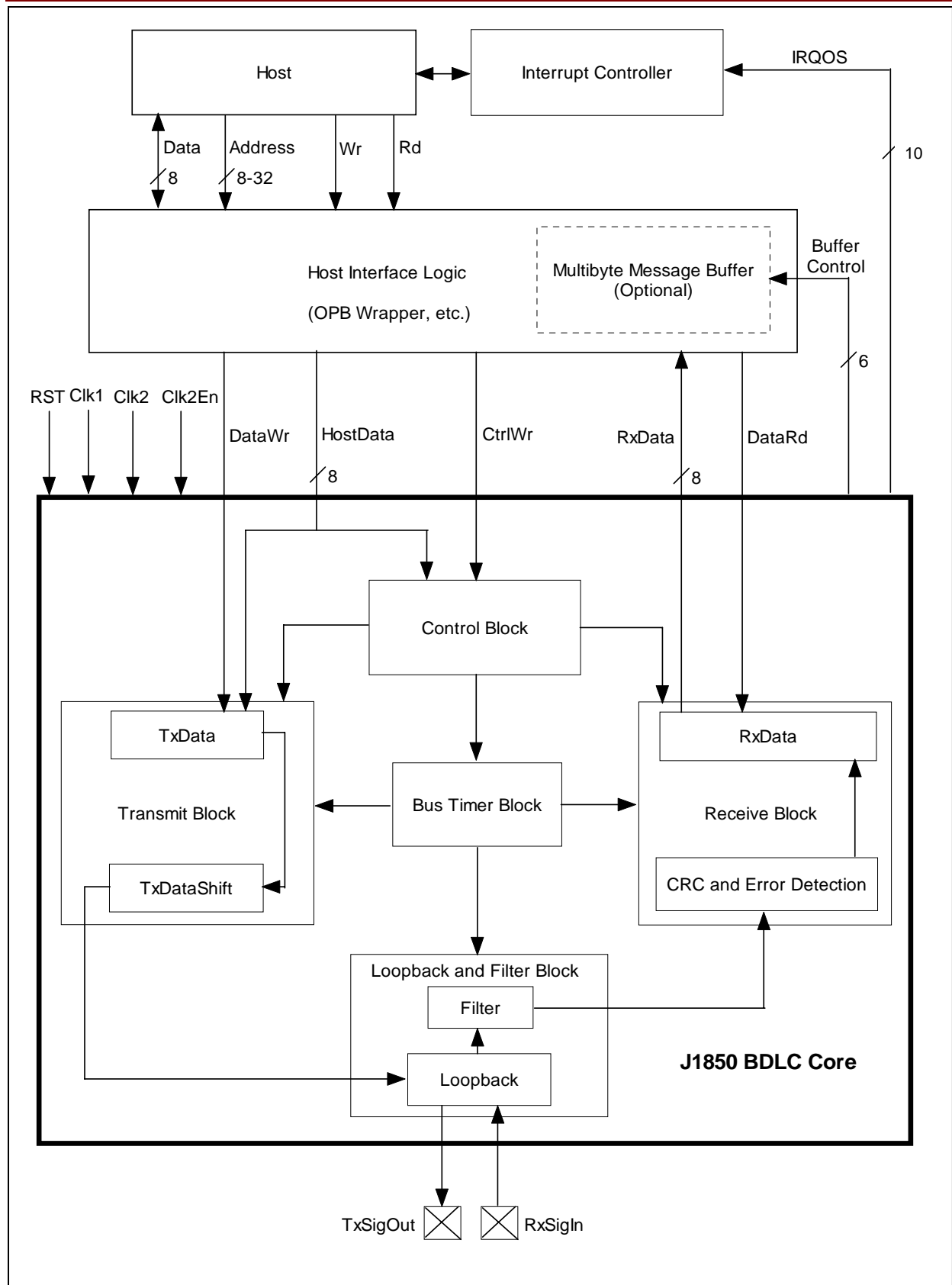
### Features

- SAE J1850 class B data communications network interface compatible for low speed (<= 125 Kbps) serial data communications in automotive applications
- Byte-by-byte software message interface
  - Supports wrapper logic for message buffering
  - Supports direct interface with internal soft-core processor or external processor
- 41.6 Kbps Pulse Width Modulated (PWM) bit format (any bit rate can be parameterized)
- Digital noise filter
- Supports block mode for receive and transmit
- Hardware Cyclical Redundancy Check (CRC) generation and checking
- Parameterized symbol timing
- Supports In-Frame-Response (IFR) types 0, 1, 2 and 3
- Ten interrupt flag one-shots for timing errors, CRC error, lost arbitration, transmit data register empty, receive data register full, IFR received, EOF received, Break, CRC complete, transmit complete.

CORE Facts	
<b>Provided with Core</b>	
Documentation	Core datasheet
Design File Formats	netlist, Verilog
Constraints Files	N/A
Verification	Test Bench
Instantiation templates	Verilog
Reference designs & application notes	Upon customer request
Additional Items	
<b>Simulation Tool Used</b>	
ModelTech's Modelsim	
<b>Support</b>	
Support provided by Drivven, Inc.	

### Table 1: Example Implementation Statistics

Contact Drivven for implementation statistics.



**Figure 1: J1850 BDLC PWM Core Block Diagram**

## Applications

- Serial data communications for Society of Automotive Engineers (SAE) automotive On Board Diagnostics II (OBDII) support
- Serial data communications for automotive body electronics
- Serial data communications for automotive service tools

## General Description

The Drivven J1850 BDLC PWM Core is a serial communications module which allows transmitting and receiving data on a SAE J1850 network. Transmitted and received messages are handled by software one byte at a time, while the J1850 IP core handles all network access, message framing, arbitration and error detection. The J1850 core provides the necessary interface signals for direct interface to typical address/data bus decoder logic and interrupt controller. It also provides additional signals useful for creating a multi-byte message buffer wrapper. The core provides a transmit output signal and a receive input signal which can be routed off-chip. These off-chip signals can be configured for TTL or CMOS specifications as available by the particular FPGA device being used, and must be interfaced to an external transceiver device which is compatible with a SAE J1850 network. It is assumed that the user of this core is familiar with the SAE J1850 protocol as described in the document "SAE Standard J1850 Class B Data Communications Network Interface."

## Functional Description

The J1850 BDLC PWM Core is comprised of five main functional blocks. Each block is described below.

### Loop-Back and Filter Block

It is possible to configure the J1850 core for a loop-back mode which disconnects the external receiver input signal from the filter, and connects the internal transmit signal directly to the filter input. This mode also drives a logic zero on the core's external transmit signal. This mode allows the user to test the functionality of the core by transmitting and receiving messages without affecting the external bus.

The filter is an integrating digital filter which samples the external receiver input at the rate of Clk2. The filter contains a parameterized up-down counter clocked at the rate of Clk2. When the filter input is high, the counter counts up. When the filter input is low, the counter counts down. When the counter reaches zero, the filter outputs a logic zero to the receiver block. When the counter reaches all ones, the filter outputs a logic one to the receiver block. This filter rejects high or low receiver signal glitches less than the all-ones value of the counter multiplied by the period of Clk2. The filter counter width is parameterized by FILTER\_WIDTH.

### Bus Timer Block

The bus timer block measures the time since the last rising or falling edge of the filter output. The bus timer is used to recognize the various J1850 PWM symbols, as well as detect any symbol timing errors.

### Control Block

The control block interfaces with the external data bus and address bus decoder to accept message bytes and control bytes. There are two 8-bit registers which can be written, TxData and Ctrl, to control the behavior of the BDLC. Data at HostData is clocked into the internal TxData buffer register on the rising edge of Clk1 while DataWr is high. DataWr must not be asserted longer than three periods of Clk1, otherwise additional data writes may occur, causing additional data bytes to be transmitted. In general, to begin transmitting a message on the J1850 bus, the host only needs to write a message byte to TxData.

The BDLC then monitors the bus for an opportunity to transfer TxData to TxDataShift for serial transmission. Details of message transmission are covered within the Transmitter Block description.

Data at HostData is clocked into the internal Ctrl register on the rising edge of Clk1 while CtrlWr is high. CtrlWr must not be asserted longer than three periods of Clk1, otherwise additional control writes may occur. Below is a description of each bit within the Ctrl register.

**Table 2: J1850 BDLC PWM Control Register**

MSB 7	6	5	4	3	2	1	LSB 0
LoopBackCtrl	-	IMSGCtrl	BreakCtrl	TMIFR1Ctrl	TMIFR0Ctrl	TSIFRCtrl	TEODCtrl

**TEODCtrl - Transmit EOD Symbol (bit 0) :** This bit must be set in order to properly terminate a transmitted message. TEODCtrl can be set at any time after the first message byte is loaded into TxData. Message bytes will continue to be transmitted as long as they are written to TxData, even while TEODCtrl is set. When there is no longer data available in TxData to be transmitted and TEODCtrl is set, a CRC byte will be transmitted, followed by an EOD symbol, thus properly terminating the message. TEODCtrl must be set before the last TxDataShift bit is transmitted, otherwise an underflow condition will be encountered, which will cause the BDLC to flush TxDataShift and transmit two one bits on the bus. Two one bits will not cause any other node to lose arbitration, but will cause an error on the bus if the local node is the only transmitting node. Therefore, other nodes will properly discard the message. The TEODCtrl bit is also used to terminate an IFR. TSIFRCtrl, TMIFR0 and TMIFR1 determine whether a CRC byte is appended to the IFR. TEODCtrl is automatically cleared upon transmitting the last bit of the CRC byte, receiving an EOD symbol, loss of arbitration, or a bus timing error.

**TSIFRCtrl, TMIFR0Ctrl, TMIFR1Ctrl – Transmit IFR Control (bits 3:1) :** These three bits control the type of IFR that is sent in response to a main message.

Setting TSIFRCtrl will cause the BDLC to transmit a single IFR byte followed by no CRC byte. Data must be written to TxData and TSIFRCtrl must be set either before an EOF symbol is received or before another node transmits a normalization bit, otherwise the IFR byte will not be transmitted. If arbitration is lost, the BDLC will continue transmission attempts until the IFR byte is successfully transmitted. TEODCtrl may be set to discontinue transmission attempts. TSIFRCtrl will be automatically cleared upon receiving an EOD symbol or bus timing error.

Setting TMIFR1Ctrl causes the BDLC to transmit multiple IFR bytes followed by a CRC byte. Data must be written to TxData and TMIFR1Ctrl must be set either before an EOF symbol is received or before another node transmits a normalization bit, otherwise no IFR bytes will be transmitted. If arbitration is lost, the BDLC will discontinue transmission. TEODCtrl is used to properly terminate a multiple-byte IFR with an EOD symbol, similar to terminating a main message. TMIFR1Ctrl is automatically cleared upon transmitting the last bit of the CRC byte, receiving an EOD symbol, loss of arbitration, underrun or a bus timing error.

Setting TMIFR0Ctrl causes the BDLC to transmit multiple IFR bytes followed by no CRC byte. Data must be written to TxData and TMIFR0Ctrl must be set either before an EOF symbol is received or before another node transmits a normalization bit, otherwise no IFR bytes will be transmitted. If arbitration is lost, the BDLC will discontinue transmission. TEODCtrl is used to properly terminate a multiple-byte IFR with an EOD symbol, similar to terminating a main message. TMIFR0Ctrl is automatically cleared upon transmitting the last bit of the CRC byte, receiving an EOD symbol, loss of arbitration, underrun or a bus timing error.

In order to carry out any of the three IFR functions, only one of the three IFR control bits may be set at a time. Otherwise an IFR will not be transmitted.

**TSIFRCtrl, TMIFR0Ctrl, TMIFR1Ctrl – Clear TxDataShift Register (bits 3:1) :** If all three IFR control bits are set to 1 simultaneously within the same control register write cycle, then the TxDataShift register will be invalidated and cleared. This will cause the TxDataShift register to load itself with new data written to TxData or wait for new data written to TxData. The state of the IFR control bits will be cleared. TxDataShift loads new data from TxData. Then the host is free to set an individual IFR control bit according to the desired function. This feature is useful for the case of scheduling a main message byte for transmission and then determining that the host must respond to another node with an IFR. This feature allows the host to cancel a scheduled main message byte and reschedule an IFR byte. Care should be taken not to set all three IFR control bits while actually transmitting a message. Otherwise, the message will be interrupted and a timing error condition on the bus will result.

**BreakCtrl – Transmit Break Symbol (bit 4) :** When this bit is set, a break symbol is transmitted on the bus. All nodes on the bus should recognize the break symbol and stop transmitting immediately. The BDLC will treat a received Break symbol internally as a bus error, and as a result wait for an IFS symbol before transmitting a new message. However, receipt of a Break symbol will generate a separate interrupt from the timing error interrupt. TxBreakCtrl is automatically cleared after a full break symbol is transmitted.

**IMSGCtrl – Ignore Message (bit 5) :** When this bit is set, the one-shot interrupt flags RxDRF\_IRQOS, RxIFR\_IRQOS and RxEOF\_IRQOS will not be active until the next SOF symbol is received. This allows the local host to ignore its own transmitted messages. The user should be aware that if arbitration is lost, the above mentioned interrupt flags will still not be asserted for the remaining portion of the message, unless IMSGCtrl is explicitly cleared by the host. IMSGCtrl is automatically cleared upon receiving a SOF symbol.

**LoopBackCtrl – Loop Back mode (bit 7) :** This bit may be set by the host to disconnect the BDLC transmit and receive signals from external logic and connect the internal transmit signal directly to the input of the receiver filter. This allows the host to send a message to itself for testing the functionality of the BDLC. The host must clear the LoopBackCtrl bit.

## Transmitter Block

To begin transmitting a message, the host must simply write a byte to the TxData register. A write to TxData causes the TxDRE flag to be cleared. The transmitter block will monitor the J1850 bus for either an idle bus, a new IFS symbol, or a new EOF symbol followed by a rising edge on the J1850 bus. When any of these conditions are detected, the transmitter will transfer the contents of TxData to TxDataShift, set the TxDRE flag, and begin transmitting an SOF symbol. The SOF symbol will be followed by each bit within TxDataShift. Data is transmitted on the J1850 bus from the most significant bit to the least significant bit. The host can use the TxDRE\_IRQOS signal to trigger an interrupt or monitor the TxDRE flag as an indication that a new message byte may be written to the BDLC. The contents of TxData may be overwritten at any time and cannot be read.

To properly terminate a main message, the host must set TEODCtrl and stop writing message bytes to TxData. TEODCtrl must be set before the last TxDataShift bit is transmitted, otherwise an underrun condition will occur. An underrun condition causes the transmitter to transmit two one-bits, which will cause a bus error if the local host is the only transmitting node. This will prevent an invalid message from being received by other nodes.

While any data are being transmitted, the TxTransmitting flag will be set. While transmitting a CRC byte, the TxTransmittingCRC flag will be set.

To transmit an IFR after the local host has transmitted a main message, the first IFR byte may be written any time after receiving the CRC byte but before an EOF symbol or an IFR data bit is received from another node. The host may use the RxDRF flag or RxDRF\_IRQOS interrupt for determining when the CRC byte has been received. To transmit an IFR in response to a main message received by another node, the host may write the first IFR byte any time after receiving the first main message byte but before an EOF symbol or an IFR data bit is received by another node. To transmit a single IFR byte, the host must accompany the IFR byte with setting the TSIFRCtrl bit. If arbitration is lost while transmitting the single IFR byte, the transmitter will continue attempting transmission until successful or until the host sets the TEODCtrl bit.

To transmit a multiple byte IFR, the host writes message bytes to the BDLC in the same manner as transmitting a main message except that the TMIFR0Ctrl bit or the TMIFR1Ctrl bit must be set, depending on the requirement of a trailing CRC byte. The TEODCtrl bit is used to properly terminate a multiple byte IFR similar to that of a main message.

Message transmission will be prematurely terminated and TxData will be flushed upon a loss of arbitration, bus timing error or underrun.

While transmitting a main message, setting TEODCtrl will cause the transmitter to send an internally calculated CRC byte followed by an EOD symbol. TEODCtrl will be cleared upon transmitting the last bit of the CRC byte. While transmitting a single IFR byte, setting TEODCtrl will cause the transmitter to attempt transmission only once, subject to arbitration. Otherwise the transmitter will keep attempting until successful transmission. A single IFR byte may be followed by additional single IFR bytes from other nodes or an EOD symbol. While transmitting a multiple byte IFR, setting TEODCtrl will cause the transmitter to transmit a CRC byte (if using TMIFR0Ctrl) followed by an EOD symbol. If arbitration is lost, the transmitter will transmit two one bits and stop. After receiving an EOD symbol TEODCtrl and the IFR control bits will be cleared.

## **Receiver Block**

The receiver block shifts data bit symbols from the J1850 bus into the internal RxDataShift register. When eight bits of data have been shifted in, the contents of RxDataShift are transferred to RxData and the RxDRF flag is set. RxDRF is cleared when the host reads RxData. The host must ensure that RxData is read before the next message byte is received, otherwise RxData will be overwritten and its previous contents will be lost. The host must assert DataRd while reading RxData. The host may poll the RxDRF flag or utilize the one-shot RxDRF\_IRQOS signal for generating an interrupt. If the received byte is an IFR byte, another one-shot interrupt signal called RxIFR\_IRQOS will be generated in conjunction with RxDRF\_IRQOS.

The receiver block updates its CRC calculation with each received data bit. The final CRC result is tested against 0xC4 upon receiving a main message EOD symbol and upon receiving an EOD symbol trailing an IFR with a CRC byte. In these cases, if the calculated CRC is not 0xC4 then the RxCRCErrror\_IRQOS one-shot will be generated.

The term "Block Mode," as applied to J1850 networking, means that the specified SAE J1850 limit of 12 bytes per message frame is not adhered to. The BDLC handles message data on a byte-by-byte basis and does not count the bytes as they arrive. Therefore no special configuration is required to support Block Mode. The host shall simply continue to respond to message bytes as they arrive.

---

## Interrupt Signals

Eight one-shot interrupt pulses are generated for aiding the host with managing message transmitting, receiving and error conditions. These signals are active high pulses which are high for one period of Clk2, enabled by Clk2En. They are intended to be connected to an interrupt controller block which handles edge-triggered interrupt sources. Each interrupt signal is described below:

**RxTimingError\_IRQOS** : This one-shot interrupt signal is generated upon the following error conditions:

1. SOF symbol too short
2. Short data bit or normalization bit too short
3. Long data bit or normalization bit too long

**CRCErrror\_IRQOS** : This one-shot interrupt signal is generated upon detecting an EOD symbol and the calculated CRC byte is not equal to 0xC4. This error is not generated if the IFR normalization bit indicated that no CRC was present.

**TxLostArb\_IRQOS** : This one-shot interrupt signal is generated upon losing arbitration to another message being transmitted by another node. When arbitration is lost on a byte boundary, the transmitter will transmit two logical one bits (except for single byte IFRs) to prevent possible receiving errors for other nodes. Please refer to the SAE J1850 documentation for details about message arbitration.

**TxDRE\_IRQOS** : This one-shot interrupt signal is generated upon the TxDRE flag going high.

**RxDRE\_IRQOS** : This one-shot interrupt signal is generated upon the RxDRF flag going high.

**RxIFR\_IRQOS** : This one-shot interrupt signal is generated upon the RxDRF flag going high and the received data byte being an IFR byte.

**RxEof\_IRQOS** : This one-shot interrupt signal is generated upon receiving an EOF symbol.

**RxBreak\_IRQOS** : This one-shot interrupt signal is generated upon receiving a Break symbol.

**TxCRCComplete\_IRQOS** : This one-shot interrupt signal is generated upon completely sending the CRC byte.

**TxComplete\_IRQOS** : This one-shot interrupt signal is generated upon completely sending a main message (including CRC) or completely sending an IFR message (single or multibyte, including CRC if applicable). It is the responsibility of the host to know whether this interrupt represents complete transmission of a main message or IFR.

It may be desired by the user to have the above signals appear as level triggered interrupts or flags which can be read from a register. This may be easily implemented in a wrapper module by using the above individual one-shot signals to set the output bit of a register until the flag is recognized and cleared by the host. This feature is not implemented directly in this core in order to maintain its flexibility.

## External Multibyte Message Buffer Implementation and Related Signals

The BDLC may be operated with a variety of interface methods. However, it was primarily designed for small logic devices, interfacing with simple 8-bit processor hosts which have few other responsibilities than managing J1850 data traffic and/or gateways between additional networks. The byte-by-byte message handling interface is useful for such devices. However, message buffering strategies may also

## J1850 BDLC PWM Core

---

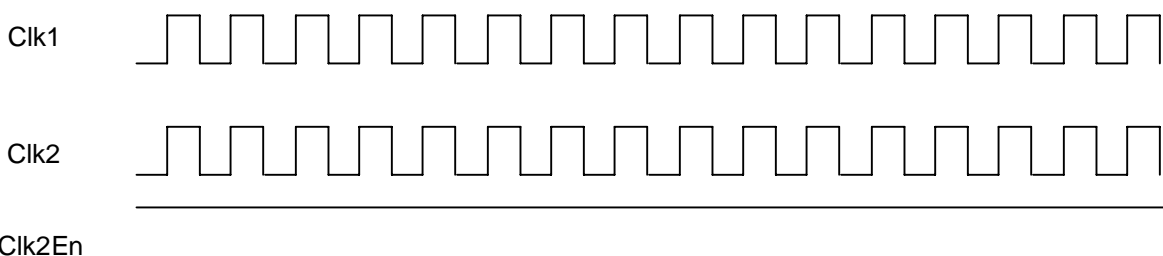
be implemented using the signals provided by the BDLC core. TxDRE, RxDRF, TxTransmitting, TxTransmittingCRC and the one-shot interrupt signals may be used for this purpose.

## Clocking and Reset

All run-time logic within the BDLC is synchronous to the rising edge of either Clk1 or Clk2. All storage elements are cleared asynchronously with an active-high RST.

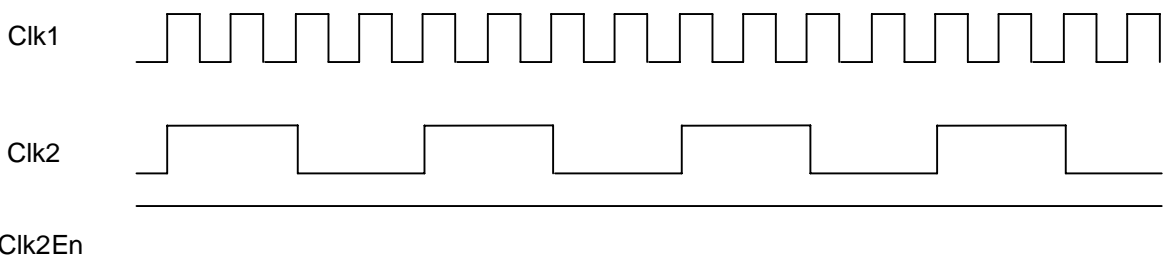
Two clock inputs, Clk1 and Clk2, and one clock enable, Clk2En, are utilized by the BDLC core. Clk1 is used to clock host data into the TxData and Ctrl registers. In most cases, Clk1 will be driven by the main system clock. All other logic for the behavior of the BDLC is clocked with the combination of Clk2 enabled by Clk2En. Clk2En shall be derived with flip-flops from Clk2 or tied to VCC. The frequency of Clk1 must be greater than or equal to Clk2, and the BDLC logic must be driven by Clk2 (enabled by Clk2En) at a minimum rate of 4 MHz. There are a variety of ways to clock the BDLC and they are described below:

**Method 1 - Clk1 = Clk2, Clk2En = 1** : This is the simplest way to clock the BDLC. The same clock which is used by the host to drive the bus interface logic (Clk1) can be used to drive the BDLC logic (Clk2). This mode requires Clk2En to be tied to VCC. Refer to Figure 2 for an example of this method.



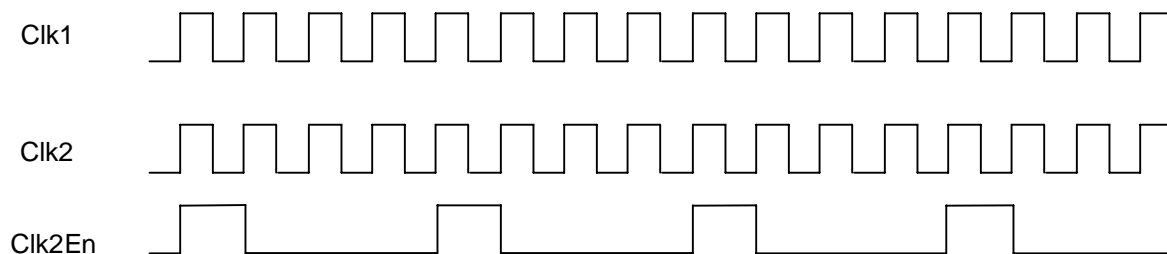
**Figure 2. Clock Method 1.**

**Method 2 - Clk1 > Clk2, Clk2En = 1** : External logic may utilize a part-specific DLL to derive a lower frequency, no-skew Clk2 signal from Clk1. The frequency of Clk1 must be an integer multiple of the frequency of Clk2. The BDLC logic can be driven at the rate of Clk2 by connecting Clk2En to VCC. Refer to Figure 3 for an example of this method. This method of clocking allows the host to operate at a higher frequency than the BDLC while keeping the area of the BDLC to a minimum.



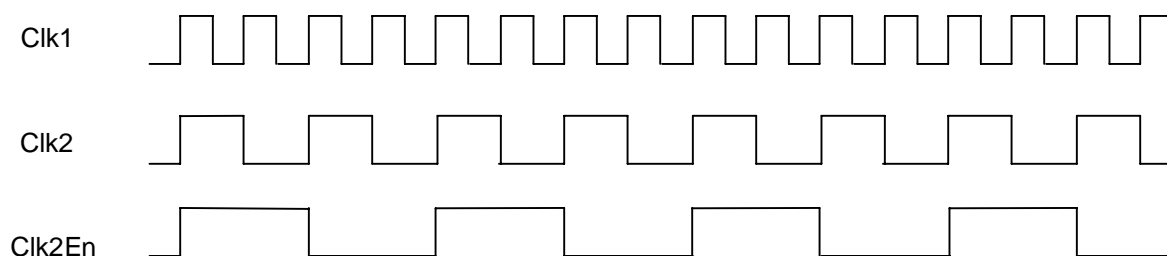
**Figure 3. Clock Method 2.**

**Method 3 - Clk1 = Clk2, Clk2En derived from Clk2 :** This method is similar to method 1, except that Clk2En is not connected to VCC, but is derived from Clk2 using flip-flops to generate a clock-enable pulse train which is high for one period of Clk2. Refer to Figure 4 for an example of Clk2En derived from Clk2. This method allows the designer to use a single system clock while keeping the effective clock rate of the BDLC to a minimum, reducing the area of the BDLC.



**Figure 4. Clock Method 3.**

**Method 4 - Clk1 > Clk2, Clk2En derived from Clk2 :** This method is similar to method 2, except that Clk2En is not connected to VCC, but is derived from Clk2 using flip-flops to generate a clock-enable pulse train which is high for one period of Clk2. Refer to Figure 5 for an example of Clk2En derived from Clk2. This method may be necessary if the DLL is not able to generate an optimum frequency for Clk2. Therefore, Clk2En may be used to further reduce the effective clock rate of the BDLC.



**Figure 5. Clock Method 4.**

## Parameters

Most of the parameters which configure the core specify the length of each J1850 PWM bus symbol. Each parameter used for receiving symbols specifies a minimum or maximum time length for the associated symbol and is represented in Clk2 ticks, enabled by Clk2En. The BDLC must operate at a minimum frequency of 4 MHz in order to properly receive and transmit each symbol. This results in a minimum value for each symbol timing parameter.

Each parameter prefixed with RX must be calculated by multiplying the BDLC operating frequency by the parameters' associated timing value, in seconds. For example, if the BDLC operating frequency is 4 MHz, RX\_MIN can be calculated according to:

$$\text{RX\_PARAMETER} = \text{BDLC Freq} * \text{Symbol Time.}$$

$$\text{RX\_ONE\_AMIN} = 4,000,000 * 0.000006 = 24 \text{ ticks.}$$

Each parameter used for transmitting symbols specifies a target time length for the associated symbol and is represented in Clk2 ticks, enabled by Clk2En. For example, if the BDLC operating frequency is 4 MHz, TX\_TARGET\_SHORT can be calculated according to:

$$\text{TX\_PARAMETER} = (\text{BDLC Freq} * (\text{Symbol Time} - \text{Round Trip Delay})) - \text{Filter Ticks}.$$

$$\text{TX\_ONE\_ATARGET} = (4,000,000 * (0.000008 - 0.000000)) - 8 = 24 \text{ ticks}.$$

**Round Trip Delay** is the total time delay required for a signal state to propagate through the transmitter and receiver of an external analog J1850 transceiver. Refer to the transceiver device datasheet for this delay value.

**Filter Ticks** is the full cycle count of the receiver filter. For a 3-bit filter, the cycle count is 8.

Each parameter is documented below in Table 3. The core's default values for symbol timing parameters assumes a 4 MHz clock at Clk2, Clk2En tied to VCC, a round trip delay of 0 usec and a 3-bit receiver filter (8 ticks).

**Table 3: J1850 BDLC PWM Parameters**

Parameter Name	Description	Symbol Time (usec)	Default Parameter Value
RX_PASSIVE_MIN	Minimum allowed passive time	4	16
RX_ONE_AMIN	Minimum active phase "1"	6	24
RX_ZERO_AMIN	Minimum active phase "0"	14	56
RX_ZERO_AMAX	Maximum active phase "0"	19	76
RX_BIT_LMIN	Minimum bit length	22	88
RX_BIT_LMAX	Maximum bit length	27	108
RX_EOD_LMIN	Minimum EOD length	46	184
RX_EOF_LMIN	Minimum EOF length	70	280
RX_SOF_AMIN	Minimum active phase SOF	30	120
RX_BREAK_AMIN	Minimum active break	38	152
RX_SOF_LMIN	Minimum SOF length	45	180
RX_SOF_LMAX	Maximum SOF length	52	208
RX_IFS_LMIN	Nominal IFS length	96	384
TX_ONE_ATARGET	Target active phase "1"	8	24
TX_ZERO_ATARGET	Target active phase "0"	16	56
TX_SOF_ATARGET	Target active phase SOF	32	120
TX_BREAK_ATARGET	Target active break	40	152
TX_BIT_LTARGET	Target bit length	24	88
TX_SOFEOD_LTARGET	Target SOF/EOD length	48	186
BUS_TIME_WIDTH <sup>(1)</sup>	Bus timer width	N/A	9
FILTER_WIDTH	Receiver filter timer width	N/A	3

1. BUS\_TIME\_WIDTH should accommodate RX\_IFS\_LMIN. BUS\_TIME\_WIDTH must also be large enough so that RX\_IFS\_LMIN does not set the 4 most significant bits of BusTime. This is because we compare the 4 most significant bits of BusTime with 4'b1111. If these bits are set then we stop incrementing BusTime to keep it from ever rolloing over. For example, if RX\_IFS\_LMIN were a value of 0xF80, then at first glance, BUS\_TIME\_WIDTH should be 12. However, BusTime would never achieve the value 0xF80 because the 4 most significant bits would be set before 0xF80 would be reached. This scenario would require that BUS\_TIME\_WIDTH be set to 13. BUS\_TIME\_WIDTH must also be >= 4.

## Core I/O Signals

The core signal I/O have not been fixed to specific device pins to provide flexibility for interfacing with user logic. All signals are active-high unless otherwise noted. Descriptions of all signal I/O are provided in Table 4.

**Table 4: Core I/O Signals.**

Signal	Signal Direction	Description
Clk1	Input	Clock input for host register interface
Clk2	Input	Clock input for BDLC logic
Clk2En	Input	Enabling signal for Clk2
RST	Input	Active high, asynchronous reset for all BDLC registered signals
RxSigIn	Input	BDLC receiver input
TxSigOut	Output	BDLC transmitter output
HostData[7:0]	Input	Host data input to Ctrl and TxData
RxData[7:0]	Output	Message data output to host
BusUnknown	Output	Indicates that the status of the external J1850 bus is unknown. This signal will be asserted upon a reset or a bus timing error until an EOF symbol is detected.
IdleStatus	Output	Indicates that an IFS symbol has been detected and no activity is present on the J1850 bus. This signal will be asserted upon detection of an IFS symbol until a rising or falling edge is detected on the J1850 bus.
RxDRF	Output	Receive Data Register Full. RxDRF will be asserted when a complete message byte has been received by the BDLC. It will remain asserted until the host reads the RxData register, indicated by the assertion of RxDataRd.
TxDRE	Output	Transmit Data Register Empty. TxDRE will be asserted when the contents of TxData are transferred to TxDataShift. TxDRE will remain asserted until TxData is written by the host, indicated by DataWr.
TxTransmitting	Output	Indicates that the transmit block is currently transmitting data.
TxTransmittingCRC	Output	Indicates that the transmit block is currently transmitting a CRC byte.
DataRd	Input	Must be asserted by the host while reading the value of RxData[7:0]. RxData[7:0] is always driven with the most recent received data byte. However, DataRd provides a mechanism for the BDLC to properly control RxDRF.
DataWr	Input	Must be asserted by the host to enable latching of data present at HostData into TxData upon the rising edge of Clk1. Must not be asserted longer than three periods of Clk1.
CtrlWr	Input	Must be asserted by the host to enable latching of data present at HostData into Ctrl upon the rising edge of Clk1. Must not be asserted longer than three periods of Clk1.
RxTimingError_IRQOS	Output	Clk2 one-shot generated upon a bus timing error.
CRCErrror_IRQOS	Output	Clk2 one-shot generated upon a CRC error.
TxLostArb_IRQOS	Output	Clk2 one-shot generated upon a loss of bus arbitration while transmitting.
TxDRE_IRQOS	Output	Clk2 one-shot generated upon TxDRE being asserted.
RxDRF_IRQOS	Output	Clk2 one-shot generated upon RxDRF being asserted.
RxIFR_IRQOS	Output	Clk2 one-shot generated upon an IFR byte being received.
RxEOF_IRQOS	Output	Clk2 one-shot generated upon an EOF symbol being received.
RxBreak_IRQOS	Output	Clk2 one-shot generated upon a Break symbol being received.
TxCRCComplete_IRQOS	Output	Clk2 one-shot generated upon completion of CRC byte transmission.
TxComplete_IRQOS	Output	Clk2 one-shot generated upon completion of main message or IFR.

### **Core Modifications**

Core modifications may be made by Drivven, Inc., for additional charge.

### **Verification Methods**

The behavior of this core was verified using a Verilog test fixture created for the ModelSim environment.

### **Recommended Design Experience**

The user of this core is expected to have a thorough understanding of programmable logic design and the SAE J1850 communications protocol.

### **Ordering Information**

This product is available directly from Drivven, Inc. Please contact Drivven, Inc. for pricing and additional information about this product. Contact information for them is on the front page of this datasheet.