


# BYTEFLIGHT

## Block Guide

### V01.17

Original Release Date: 29 Dec 2000  
Revised: 11 Mar 2003

**Motorola, Inc.**

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

©Motorola, Inc., 2001

# Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
00.01	29th Dec,2000	29th Dec,2000		Initial Version.
V00.02	5th Mar,2001	5th Mar,2001		<p>Summary Of Changes</p> <p>Changed block name to Byteflight from SIBUS.</p> <p>Changed all references in the specification from SI-Bus to Byteflight.</p> <p>Modification in Features Section.</p> <p>Added names of two temporary Receive Buffers, RXFG and RXBG.</p> <p>Added Rejection filter.</p> <p>Corrected some typo errors.</p> <p>Modified Byteflight Block Diagram (Fig 1-2).</p> <p>Modified Byteflight Top-Level Block Diagram as per latest MSRSV2.0.</p> <p>Modified External Pins description.</p> <p>Modified Table of signal properties (Table 2-1).</p> <p>Added latest module specific signals description.</p> <p>Added registers IDX, SITEST, BUFTEST, BUFLOCK in the memory map and also added their appropriate description.</p> <p>Corrected Reset value of register bits TIVEC and FIDMR.</p> <p>Modified Byteflight Timing parameters.</p> <p>Changed all ms timing parameters to us.</p> <p>Added t_latest_rx timing parameter.</p> <p>Added few more points in Initialization Sequence to make it more clear.</p> <p>Modified the Interrupt definition.</p>
V00.03	19th March, 2001	19th March, 2001		<p>Summary Of Changes</p> <p>Capitalized the Block Name.</p> <p>Incorporated the feedback sent on Mar 14, 2001.</p> <p>Added "NOTE" for the register BUFTEST and BUFLOCK.</p> <p>Corrected some typo errors.</p> <p>Corrected the part number in Fig 1-1 from M68HC12 to MCS912DTB128.</p> <p>Corrected the pin names for TX and for RX in Fig 1-1.</p> <p>Added general NOTE in Register Descriptions.</p>
V01.04	23rd March, 2001	23rd March, 2001		<p>Summary Of Changes</p> <p>Modified the new BUG version number from 0.4 to 1.4.</p> <p>Changed all references of ELMOS 100.34 to Infineon SPF BFT003.</p> <p>Corrected some typo errors.</p> <p>Changed Initialization Mode value for the registers to IP Reset Mode to avoid confusion with module Initialization mode (INITRQ = INTAK = 1) and added the description in Section 1, Modes Of Operation.</p>

Version Number	Revision Date	Effective Date	Author	Description of Changes
V01.05	30th March, 2001	30th March, 2001		<p>Summary Of Changes</p> <p>Corrected some typo errors.</p> <p>Added TABLE in the description of BFEN bit and BUFLOCK register bits for better understanding.</p> <p>Deleted non required statement from the description of SYNLFIF flag bit in GISR register.</p> <p>Corrected IP Reset mode value of SITEST register and added a NOTE.</p> <p>Added NOTE in the description of IFLG bit.</p> <p>Modified NOTE description of BUFLOCK register.</p>
V01.06	24th April, 2001	24th April, 2001		<p>Summary Of Changes</p> <p>Corrected typo error in the NOTE for IFLG bit description (Section 3.3.23).</p> <p>Removed redundant statement of "BFEN = 0" in the NOTE of IDX register.</p> <p>Added general NOTE in Register Descriptions.</p> <p>Removed the invalid values for FSIZR register in the Table 3-2.</p> <p>Corrected some typo errors.</p> <p>Corrected the pin names in the Byteflight Block Diagram Figure 1-2.</p> <p>Corrected part number in the Section 1.1 Overview.</p> <p>Updated registers to have BF as prefix.</p> <p>Added AND gate with an enable signal of BFEN in the Figure 3-20 and also corrected port pin number for ROK and SYNC signal.</p> <p>Added definition for t_latest_tx, t_start_seq_tx, t_start_seq_rx_max and t_start_seq_rx_min in Table 4-1.</p> <p>Added description of Abbreviation used in Table 6-1.</p> <p>Modified the statement in Section 1.3.2 which states that buffer lock/unlock can't takes place during Sleep mode.</p> <p>Corrected the IP Reset value of LOCKIF.</p>
V01.07	11th June, 2001	11th June, 2001		<p>Summary Of Changes</p> <p>Corrected the Access type of BUFLOCK register in Module Memory Map.</p> <p>Added condition of BFEN bit in Table 3-1 NOTES 1 and 2, in the NOTE for BFIDX register, in the LOCKIE bit description.</p> <p>Refined definition for SLPK and SLPRQ bits in BFMCR register, WAKEIF bit in BFGISR register and CFG bit in BFBUFCTL0-BFBUFCTL15 registers.</p> <p>Refined NOTE for BFFIZR register.</p> <p>Corrected some cosmetic defects.</p> <p>Corrected register name in the NOTE of register BFFIDAC BFFIDMR, BFFIDRJ, BFFIDRMR, in Section 4.2.1 and in Table 4-1.</p> <p>Corrected access type of the register BFSITEST.</p> <p>Added Reset clearing statement in the description of registers BFSITEST and BFBUFTEST.</p> <p>Added NOTE in the description for Transmit, receive and receive FIFO message buffers.</p> <p>Corrected pin names in Fig 4-6.</p>
V01.08	16th July, 2001	16th July, 2001		Incorporated the feedback sent on 21st June, 2001.
V01.09	23rd July, 2001	23rd July, 2001		Made SRSV2.0 compliant and completed internal review on the BUG.
V01.10	27th July, 2001	27th July, 2001		Incorporated the feedback sent by on 24th July, 2001.

<b>Version Number</b>	<b>Revision Date</b>	<b>Effective Date</b>	<b>Author</b>	<b>Description of Changes</b>
V01.11	2nd August, 2001	2nd August, 2001		Incorporated the feedback sent by on 30th July, 2001.
V01.12	26th Sept, 2001	26th Sept, 2001		Removed inconsistencies reported by BMW on 25th Sept, 2001.
V01.13	3rd Dec, 2001	3rd Dec, 2001		Incorporated the feedback sent on 12th November, 2001.
V01.14	08 MAR 2002	08 MAR 2002		Document format updates.
V01.15	23 SEP 2002	23 SEP 2002		Non-customer information update only.
V01.16	29 JAN 2003	29 JAN 2003		Non-customer information updated. Minor formal updates. Corrected memory map. Added register description for range \$ _20-\$ _4F. Removed reset value in BFMVR register description. Renamed 'test' mode to Special mode in BFEN bit description. Removed CPU freeze mode description.
V01.17	11 FEB 2003	11 FEB 2003		Changed document format to SRSv3. Reworked initialization/application information and updated application procedures in Functional Overview accordingly. Added BFEN = 1 requirement to Functional Overview. Removed note from IFLG bit description. Replaced expression 'MCU' with 'CPU' in several places. Replaced expression 'port' with 'pin' in several places. Various minor corrections.

# Table of Contents

## Section 1 Introduction

1.1	Overview . . . . .	13
1.2	Features . . . . .	13
1.3	Modes of Operation . . . . .	14
1.3.1	CPU Wait Mode . . . . .	14
1.3.2	CPU Stop Mode . . . . .	14
1.4	Block Diagram . . . . .	14

## Section 2 External Signal Description

2.1	Overview . . . . .	16
2.2	Detailed Signal Description . . . . .	16
2.2.1	RX_BF – Byteflight Input Data Pin . . . . .	16
2.2.2	TX_BF – Byteflight Output Data Pin . . . . .	16
2.2.3	BF_PSYN – Byteflight Correct Sync Pulse Reception/Transmission Pin . . . . .	16
2.2.4	BF_PROK – Byteflight Correct Message/Sync Reception Pulse Pin . . . . .	16
2.2.5	BF_PERR – Byteflight Illegal Pulse Error Pin . . . . .	16
2.2.6	BF_PSLM – Byteflight Slot Mismatch Pin . . . . .	16

## Section 3 Memory Map/Register Definition

3.1	Overview . . . . .	17
3.2	Module Memory Map . . . . .	17
3.2.1	Programmer's Model . . . . .	21
3.3	Register Descriptions . . . . .	22
3.3.1	Module Configuration Register (BFMCR) . . . . .	22
3.3.2	FIFO Size Register (BFFSIZR) . . . . .	24
3.3.3	Time Configuration Register 1 (BFTCR1) . . . . .	25
3.3.4	Time Configuration Register 2 (BFTCR2) . . . . .	26
3.3.5	Time Configuration Register 3 (BFTCR3) . . . . .	27
3.3.6	Receive Interrupt Status Register (BFRISR) . . . . .	28
3.3.7	General Interrupt Status Register (BFGISR) . . . . .	30
3.3.8	Receive Interrupt Enable Register (BFRIER) . . . . .	32
3.3.9	General Interrupt Enable Register (BFGIER) . . . . .	33
3.3.10	Receive Interrupt Source Register (BFRIVEC) . . . . .	35

3.3.11	Transmit Interrupt Source Register (BFTIVEC) . . . . .	36
3.3.12	FIFO Identifier Acceptance Register (BFFIDAC) . . . . .	37
3.3.13	FIFO Identifier Mask Register (BFFIDMR) . . . . .	38
3.3.14	Module Version Register (BFMVR) . . . . .	39
3.3.15	Byteflight Port Control Register (BFPCTLBF) . . . . .	40
3.3.16	Buffer Lock Register (BFBUFLOCK) . . . . .	42
3.3.17	FIFO Identifier Rejection Register (BFFIDRJ) . . . . .	43
3.3.18	FIFO Identifier Rejection Mask Register (BFFIDRMR) . . . . .	44
3.3.19	Transmit, Receive and Receive FIFO Buffer Registers . . . . .	45
3.3.20	Message Buffer Control Registers (BFBUFCTL0-BFBUFCTL15) . . . . .	47

## Section 4 Functional Description

4.1	Byteflight Protocol . . . . .	49
4.1.1	Byteflight Format and Timing . . . . .	49
4.1.2	Cyclic Redundancy Check (CRC) . . . . .	51
4.1.3	Byteflight Timing Parameters . . . . .	52
4.1.4	Tolerances . . . . .	53
4.1.5	Glitch filtering . . . . .	54
4.2	Functional Overview . . . . .	54
4.2.1	Receive Process . . . . .	54
4.2.2	Receive FIFO Function . . . . .	55
4.2.3	Transmit Process . . . . .	57
4.3	Synchronization Process . . . . .	58
4.3.1	Error Handling . . . . .	58
4.3.2	SYNC Pulse Detection . . . . .	59
4.4	Reset Initialization . . . . .	60
4.5	Interrupts . . . . .	61
4.5.1	General . . . . .	61
4.5.2	Description of Interrupt Operation . . . . .	61

## Section 5 Initialization/Application Information

5.1	Initialization . . . . .	63
5.2	FIFO Usage . . . . .	63

# List of Figures

Figure 1-1	Byteflight System . . . . .	14
Figure 1-2	Byteflight Block Diagram . . . . .	15
Figure 3-1	Register Map . . . . .	21
Figure 3-2	Module Configuration Register (BFMCR) . . . . .	22
Figure 3-3	Wakeup Pulses . . . . .	23
Figure 3-4	FIFO Size Register (BFFSIZR) . . . . .	24
Figure 3-5	Time Configuration Register 1 (BFTCR1) . . . . .	25
Figure 3-6	Time Configuration Register 2 (BFTCR2) . . . . .	26
Figure 3-7	Time Configuration Register 3 (BFTCR3) . . . . .	27
Figure 3-8	Receive Interrupt Status Register (BFRISR) . . . . .	28
Figure 3-9	General Interrupt Status Register (BFGISR) . . . . .	30
Figure 3-10	Receive Interrupt Enable Register (BFRIER) . . . . .	32
Figure 3-11	General Interrupt Enable Register (BFGIER) . . . . .	33
Figure 3-12	Receive Interrupt Source Register (BFRIVEC) . . . . .	35
Figure 3-13	Transmit Interrupt Source Register (BFTIVEC) . . . . .	36
Figure 3-14	FIFO Identifier Acceptance Register (BFFIDAC) . . . . .	37
Figure 3-15	FIFO Identifier Mask Register (BFFIDMR) . . . . .	38
Figure 3-16	Module Version Register (BFMVR) . . . . .	39
Figure 3-17	Byteflight Port Control Register (BFPCTLBF) . . . . .	40
Figure 3-18	SLMM, ERR, ROK, SYNC Signal Generation . . . . .	41
Figure 3-19	Buffer Lock Register (BFBUFLOCK) . . . . .	42
Figure 3-20	FIFO Identifier Rejection Register (BFFIDRJ) . . . . .	43
Figure 3-21	FIFO Identifier Rejection Mask Register (BFFIDRMR) . . . . .	44
Figure 3-22	Active Transmit, Receive and Receive FIFO Buffer Registers . . . . .	45
Figure 3-23	Buffer Control Registers (BFBUFCTL0-BFBUFCTL15) . . . . .	47
Figure 4-1	Byteflight Message Format . . . . .	49
Figure 4-2	Byteflight Complete Message Sequence . . . . .	49
Figure 4-3	Byteflight Byte Format . . . . .	50
Figure 4-4	Byteflight Message Start Sequence . . . . .	50
Figure 4-5	Byteflight Cycle Timing . . . . .	50
Figure 4-6	Pulse Time Tolerance . . . . .	54
Figure 4-7	FIFO Status (empty, not empty, overrun) – Example with 3 buffers . . . . .	56
Figure 4-8	SYNC Pulse Errors Detection . . . . .	60





# List of Tables

Table 3-1	Module Memory Map . . . . .	18
Table 3-2	FIFO Size. . . . .	24
Table 3-3	Time $t_{wx0\_tx}$ . . . . .	25
Table 3-4	Time $t_{wx0\_rx}$ . . . . .	26
Table 3-5	Time $t_{wx0\_delta}$ . . . . .	27
Table 3-6	Conditions for updating the BFEN bit. . . . .	41
Table 3-7	Setting conditions for the status bits . . . . .	42
Table 3-8	Message Buffer Organization . . . . .	46
Table 4-1	Parameters of the Byteflight System . . . . .	52
Table 4-2	Tolerances . . . . .	54
Table 4-3	Acceptance/Rejection filter mechanism – Examples . . . . .	57
Table 4-4	Error Handling . . . . .	58
Table 4-5	Interrupt Summary . . . . .	61



# Preface

## Terminology

Acronyms and Abbreviations	
CRC	Cyclic Redundancy Check
FIFO	First-In-First-Out Memory



# Section 1 Introduction

## 1.1 Overview

The Byteflight module is the specific implementation of the BMW Byteflight concept targeted for the Motorola microcontroller family. The device interfaces a microcontroller (MCU) and the optical transceiver device Infineon (*SPF BFT003*) for receiving and transmitting messages according to latest BMW Byteflight specification.

For the BMW Byteflight specification and related matters refer to the following documents:

1. BMW – Lastenheft Byteflight, Ident-Nr.: LH 8 385 743.4
2. Infineon SPF BFT003

## 1.2 Features

The Byteflight includes these distinctive features:

- Modular Architecture.
- Implementation of the BMW Byteflight protocol.
  - 1 byte identifier.
  - 1 byte data length field (4 bits length, 4 bits reserved).
  - 0 – 12 bytes data.
  - 2 bytes checksum, hardware CRC generation and checking.
  - bit rate 10 Mbps.
  - no collision on the bus.
  - non-return-to-zero (NRZ) format.
- Double buffered receive storage scheme.
- 16 Message Buffers of 0-12 bytes data length and 2 temporary receive buffers (RXFG, RXBG).
- Programmable Message Buffer Configuration (Transmit, Receive, FIFO).
- Content-related addressing.
- Receive FIFO for bus monitoring with programmable acceptance and rejection filter.
- 11 maskable interrupt sources, generating five CPU interrupt vectors.
- Programmable bus master function.
- Programmable wake-up function.
- Low power sleep mode.

## 1.3 Modes of Operation

### 1.3.1 CPU Wait Mode

The WAIT instruction places the MCU in the low-power consumption WAIT mode. Depends on the SSWAI-bit the module can be stopped or remains active during CPU wait mode. When SSWAI is cleared the module will stay synchronized to the Byteflight and can generate interrupts to the CPU if enabled. Any such interrupt will bring the MCU out of wait mode.

### 1.3.2 CPU Stop Mode

A CPU STOP instruction stops the internal oscillator and halts all internal processing. The application software has to bring the module into the module sleep mode before the CPU STOP instruction is executed if the Byteflight has to wake up the CPU. The processor can be brought out of the STOP mode only by an external interrupt, RESET or WAKEUP interrupt of the module.

## 1.4 Block Diagram

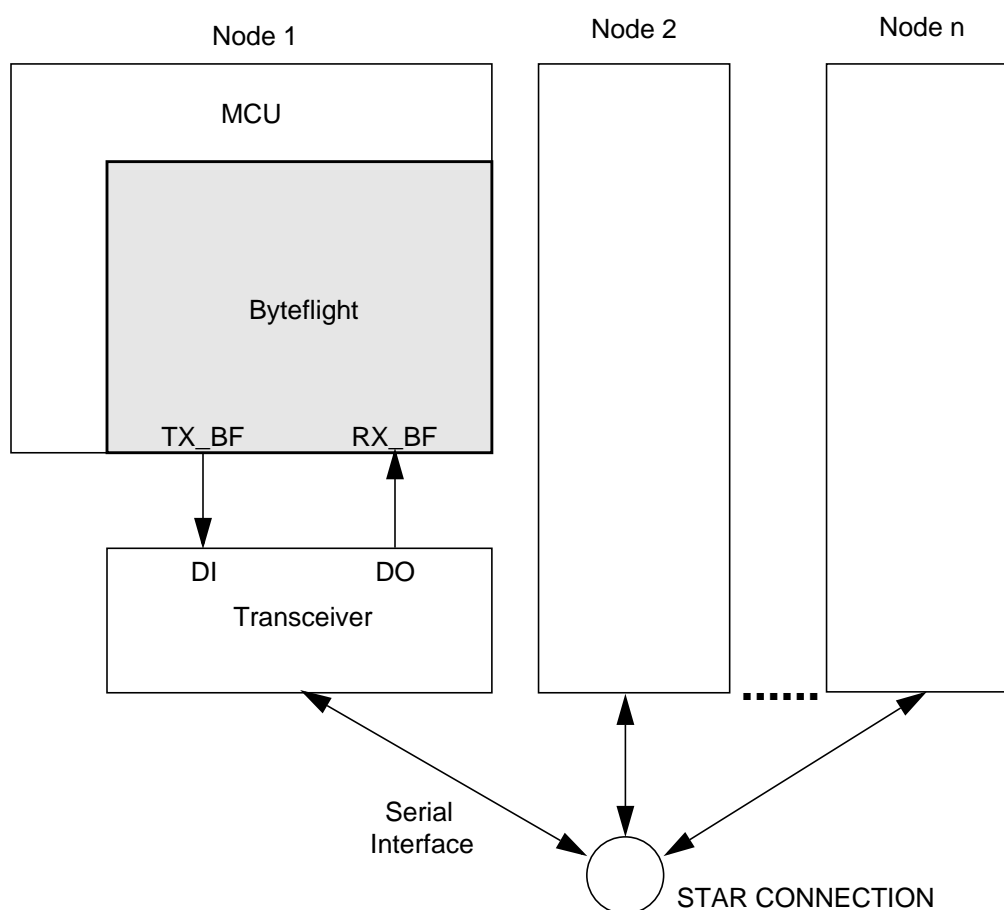


Figure 1-1 Byteflight System

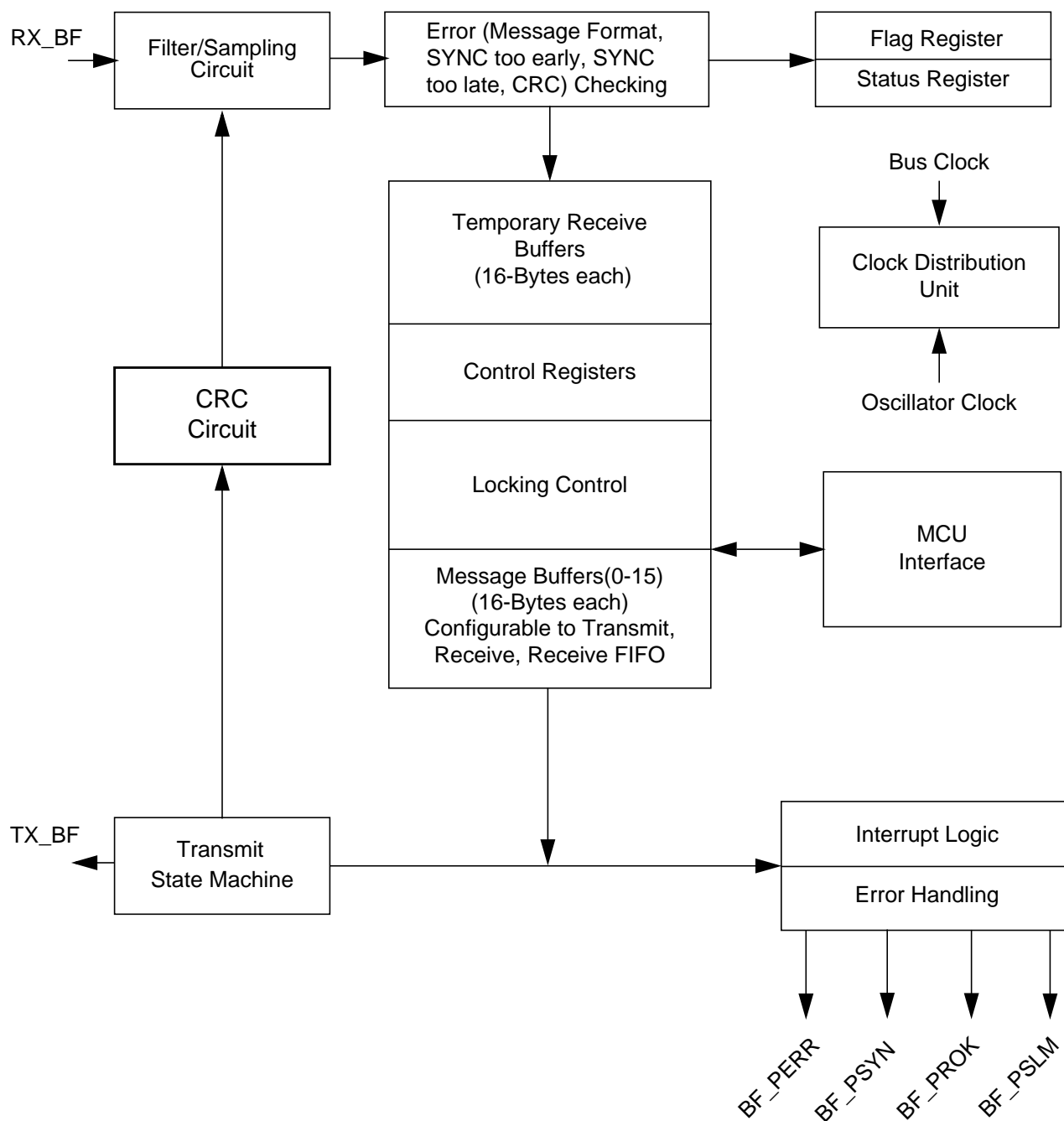


Figure 1-2 Byteflight Block Diagram

## Section 2 External Signal Description

### 2.1 Overview

The Byteflight module has a total of 6 external pins.

### 2.2 Detailed Signal Description

#### 2.2.1 RX\_BF – Byteflight Input Data Pin

This pin serves as the receiver input of the Byteflight module.

#### 2.2.2 TX\_BF – Byteflight Output Data Pin

This pin serves as the transmitter output of the Byteflight module.

#### 2.2.3 BF\_PSYN – Byteflight Correct Sync Pulse Reception/Transmission Pin

If enabled, a 50ns pulse is driven on this pin after a successful reception or transmission of a sync pulse.

#### 2.2.4 BF\_PROK – Byteflight Correct Message/Sync Reception Pulse Pin

If enabled, a 50ns pulse is driven on this pin after a successful reception of a message or sync pulse.

#### 2.2.5 BF\_PERR – Byteflight Illegal Pulse Error Pin

If enabled, a 50ns pulse is driven on this pin after a message format or illegal pulse error or after a slot mismatch.

#### 2.2.6 BF\_PSLM – Byteflight Slot Mismatch Pin

If enabled, a 50ns pulse is driven on this pin after a slot mismatch.



## Section 3 Memory Map/Register Definition

### 3.1 Overview

This section provides a detailed description of all memory and registers accessible to the end user. The special-purpose registers and register bit functions that would not normally be made available to device end users, such as factory test control registers and reserved registers are clearly identified by means of shading the appropriate portions of address maps and register diagrams. Notes explaining the reasons for restricting access to the registers and functions are also explained in the individual register descriptions.

### 3.2 Module Memory Map

The memory map for the Byteflight module is given below in **Table 3-1**. The total address for each register is the sum of the base address for the Byteflight module and the address offset for each register. The base address of the Byteflight module is determined at MCU level (*refer to the Device User Guide*). The register decode map is fixed and begins at the first address of the module address offset. The detailed register description follows the order they appear in the register map.

Reserved bits within a register will always read as “0” and writing to these bits will have no effect. Reserved and unimplemented bits are indicated by shading.

Address	Use	Access
\$_00	Module Configuration Register (BFMCR)	Read/Write <sup>2</sup>
\$_01	FIFO Size Register (BFFSIZR)	Read/Write <sup>1</sup>
\$_02	Time Configuration Register 1 (BFTCR1)	Read/Write <sup>1</sup>
\$_03	Time Configuration Register 2 (BFTCR2)	Read/Write <sup>1</sup>
\$_04	Time Configuration Register 3 (BFTCR3)	Read/Write <sup>1</sup>
\$_05	Reserved	
\$_06	Receive Interrupt Status Register (BFRISR)	Read/Write <sup>3</sup>
\$_07	General Interrupt Status Register (BFGISR)	Read/Write <sup>3</sup>
\$_08	Receive Interrupt Enable Register (BFRIER)	Read/Write
\$_09	General Interrupt Enable Register (BFGIER)	Read/Write <sup>2</sup>
\$_0A	Receive Interrupt Vector Register (BFRIVEC)	Read Only
\$_0B	Transmit Interrupt Vector Register (BFTIVEC)	Read Only
\$_0C	FIFO Identifier Acceptance Register (BFFIDAC)	Read/Write <sup>1</sup>
\$_0D	FIFO Identifier Mask Register (BFFIDMR)	Read/Write <sup>1</sup>
\$_0E	Module Version Register (BFMVR)	Read Only
\$_0F	Reserved	
\$_10	Byteflight Port Control Register (BFPCTLBF)	Read/Write
\$_11	Reserved	
\$_12	Buffer Lock Register (BFBUFLOCK)	Read Only
\$_13	Reserved	
\$_14	FIFO Identifier Rejection Register (BFFIDRJ)	Read/Write <sup>1</sup>
\$_15	FIFO Identifier Rejection Mask Register (BFFIDRMR)	Read/Write <sup>1</sup>
\$_16- \$_1F	Reserved	
\$_20	Transmit Identifier Register (BFTIDENT)	Read/Write
\$_21	Transmit Data Length Register (BFTLEN)	Read/Write
\$_22	Transmit Data Register 0 (BFTDATA0)	Read/Write
\$_23	Transmit Data Register 1 (BFTDATA1)	Read/Write
\$_24	Transmit Data Register 2 (BFTDATA2)	Read/Write
\$_25	Transmit Data Register 3 (BFTDATA3)	Read/Write
\$_26	Transmit Data Register 4 (BFTDATA4)	Read/Write
\$_27	Transmit Data Register 5 (BFTDATA5)	Read/Write
\$_28	Transmit Data Register 6 (BFTDATA6)	Read/Write
\$_29	Transmit Data Register 7 (BFTDATA7)	Read/Write
\$_2A	Transmit Data Register 8 (BFTDATA8)	Read/Write
\$_2B	Transmit Data Register 9 (BFTDATA9)	Read/Write
\$_2C	Transmit Data Register 10 (BFTDATA10)	Read/Write

**Table 3-1 Module Memory Map**

\$_2D	Transmit Data Register 11 (BFTDATA11)	Read/Write
\$_2E	Reserved	
\$_2F	Reserved	
\$_30	Receive Identifier Register (BFRIDENT)	Read/Write <sup>5</sup>
\$_31	Receive Data Length Register (BFRLEN)	Read/Write <sup>5</sup>
\$_32	Receive Data Register 0 (BFRDATA0)	Read/Write <sup>5</sup>
\$_33	Receive Data Register 1 (BFRDATA1)	Read/Write <sup>5</sup>
\$_34	Receive Data Register 2 (BFRDATA2)	Read/Write <sup>5</sup>
\$_35	Receive Data Register 3 (BFRDATA3)	Read/Write <sup>5</sup>
\$_36	Receive Data Register 4 (BFRDATA4)	Read/Write <sup>5</sup>
\$_37	Receive Data Register 5 (BFRDATA5)	Read/Write <sup>5</sup>
\$_38	Receive Data Register 6 (BFRDATA6)	Read/Write <sup>5</sup>
\$_39	Receive Data Register 7 (BFRDATA7)	Read/Write <sup>5</sup>
\$_3A	Receive Data Register 8 (BFRDATA8)	Read/Write <sup>5</sup>
\$_3B	Receive Data Register 9 (BFRDATA9)	Read/Write <sup>5</sup>
\$_3C	Receive Data Register 10 (BFRDATA10)	Read/Write <sup>5</sup>
\$_3D	Receive Data Register 11 (BFRDATA11)	Read/Write <sup>5</sup>
\$_3E	Reserved	
\$_3F	Reserved	

**Table 3-1 Module Memory Map**

\$_40	Receive FIFO Identifier Register (BFFIDENT)	Read/Write <sup>1</sup>
\$_41	Receive FIFO Data Length Register (BFFLEN)	Read/Write <sup>1</sup>
\$_42	Receive FIFO Data Register 0 (BFFDATA0)	Read/Write <sup>1</sup>
\$_43	Receive FIFO Data Register 1 (BFFDATA1)	Read/Write <sup>1</sup>
\$_44	Receive FIFO Data Register 2 (BFFDATA2)	Read/Write <sup>1</sup>
\$_45	Receive FIFO Data Register 3 (BFFDATA3)	Read/Write <sup>1</sup>
\$_46	Receive FIFO Data Register 4 (BFFDATA4)	Read/Write <sup>1</sup>
\$_47	Receive FIFO Data Register 5 (BFFDATA5)	Read/Write <sup>1</sup>
\$_48	Receive FIFO Data Register 6 (BFFDATA6)	Read/Write <sup>1</sup>
\$_49	Receive FIFO Data Register 7 (BFFDATA7)	Read/Write <sup>1</sup>
\$_4A	Receive FIFO Data Register8 (BFFDATA8)	Read/Write <sup>1</sup>
\$_4B	Receive FIFO Data Register 9 (BFFDATA9)	Read/Write <sup>1</sup>
\$_4C	Receive FIFO Data Register 10 (BFFDATA10)	Read/Write <sup>1</sup>
\$_4D	Receive FIFO Data Register 11 (BFFDATA11)	Read/Write <sup>1</sup>
\$_4E	Reserved	
\$_4F	Reserved	
\$_50	Message Buffer Control Register 0 (BFBUFCTL0)	Read/Write <sup>2</sup>
\$_51	Message Buffer Control Register 1 (BFBUFCTL1)	Read/Write <sup>2</sup>
\$_52	Message Buffer Control Register 2 (BFBUFCTL2)	Read/Write <sup>2</sup>
\$_53	Message Buffer Control Register 3 (BFBUFCTL3)	Read/Write <sup>2</sup>
\$_54	Message Buffer Control Register 4 (BFBUFCTL4)	Read/Write <sup>2</sup>
\$_55	Message Buffer Control Register 5 (BFBUFCTL5)	Read/Write <sup>2</sup>
\$_56	Message Buffer Control Register 6 (BFBUFCTL6)	Read/Write <sup>2</sup>
\$_57	Message Buffer Control Register 7 (BFBUFCTL7)	Read/Write <sup>2</sup>
\$_58	Message Buffer Control Register 8 (BFBUFCTL8)	Read/Write <sup>2</sup>
\$_59	Message Buffer Control Register 9 (BFBUFCTL9)	Read/Write <sup>2</sup>
\$_5A	Message Buffer Control Register 10 (BFBUFCTL10)	Read/Write <sup>2</sup>
\$_5B	Message Buffer Control Register 11 (BFBUFCTL11)	Read/Write <sup>2</sup>
\$_5C	Message Buffer Control Register 12 (BFBUFCTL12)	Read/Write <sup>2</sup>
\$_5D	Message Buffer Control Register 13 (BFBUFCTL13)	Read/Write <sup>2</sup>
\$_5E	Message Buffer Control Register 14 (BFBUFCTL14)	Read/Write <sup>2</sup>
\$_5F	Message Buffer Control Register 15 (BFBUFCTL15)	Read/Write <sup>2</sup>

**Table 3-1 Module Memory Map**

NOTES:

1. Registers can only be updated during Initialization Mode (INITRQ=INITAK=1, BFEN="X").
2. Registers have some bits which can be written only during Initialization Mode (INITRQ=INITAK=1, BFEN="X").
3. Registers have some bits which are read and clear only.
5. Registers can only be updated during Initialization Mode (INITRQ=INITAK=1, BFEN="X") or test mode.

### 3.2.1 Programmer’s Model

This section describes the content and use of the registers in the Byteflight module. An overview of all registers is shown in **Figure 3-1**.

There are three sets of registers which are accessible by the CPU:

- the general control registers,
- the 16 buffer control registers,
- and the 16 configured message buffers with one 16 bytes active transmit buffer, one 16 bytes active receive buffer and one 16 bytes active receive FIFO buffer mirrored to the memory map.

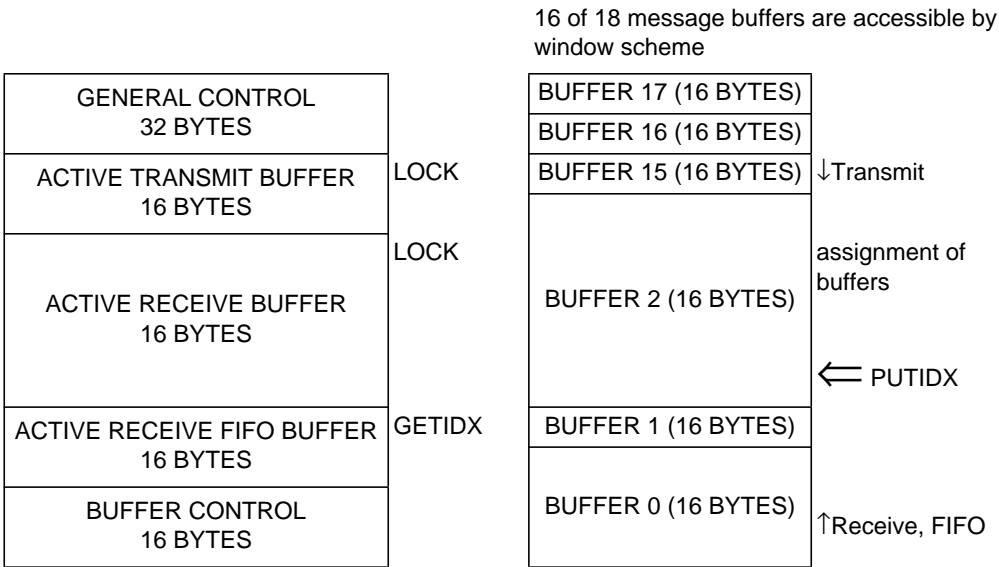


Figure 3-1 Register Map

### 3.3 Register Descriptions

#### 3.3.1 Module Configuration Register (BFMCR)

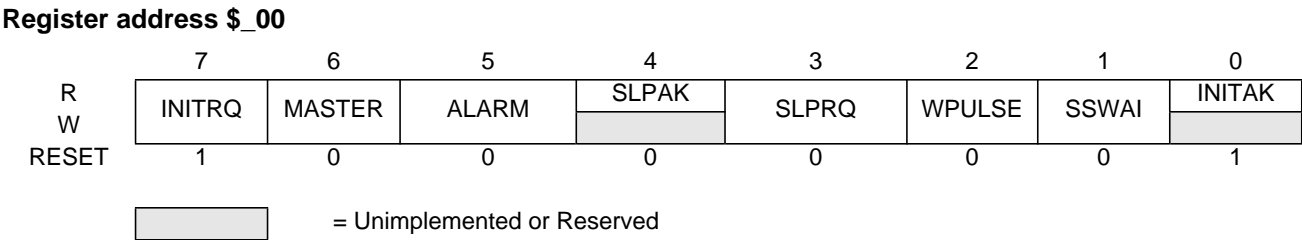


Figure 3-2 Module Configuration Register (BFMCR)

Read: Anytime

Write: Anytime; exceptions are bits MASTER, WPULSE which can be written only during Initialization Mode (INITRQ=INITAK=1) and SLPRQ which can only be written when BFEN bit in BFPCTLBF register is set

##### INITRQ — Initialization Request

When this bit is set by the CPU, the module immediately enters the initialization state. Any ongoing transmission or reception is aborted immediately and synchronization to the bus is lost. The module indicates entry to Initialization Mode by setting INITAK=1.

When this bit is cleared by the CPU, the interface will start to connect back to the bus. It will be synchronized after the next SYNC pulse on the bus.

- 1 = Initialization Mode.
- 0 = Normal operation.

##### MASTER — Master Select

This bit selects the node as bus master or as bus slave. This bit can only be written in Initialization Mode (INITRQ=INITAK=1).

- 1 = The interface is a master and generates the SYNC pulses.
- 0 = The interface is a slave and verifies the SYNC pulses.

##### ALARM — Master Alarm Pulses

If the MASTER bit and the ALARM bit are set, ALARM pulses are transmitted.

- 1 = The interface generates ALARM pulses on next t<sub>cyc</sub> (**Table 4-1**).
- 0 = The interface generates normal SYNC pulses.

The ALARM bit is reset after t<sub>alarm\_rst</sub> (**Table 4-1**) if it has not been set again by the CPU within that period. This bit can be set and cleared any time.

### SLPAK — Sleep Acknowledge

This read-only bit indicates that the interface has reached the sleep mode state. This allows an accurate shut off of the module before the shut off of the remaining system. As soon as the on-going message transmission or reception is finished this bit gets set. Any bus activity on receive pin will clear this bit.

1 = Module is in Sleep mode.

0 = Module is in Normal run mode.

### SLPRQ — Sleep Request, Enter Low Power Module Sleep Mode

This bit requests the module to enter sleep mode, the clock will be stopped. This allows the module to be shut off when not in use. The CPU can still access the Module Configuration Register. If SLPRQ is asserted in the middle of transmission or reception of a message or is asserted while the temporary message buffers are full, the clock will be stopped at the end of this message, respectively when the temporarily buffers are empty. It remains stopped until a wakeup occurs. Any bus activity on receive pin will clear this bit asynchronously. The module indicates entry to Sleep Mode by setting SLPK=1. This bit can only be written if BFEN bit in BFPCTLBF register is set.

1 = Enter Low Power Sleep Mode.

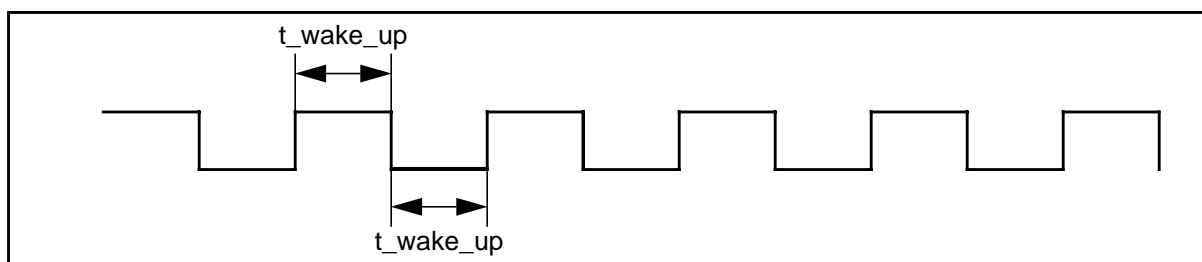
0 = Normal Mode.

### WPULSE — Wakeup Pulses

This bit causes to transmit wakeup pulses with the pulse width of  $t_{wake\_up}$  (**Table 4-1**) on transmit pin. The wakeup sequence is used to wakeup the transceiver from sleep mode. This bit can only be written in Initialization Mode (INITRQ=INITAK=1).

1 = The interface generates wakeup pulses.

0 = The interface does not generate wakeup pulses.



**Figure 3-3 Wakeup Pulses**

### SSWAI — Byteflight Module Stops in Wait Mode

1 = The module ceases to be clocked during WAIT mode.

0 = The module is not affected during WAIT mode.

### INITAK — Initialization Acknowledge

This read-only bit indicates that the interface has reached the initialization mode state. It is used as a handshake flag for the INITRQ Initialization Mode request. This allows an accurate shut off of the module before the shut off of the remaining system. Initialization Mode is active when INITRQ=1 and INITAK=1.

1 = Module is in Initialization mode.

0 = Module is in Normal run mode.

3.3.2 FIFO Size Register (BFFSIZR)

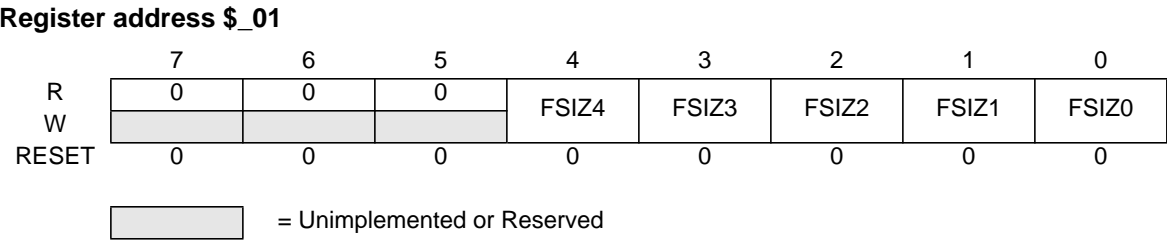


Figure 3-4 FIFO Size Register (BFFSIZR)

Read: Anytime

Write: Can only be written in Initialization Mode (INITRQ=INITAK=1)

FSIZ4:FSIZ0 — FIFO Size Bits

These bits are used to configure the FIFO. The number of buffers, starting from buffer 0, assigned to the receive FIFO can be selected. The corresponding buffer(s) must be assigned as Receive buffer by the CFG bits in the message buffer control register(s).

Table 3-2 FIFO Size

FSIZ[4:0]	FIFO Size
00000	No FIFO
00001	buffer 0
00010	buffers 0-1
00011	buffers 0-2
:	:
01111	buffers 0-14
10000	buffers 0-15

**NOTE:** The BFFSIZR register can only be written if the INITRQ and INITAK bits in the Module Configuration Register (BFMCR) are set. If FIFO depth overlaps with buffers configured as dedicated RX buffer then all overlapping buffers would be treated as FIFO buffers. Only allowed values for FSIZ[4:0] bits are “00000” to “10000”.



3.3.3 Time Configuration Register 1 (BFTCR1)

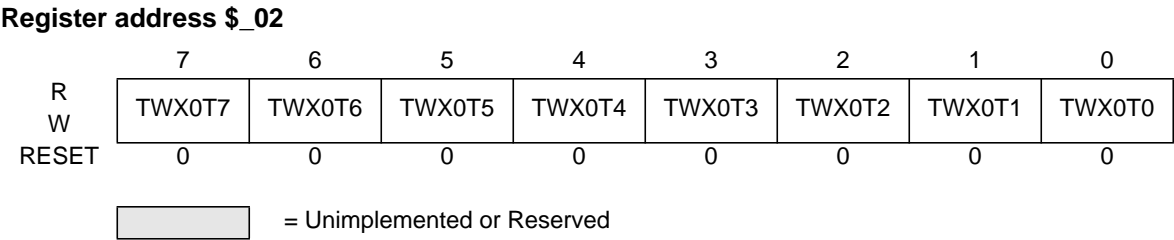


Figure 3-5 Time Configuration Register 1 (BFTCR1)

Read: Anytime

Write: Can only be written in Initialization Mode (INITRQ=INITAK=1)

The time configuration register 1 contains the specific programmable time  $t_{wx0\_tx}$  (Table 4-1) for the node. Only a hard reset will clear the register.

TWX0T7:TWX0T0 — Time  $t_{wx0\_tx}$

These bits select the Offset Time  $t_{wx0\_tx}$  as shown in Table 3-3.

Table 3-3 Time  $t_{wx0\_tx}$

TWX0T[7:0]	$t_{wx0\_tx}[ns]/25 - 7$
------------	--------------------------

**NOTE:** The BFTCR1 register can only be written if the INITRQ and INITAK bits in the Module Configuration Register (BFMCR) are set.

3.3.4 Time Configuration Register 2 (BFTCR2)

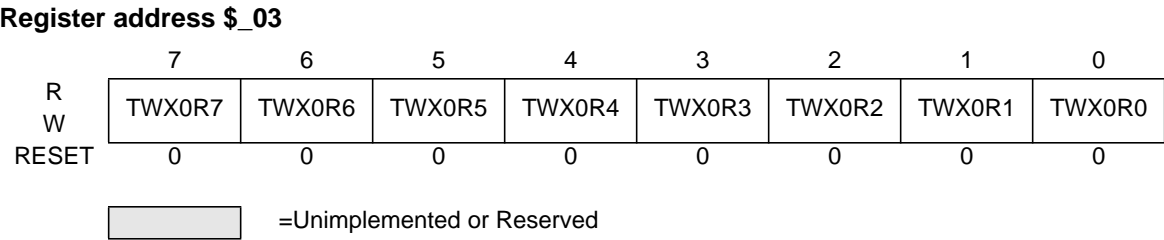


Figure 3-6 Time Configuration Register 2 (BFTCR2)

Read: Anytime

Write: Can only be written in Initialization Mode (INITRQ=INITAK=1)

The time configuration register 2 contains the specific programmable time  $t_{wx0\_rx}$  (Table 4-1) for the node. Only a hard reset will clear the register.

TWX0R7:TWX0R0 — Time  $t_{wx0\_rx}$

These bits select the Offset Time  $t_{wx0\_rx}$  as shown in Table 3-4.

Table 3-4 Time  $t_{wx0\_rx}$

TWX0R[7:0]	$t_{wx0\_rx}[ns]/25 - 7$
------------	--------------------------

**NOTE:** The BFTCR2 register can only be written if the INITRQ and INITAK bits in the Module Configuration Register (BFMCR) are set.

3.3.5 Time Configuration Register 3 (BFTCR3)

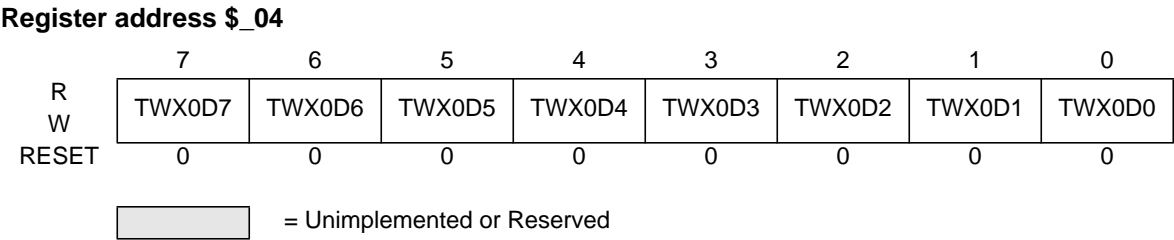


Figure 3-7 Time Configuration Register 3 (BFTCR3)

Read: Anytime

Write: Can only be written in Initialization Mode (INITRQ=INITAK=1)

The time configuration register 3 contains the specific programmable time `t_wx0_delta` (Table 4-1) for the node. Only a hard reset will clear the register.

TWX0D7:TWX0D0 — Time `t_wx0_delta`

These bits contain the value for the time `t_wx0_delta` as shown in Table 3-5.

Table 3-5 Time `t_wx0_delta`


TWX0D[7:0]	<code>t_wx0_delta[ns]/25 – 1</code>
------------	-------------------------------------

**NOTE:** The BFTCR3 register can only be written if the INITRQ and INITAK bits in the Module Configuration Register (BFMCR) are set. The lower bound for BFTCR3 is 3 which is not enforced by the hardware and can be taken care by the application software.

### 3.3.6 Receive Interrupt Status Register (BFRISR)

Register address \$\_06

	7	6	5	4	3	2	1	0
R	RCVFIF	RXIF	SYNAIF	SYNNIF	SLMMIF	0	XSYNIF	OPTDF
W								
RESET	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 3-8 Receive Interrupt Status Register (BFRISR)**

Read: Anytime

Write: Can't be written

The Receive Interrupt Status Register indicates the occurrence of receive or SYNC pulse events, and together with the Receive Interrupt Enable Register allows the module to operate in a polled or interrupt driven system. Different interrupts can be generated. The Optical Diagnosis Flag (OPTDF) is an indicator for the quality of the optical line between two nodes. The OPTDF flag uses the internal error bit of the optical transceiver which is driven on the receive line when the photo stream is below a certain level (*Refer to Infineon SPF BFT003*).

SYNAIF, SYNNIF, SLMMIF, XSYNIF and OPTDF are read and clear only. RCVFIF and RXIF are read only and cleared when the set condition is no longer valid. A flag can be cleared by writing a “1” to the corresponding bit position. Only one write is necessary to request the clearing, i.e. the clear request is stored and executed as soon as the condition which caused the setting is no longer valid. Writing a “0” has no effect on the flag setting. Every flag has an associated interrupt enable flag in the Receive Interrupt Enable Register. A hard reset or setting of INITRQ bit in BFMCR will clear the register.

#### RCVFIF — Receive FIFO Not Empty Interrupt Flag

This read-only bit will be set when the Receive FIFO is not empty so it indicates that RX FIFO is ready to read. If enabled, a FIFO not empty interrupt is pending while this flag is set. The flag will be cleared if the FIFO is empty and when buffer 0 is unlocked. The flag remains set if the FIFO is not empty.

1 = Receive FIFO is not empty or at least one RX FIFO buffer is full.

0 = Receive FIFO is empty or none of the RX FIFO buffer is full.

**NOTE:** *RCVFIF bit is cleared immediately after the last FIFO buffer is read.*

#### RXIF — Receive Interrupt Flag

This read-only bit will be set when any of the enabled (IENAn=1) receive buffers has successfully received a message so it indicates that RX buffer is ready to read. It can be cleared by clearing of the IFLG bit(s) of the corresponding buffer(s). If RXIE is set, a receive interrupt is pending while this flag is set.

1 = At least one receive buffer is full.

0 = All receive buffers are empty.

#### SYNAIF — Synchronization Pulse ALARM Interrupt Flag

This bit will be set when a SYNC pulse ALARM has been received. If enabled, a SYNC pulse interrupt is pending while this flag is set.

- 1 = SYNC pulse ALARM has been received.
- 0 = No SYNC pulse ALARM has occurred.

#### SYNNIF — Synchronization Pulse NORMAL Interrupt Flag

This bit will be set when a SYNC pulse NORMAL has been received. If enabled, a SYNC pulse interrupt is pending while this flag is set.

- 1 = SYNC pulse NORMAL has been received.
- 0 = No SYNC pulse NORMAL has occurred.

#### SLMMIF — Slot Mismatch Interrupt Flag

This bit will be set when the identifier of a successfully received message differs from the current time slot.

- 1 = A mismatch between successfully received identifier and slot counter has occurred.
- 0 = No slot mismatch has occurred.

#### XSYNIF — XSYNC Pulse Interrupt Flag

This bit will be set when a valid SYNC pulse (NORMAL or ALARM Sync pulse) has been received. If enabled, a XSYNC pulse interrupt is pending while this flag is set.

- 1 = SYNC pulse has been received.
- 0 = No SYNC pulse has occurred.

#### OPTDF — Optical Diagnosis Flag


This bit will be set when the internal error bit of the optical transceiver is driven on the receive line during transmission.

- 1 = A pulse on receive pin has been detected.
- 0 = No pulse on receive pin has been detected.

### 3.3.7 General Interrupt Status Register (BFGISR)

Register address \$\_07

	7	6	5	4	3	2	1	0
R	TXIF	OVRNIF	ERRIF	SYNEIF	SYNLIF	ILLPIF	LOCKIF	WAKEIF
W								
RESET	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 3-9 General Interrupt Status Register (BFGISR)**

Read: Anytime

Write: Can't be written

The General Interrupt Status Register indicates the occurrence of system events, and together with the General Interrupt Enable Register allows the module to operate in a polled or interrupt driven system. A general interrupt can be generated.

OVRNIF, ERRIF, SYNEIF, SYNLIF, ILLPIF and WAKEIF are read and clear only. TXIF and LOCKIF are read only and cleared when the set condition is no longer valid. A flag can be cleared by writing a “1” to the corresponding bit position. Only one write is necessary to request the clearing, i.e. the clear request is stored and executed as soon as the condition which caused the setting is no longer valid. SYNLIF can only cleared after receiving a valid sync pulse. Writing a “0” has no effect on the flag setting. Every flag has an associated interrupt enable bit in the General Interrupt Enable Register. A hard or setting of INTRQ bit in BFMCR will clear the register (except LOCKIF, which can only be cleared by a hard or system reset).

#### TXIF — Transmit Interrupt Flag

This read-only bit will be set when any of the enabled (IENAn=1) transmit buffers is empty. It can be cleared by clearing the IFLG bit(s) of the corresponding buffer(s). After leaving through the Initialization Mode the flag is set if there is at least one transmit buffer configured. If TXIE is set, a general interrupt is pending while this flag is set.

- 1 = At least one transmit buffer is empty.
- 0 = All transmit buffers are full.

#### OVRNIF — Receive FIFO Overrun Interrupt Flag

This bit will be set when a Receive FIFO overrun occurred. If enabled, a general interrupt is pending while this flag is set.

- 1 = A Receive FIFO overrun has been detected.
- 0 = No Receive FIFO overrun has occurred.

#### ERRIF — Message Format Error (CRC, Frame) Interrupt Flag

This bit will be set when a Message Format Error (CRC Error, Frame Error, **Table 4-4**) occurred. If enabled, a general interrupt is pending while this flag is set.

- 1 = A Message Format Error has been detected.

0 = No Message Format Error has occurred.

#### SYNEIF — SYNC Pulse Too Early Error Interrupt Flag

This bit will be set when a SYNC pulse too early error (**Table 4-4**) appears. If enabled, a general interrupt is pending while this flag is set.

1 = SYNC pulse too early error.

0 = No SYNC pulse too early error has occurred.

#### SYNLIF — SYNC Pulse Lost Error Interrupt Flag

This bit will be set when a SYNC pulse lost error (**Table 4-4**) appears. If enabled, a general interrupt is pending while this flag is set.

1 = SYNC pulse lost error.

0 = No SYNC pulse lost error has occurred.

#### ILLPIF — Illegal Pulse Error Interrupt Flag

This bit will be set when an illegal pulse error (**Table 4-4**) appears. If enabled, a general interrupt is pending while this flag is set.

1 = Illegal pulse error.

0 = No illegal pulse error has occurred.

#### LOCKIF — Locking Error Interrupt Flag

This bit will be set when two receive buffers or two transmit buffers are locked at the same time. If enabled, a general interrupt is pending while this flag is set and the module enters initialization mode. The bit is cleared by a hard or system reset. This bit can also be cleared by unlocking the second buffer. The bit value doesn't change while entering into Initialization Mode.

1 = Locking error.

0 = No locking error has occurred.

#### WAKEIF — WAKEUP Interrupt Flag

This bit will be set when the module detects bus activity on receive pin while it is asleep (SLPAK=1). If enabled, a general interrupt is pending while this flag is set.


1 = Detected activity on the receive pin.

0 = No wake-up activity has been observed while in sleep mode (SLPRQ=SLPAK=1).

### 3.3.8 Receive Interrupt Enable Register (BFRIER)

Register address \$\_08

	7	6	5	4	3	2	1	0
R	RCVFIE	RXIE	SYNAIE	SYNNIE	SLMMIE	0	XSYNIE	0
W								
RESET	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 3-10 Receive Interrupt Enable Register (BFRIER)**

Read: Anytime

Write: Anytime

The Receive Interrupt Enable Register allows to enable/disable the different receive or SYNC pulse interrupt sources for different interrupt requests. A hard reset or setting of INITRQ bit in BFMCR will clear the register.

**RCVFIE** — Receive FIFO Not Empty Interrupt Enable

- 1 = A Receive FIFO not empty event will result in a receive FIFO not empty interrupt.
- 0 = No interrupt will be generated from this event.

**RXIE** — Receive Interrupt Enable

- 1 = A receive event will result in a receive interrupt.
- 0 = No interrupt will be generated from this event.

**SYNAIE** — Synchronization Pulse ALARM Interrupt Enable

- 1 = A SYNC pulse ALARM event will result in a SYNC pulse interrupt.
- 0 = No interrupt will be generated from this event.

**SYNNIE** — Synchronization Pulse NORMAL Interrupt Enable

- 1 = A SYNC pulse NORMAL event will result in a SYNC pulse interrupt.
- 0 = No interrupt will be generated from this event.

**SLMMIE** — Slot Mismatch Interrupt Enable

- 1 = A slot mismatch event will result in a general interrupt.
- 0 = No interrupt will be generated from this event.

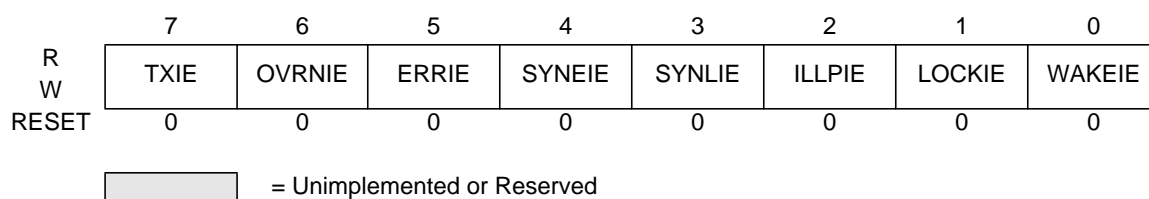
**XSYNIE** — XSYNC Pulse Interrupt Enable

- 1 = A SYNC pulse event will result in a XSYNC pulse interrupt.
- 0 = No interrupt will be generated from this event.



### 3.3.9 General Interrupt Enable Register (BFGIER)

Register address \$\_09



**Figure 3-11 General Interrupt Enable Register (BFGIER)**

Read: Anytime

Write: Anytime; exception is bit LOCKIE which can be written only during Initialization Mode (INITRQ=INITAK=1)

The General Interrupt Enable Register allows to enable/disable the different interrupt sources for the general interrupt request. A hard reset or setting of INITRQ bit in BFMCR will clear the register except LOCKIE which can be cleared only during hard reset.

**TXIE** — Transmit Interrupt Enable

1 = A transmit event will result in a general interrupt.

0 = No interrupt will be generated from this event.

**OVRNIE** — Receive FIFO Overrun Interrupt Enable

1 = A Receive FIFO overrun event will result in a general interrupt.

0 = No interrupt will be generated from this event.

**ERRIE** — Message Format Error (CRC or Frame) Interrupt Enable

1 = A Message Format Error (**Table 4-4**) will result in a general interrupt.

0 = No interrupt will be generated from this event.

**SYNEIE** — SYNC Pulse Too Early Error Interrupt Enable

1 = A SYNC pulse too early error (**Table 4-4**) will result in a general interrupt.

0 = No interrupt will be generated from this event.

**SYNLIE** — SYNC Pulse Lost Error Interrupt Enable

1 = A SYNC pulse lost error (**Table 4-4**) will result in a general interrupt.

0 = No interrupt will be generated from this event.

**ILLPIE** — Illegal Pulse Error Interrupt Enable

1 = An illegal pulse error (**Table 4-4**) will result in a general interrupt.

0 = No interrupt will be generated from this event.

**LOCKIE** — Locking Error Interrupt Enable

1 = A locking error will result in a general interrupt and the module will enter in initialization mode (INITRQ=INITAK=1).

0 = No interrupt will be generated from this event and the module does not enter into initialization mode.

**NOTE:** *The LOCKIE bit can only be written if the INITRQ and INITAK bits in the Module Configuration Register (BFMCR) are set.*

WAKEIE — WAKEUP Interrupt Enable


1 = A requested wake-up will result in a general interrupt.

0 = No interrupt will be generated from this event.

### 3.3.10 Receive Interrupt Source Register (BFRIVEC)

Register address \$\_0A

	7	6	5	4	3	2	1	0
R	0	0	0	0	RIVEC3	RIVEC2	RIVEC1	RIVEC0
W								
RESET	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 3-12 Receive Interrupt Source Register (BFRIVEC)**

Read: Anytime but the register contains the valid data only when RXIF bit in BFRISR register is set

Write: Can't be written

The read-only Receive Interrupt Vector Register shows the lowest numbered message buffer which has its interrupt status flag (IFLG) and its interrupt enable bit (IENA) set. A hard or setting of INITRQ bit in BFMCR will clear the register.

**NOTE:** *The Receive Interrupt Vector Register contains only valid data if RXIF is set.*

3.3.11 Transmit Interrupt Source Register (BFTIVEC)

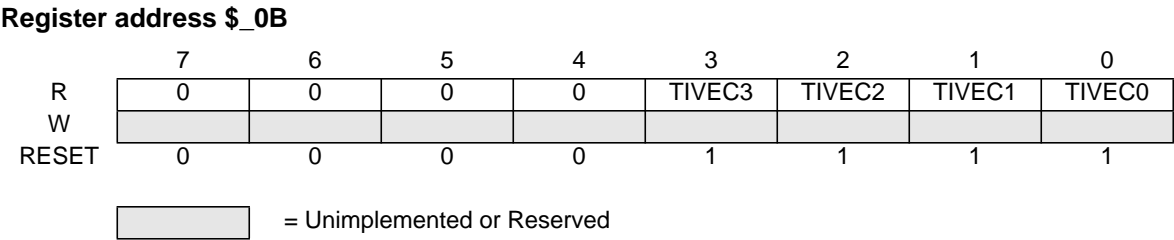


Figure 3-13 Transmit Interrupt Source Register (BFTIVEC)

Read: Anytime but the register contains valid data only when TXIF bit in BFGISR register is set

Write: Can't be written


The read-only Transmit Interrupt Vector Register shows the highest numbered message buffer which has its interrupt status flag (IFLG) and its interrupt enable bit (IENA) set. A hard or setting of INITRQ bit in BFMCR will set the register.

**NOTE:** The Transmit Interrupt Vector Register contains only valid data if TXIF is set.

### 3.3.12 FIFO Identifier Acceptance Register (BFFIDAC)

Register address \$\_0C

	7	6	5	4	3	2	1	0
R	FIDAC7	FIDAC6	FIDAC5	FIDAC4	FIDAC3	FIDAC2	FIDAC1	FIDAC0
W								
RESET	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 3-14 FIFO Identifier Acceptance Register (BFFIDAC)**

Read: Anytime

Write: Can only be written in Initialization Mode (INITRQ=INITAK=1)

The FIFO identifier acceptance register defines a user specified pattern of bits to which the incoming identifier is compared. This determines whether the message is accepted by the FIFO. Only a hard reset will clear the register.

FIDAC7:FIDAC0 — FIFO Identifier Acceptance bit 7:0

1 = Compare respective bit location with "1"

0 = Compare respective bit location with "0"

**NOTE:** The BFFIDAC register can only be written if the INITRQ and INITAK bits in the Module Configuration Register (BFMCR) are set.

3.3.13 FIFO Identifier Mask Register (BFFIDMR)

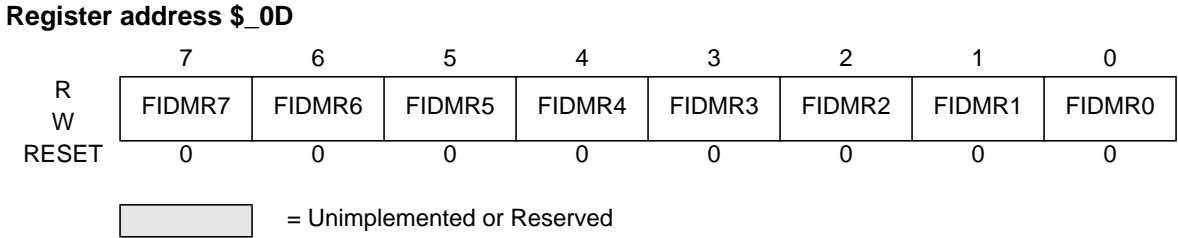


Figure 3-15 FIFO Identifier Mask Register (BFFIDMR)

Read: Anytime

Write: Can only be written in Initialization Mode (INITRQ=INITAK=1)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. A cleared bit in this register indicates that the corresponding bit in the identifier acceptance register must be the same as the incoming identifier bit, before a match will be detected. The message will be accepted by the FIFO if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted by the FIFO. Only a hard reset will clear the register.

FIDMR7:FIDMR0 — FIFO Identifier Mask bit 7:0

- 1 = Ignore corresponding acceptance register bit
- 0 = Match corresponding acceptance register bit

**NOTE:** The BFFIDMR register can only be written if the INITRQ and INITAK bits in the Module Configuration Register (BFMCR) are set.

3.3.14 Module Version Register (BFMVR)

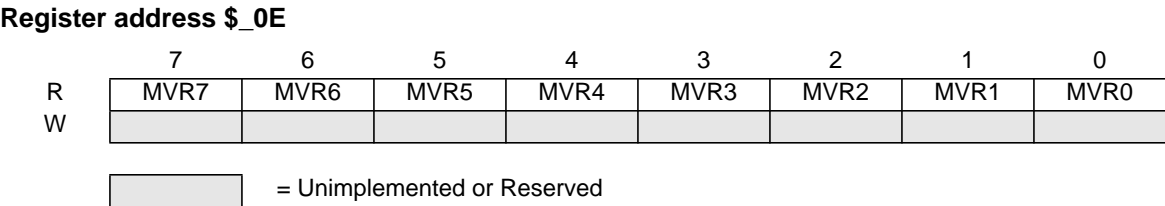


Figure 3-16 Module Version Register (BFMVR)

Read: Anytime


Write: Can't be written

The read-only Module Version Register contains the version number of the implementation. *Refer to the specific Device User Guide to determine the actual reset state of this register.*

### 3.3.15 Byteflight Port Control Register (BFPCTLBF)

Register address \$\_10

	7	6	5	4	3	2	1	0
R	PMEREN	0	PSLMEN	PERREN	PROKEN	PSYNEN	0	BFEN
W								
RESET	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 3-17 Byteflight Port Control Register (BFPCTLBF)**

Read: Anytime

Write: Anytime; exception is bit BFEN which is write once in normal modes and anytime in special modes when the Byteflight module is in Initialization Mode (INITRQ=1 and INITAK=1)

The following bits control the Byteflight module as well as the status output pins. *Refer to the Port Integration User Guide and the Device User Guide for more details on pin assignments and priorities.* Only a hard reset will clear all the bits of the register.

**PMEREN** — Slot Mismatch Error Enable

- 1 = A 50ns pulse is driven on pin *BF\_PERR* after a slot mismatch, if enabled by the PERREN bit. This function is only valid if BFEN is set.
- 0 = the Byteflight module does not request ownership of the associated pin.

**PSLMEN** — Slot Mismatch Enable

- 1 = A 50ns pulse is driven on pin *BF\_PSLM* after a slot mismatch. This function is only valid if BFEN is set.
- 0 = the Byteflight module does not request ownership of the associated pin.

**PERREN** — Error Pulse Enable

- 1 = A 50ns pulse is driven on pin *BF\_PERR* after a message format or illegal pulse error or after slot mismatch if enabled by the PMEREN bit. This function is only valid if BFEN is set.
- 0 = the Byteflight module does not request ownership of the associated pin.

**PROKEN** — Reception OK Pulse Enable

- 1 = A 50ns pulse is driven on pin *BF\_PROK* after the successful reception of a message or sync pulse. This function is only valid if BFEN is set.
- 0 = the Byteflight module does not request ownership of the associated pin.

**PSYNEN** — Sync Pulse Enable

- 1 = A 50ns pulse is driven on pin *BF\_PSYN* after the successful reception or transmission of a sync pulse. This function is only valid if BFEN is set.
- 0 = the Byteflight module does not request ownership of the associated pin.

**NOTE:** If BFEN is “0” then all these bits are don’t care.



## BFEN — Byteflight Enable

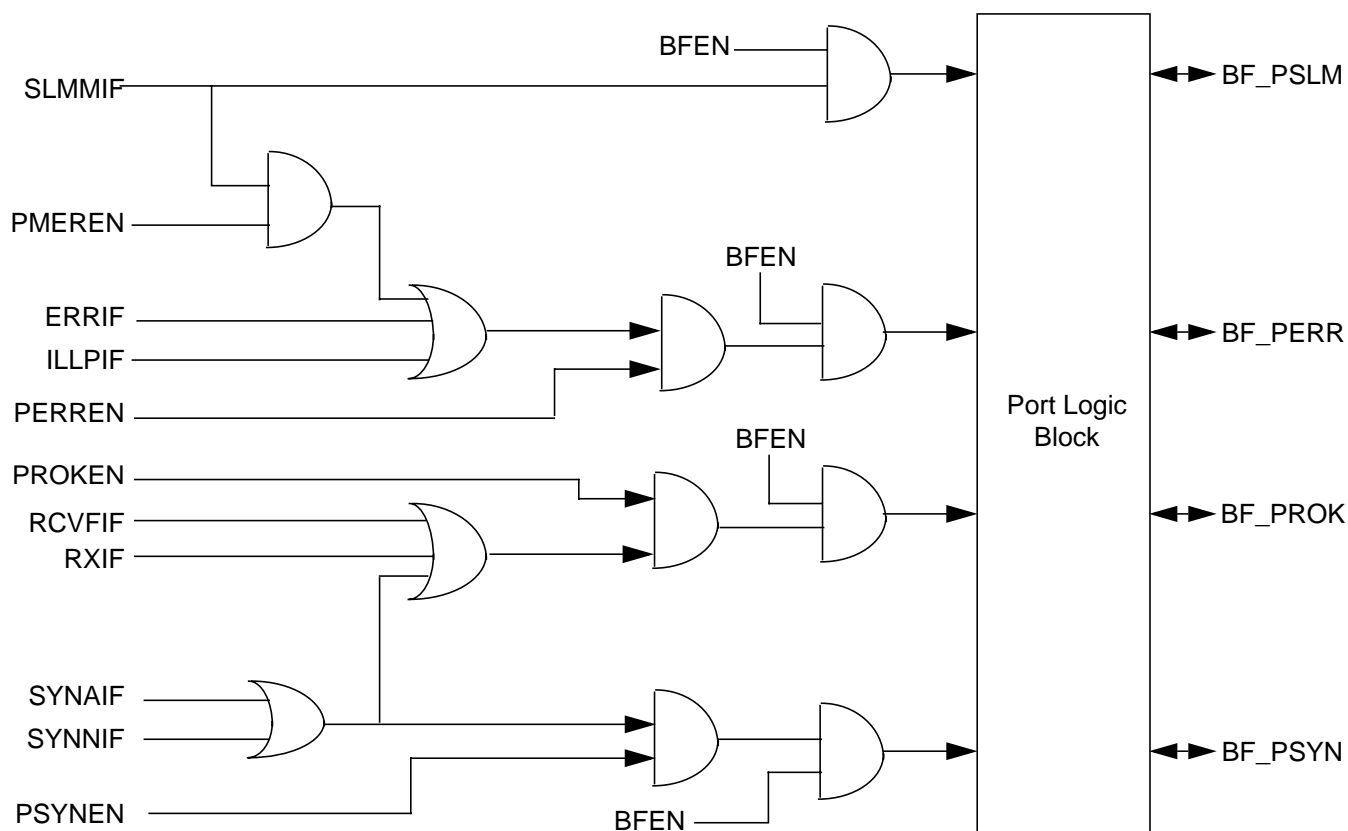
This bit indicates whether the Byteflight module is enabled or not. Writing to this bit is allowed only once in non-Special modes and anytime in Special mode when the Byteflight module is in Initialization mode (INITRQ=INITAK=1) (**Table 3-6**).

1 = Byteflight module is enabled and requests ownership of pins *RX\_BF* and *TX\_BF*.

0 = Byteflight module disabled.

**Table 3-6 Conditions for updating the BFEN bit**

Initialization Mode (INITRQ=INITAK=1)	Special Mode	BFEN
0	0	Can't be written
0	1	Can't be written
1	0	Can be written only once
1	1	Can be written any number of times



**Figure 3-18 SLMM, ERR, ROK, SYNC Signal Generation**

3.3.16 Buffer Lock Register (BFBUFLOCK)

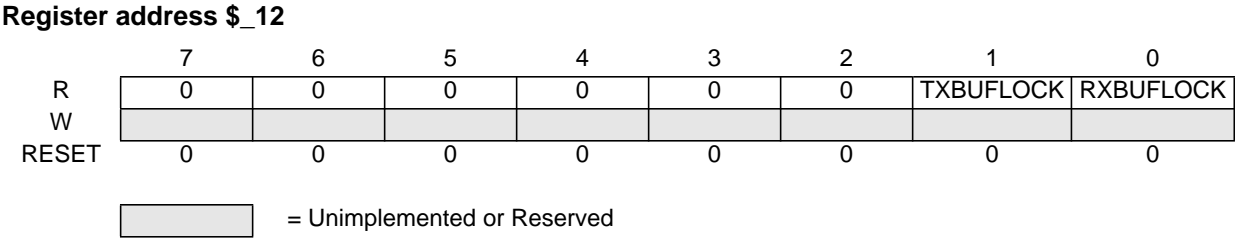


Figure 3-19 Buffer Lock Register (BFBUFLOCK)

Read: Anytime

Write: Can't be written

The register bits are read only. The status bits are cleared as soon as the set condition is no longer valid.

TXBUFLOCK — TX Buffer Locked

- 1 = A buffer configured as transmit buffer is locked by CPU.
- 0 = None of the buffers configured as transmit buffer is locked by CPU.

RXBUFLOCK — RX Buffer Locked

- 1 = A buffer configured as receive buffer is locked by CPU.
- 0 = None of the buffers configured as receive buffer is locked by CPU.

**NOTE:** When Lock Error occurs (LOCKIF=1) or all message buffers are configured as FIFO buffers then the value of TXBUFLOCK and RXBUFLOCK will be “0”.


Table 3-7 Setting conditions for the status bits

No. Of Transmit Buffer(s) Locked by CPU	No. Of Receive Buffer(s) Locked by CPU	FIFO buffer(s)	TXBUFLOCK	RXBUFLOCK
0	0	16	Always “0”	Always “0”
>0	0	“X”	Can become “1”	Always “0”
0	>0	“X”	Always “0”	Can become “1”
>0	>0	“X”	Can become “1”	Can become “1”

### 3.3.17 FIFO Identifier Rejection Register (BFFIDRJ)

Register address \$\_14

	7	6	5	4	3	2	1	0
R	FIDRJ7	FIDRJ6	FIDRJ5	FIDRJ4	FIDRJ3	FIDRJ2	FIDRJ1	FIDRJ0
W								
RESET	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 3-20 FIFO Identifier Rejection Register (BFFIDRJ)**

Read: Anytime

Write: Can only be written in Initialization Mode (INITRQ=INITAK=1)

The FIFO identifier rejection register defines a user specified pattern of bits to which the incoming identifier is compared. This determines whether the message is rejected by the FIFO. Only a hard reset will clear the register.

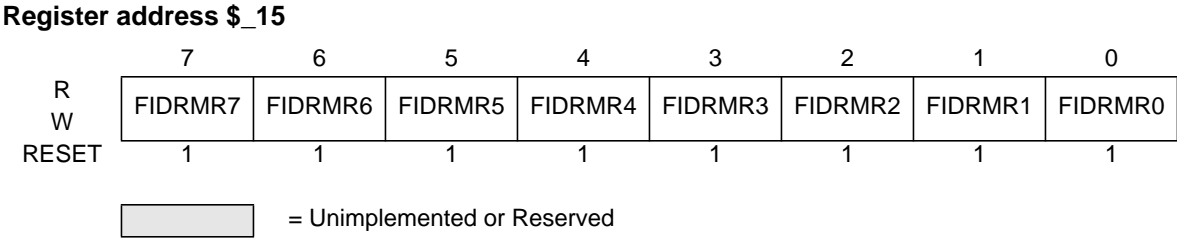
FIDRJ7:FIDRJ0 — FIFO Identifier Rejection bit 7:0

1 = Compare respective bit location with "1"

0 = Compare respective bit location with "0"

**NOTE:** The BFFIDRJ register can only be written if the INITRQ and INITAK bits in the Module Configuration Register (BFMCR) are set.

### 3.3.18 FIFO Identifier Rejection Mask Register (BFFIDRMR)



**Figure 3-21** FIFO Identifier Rejection Mask Register (BFFIDRMR)

Read: Anytime

Write: Can only be written in Initialization Mode (INITRQ=INITAK=1)

The identifier rejection mask register specifies which of the corresponding bits in the identifier rejection register are relevant for rejection filtering. A cleared bit in this register indicates that the corresponding bit in the identifier rejection register must be the same as the incoming identifier bit, before a match will be detected. The message will be rejected by the FIFO if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier rejection register will not affect whether or not the message is rejected by the FIFO. Only a hard reset will set the register.

FIDRMR7:FIDRMR0 — FIFO Identifier Rejection Mask bit 7:0

- 1 = Ignore corresponding rejection register bit
- 0 = Match corresponding rejection register bit

**NOTE:** *The BFFIDRMR register can only be written if the INITRQ and INITAK bits in the Module Configuration Register (BFMCR) are set. If acceptance and rejection filter set to match the same identifier, the message will be rejected.*

### 3.3.19 Transmit, Receive and Receive FIFO Buffer Registers

Register address \$\_20-\$\_4F

Address	Register		7	6	5	4	3	2	1	0
\$_20	BFTIDENT	R	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
		W								
\$_21	BFTLEN	R					LEN3	LEN2	LEN1	LEN0
		W								
\$_22-\$ \$_2D	BFTDATA0- BFTDATA11	R	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
		W								
\$_2E-\$ \$_2F	Reserved	R								
		W								
\$_30	BFRIDENT	R	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
		W								
\$_31	BFRLEN	R					LEN3	LEN2	LEN1	LEN0
		W								
\$_32-\$ \$_3D	BFRDATA0- BFRDATA11	R	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
		W								
\$_3E-\$ \$_3F	Reserved	R								
		W								
\$_40	BFFIDENT	R	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
		W								
\$_41	BFFLEN	R					LEN3	LEN2	LEN1	LEN0
		W								
\$_42-\$ \$_4D	BFFDATA0- BFFDATA11	R	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
		W								
\$_4E-\$ \$_4F	Reserved	R								
		W								

= Unimplemented or Reserved

**Figure 3-22 Active Transmit, Receive and Receive FIFO Buffer Registers**

There are 16 configurable message buffers, consisting of 16 bytes each. The organization of one message buffer is shown in **Table 3-8**. The first byte contains the 8-bit identifier (ID), the second byte contains the length (LEN) byte (the 4 msb are reserved), followed by a maximum of 12 data bytes (DATA0..11) and two reserved bytes. A LEN value greater than 12 will be transmitted and used in the CRC calculation of transmitter and receiver, but the lower nibble (bits 3-0) of LEN byte in the receive buffer are set to 12.

The message buffers are configurable as receive, receive FIFO or transmit buffers. The receive and receive FIFO buffer system starts with message buffer 0 and can be configured to the maximum of all 16 message buffers (no transmit buffer). The transmit buffer system starts with message buffer 15 and can be configured to the maximum of all 16 message buffers (no receive buffer).

**NOTE:** No mixing of receiver, FIFO or transmit buffers is allowed.

Only the ‘active’ buffers (transmit buffer, receive buffer and receive FIFO buffer) are addressable by the CPU and allocate 16 bytes (2 Bytes are Reserved) each in the memory map. The ‘non-active’ buffers do not appear in the memory map. To activate a buffer the buffer must be locked. Only one transmit

buffer, one receive buffer and one receive FIFO buffer may be locked at a time. A locking error interrupt flag is set and if enabled a locking interrupt is generated when the CPU tries to lock two buffers of the same type.

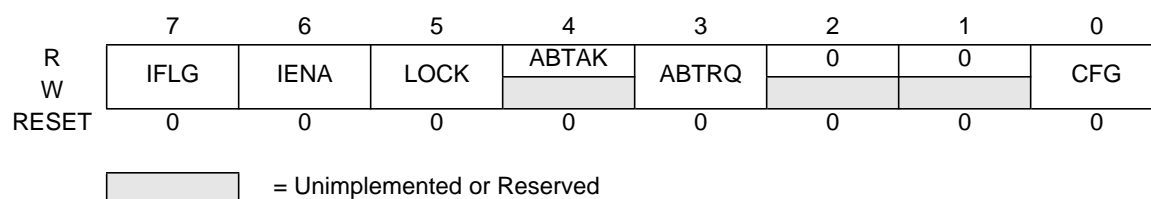
**NOTE:** *Active buffers (Transmit buffer(s), dedicated receive buffer(s) and receive FIFO buffer(s)) and all non-active buffers are physically located in SDPRAM. Reset value of active buffers (\$\_20-\$\_4F) depends upon the initialization value of SDPRAM.*

Addr	Register Name
\$xxx0	Identifier Register
\$xxx1	Data Length Register (4 msb reserved)
\$xxx2	Data Register 0
\$xxx3	Data Register 1
\$xxx4	Data Register 2
\$xxx5	Data Register 3
\$xxx6	Data Register 4
\$xxx7	Data Register 5
\$xxx8	Data Register 6
\$xxx9	Data Register 7
\$xxxA	Data Register 8
\$xxxB	Data Register 9
\$xxxC	Data Register 10
\$xxxD	Data Register 11
\$xxxE	Reserved
\$xxxF	Reserved

**Table 3-8 Message Buffer Organization**

### 3.3.20 Message Buffer Control Registers (BFBUFCTL0-BFBUFCTL15)

Register address \$\_50-\$\_5F



**Figure 3-23 Buffer Control Registers (BFBUFCTL0-BFBUFCTL15)**

Read: Anytime

Write: Anytime; exceptions are bit CFG which can be written only in Initialization Mode (INITRQ=INITAK=1) and ABTRQ bit which can only be written when BFEN bit in BFPCTLBF register is set

Each of the 16 configurable message buffers is associated with one Buffer Control Register. All 16 registers are addressable by the CPU. Only a hard reset will clear the register.

#### CFG — Message Buffer Configuration Bit

This bit is used to configure the corresponding buffer as transmit buffer or as receive or receive FIFO buffer. This bit is readable and is only cleared by hard reset. Writing to this bit is allowed during initialization mode only (INITRQ=INITAK =1).

- 1 = The corresponding buffer is configured as transmit buffer.
- 0 = The corresponding buffer is configured as receive buffer or receive FIFO buffer.

#### ABTRQ — Abort Request

The CPU sets the ABTRQ bit to request that a full transmit buffer (IFLG=0) is aborted. The request is granted at the next sync pulse if the buffer is still full. The IFLG and the ABTAK flag are set when the message is aborted. The ABTRQ bit is cleared implicitly by setting of the IFLG. This bit is applicable only for transmit buffer. This bit can only be written if BFEN bit in BFPCTLBF register is set.

- 1 = Abort request pending.
- 0 = No abort request.

#### ABTAK — Abort Acknowledge

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. The message is aborted at the next sync pulse if the request was pending before the this sync pulse. The application software can use this flag to determine if the message has been sent out or has been aborted. The flag is set when message is aborted at the next sync pulse. The ABTAK flag is cleared implicitly by clearing the IFLG, i.e. the buffer has been set up again. This bit is applicable only for transmit buffer.

- 1 = The message has been aborted.
- 0 = The message has been sent out successfully.

#### LOCK — Message Buffer Lock

The bit has various functions depending on the configuration of the corresponding message buffer.

If the buffer is configured as **receive** buffer and the CPU wants to access the buffer it has to lock it first to prevent an overwrite. The locking is done by writing a “1” to the LOCK bit. Only one receive buffer must be locked at a time. The buffer is released by writing a “0” to the lock bit.

If the buffer is configured as **transmit** buffer and the CPU wants to access the buffer it has to lock it first. The locking is done by writing a “1” to the LOCK bit. Only one transmit buffer must be locked at a time. The buffer is released by writing a “0” to the lock bit. The request to lock a full transmit buffer (IFLG=0) is granted as soon as the buffer has been transmitted.

If there are buffers configured as **receive FIFO** the LOCK bit of buffer 0 is used for locking the FIFO. Writing a “1” locks the FIFO, no verifying is required. The lock controls the GETIDX. The FIFO buffer addressed by the GETIDX is “visible” in the memory map. The GETIDX is incremented when unlocking the FIFO by writing a “0” to the lock bit of buffer 0. Locking a buffer other than buffer 0 within the FIFO does not influence the FIFO window scheme.

**NOTE:** *The lock request is stored, i.e. only one write is necessary. LOCK bit is updated immediately if the Transmit buffer is Empty or Receive buffer is Full i.e. the application software should access a locked buffer after verifying the lock bit.*

#### IENA — Interrupt Enable Bit

This bit enables the corresponding buffer as interrupt source. This flag enables the setting of RXIF or TXIF flag, when the corresponding IFLG bit is set.

- 1 = The corresponding buffer interrupt (RXIF or TXIF) enabled.
- 0 = The corresponding buffer interrupt (RXIF or TXIF) disabled.

#### IFLG — Interrupt Status Flag

This flag has various functions depending on the configuration of the corresponding message buffer.

If the buffer is configured as **receive** buffer this flag indicates that the buffer is full. The status flag is cleared if a “1” is written to the bit position. The initialization mode value is “0”.

Flag set when the receive buffer is full.

Flag cleared when the receive buffer is empty.

If the buffer is configured as **transmit** buffer this flag indicates that the buffer is empty. The status flag is cleared if a “1” is written to the bit position. The initialization mode value is “1”.

Flag set when the transmit buffer is empty (i.e. has been transmitted or has been aborted).

Flag cleared when the transmit buffer is full and ready for transmit.

If the buffer is configured as **receive FIFO** buffer this flag has no meaning. It will never be set. The initialization mode value is “0”.

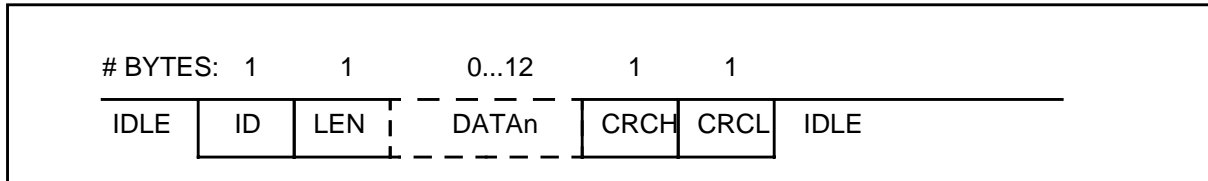


## Section 4 Functional Description

### 4.1 Byteflight Protocol

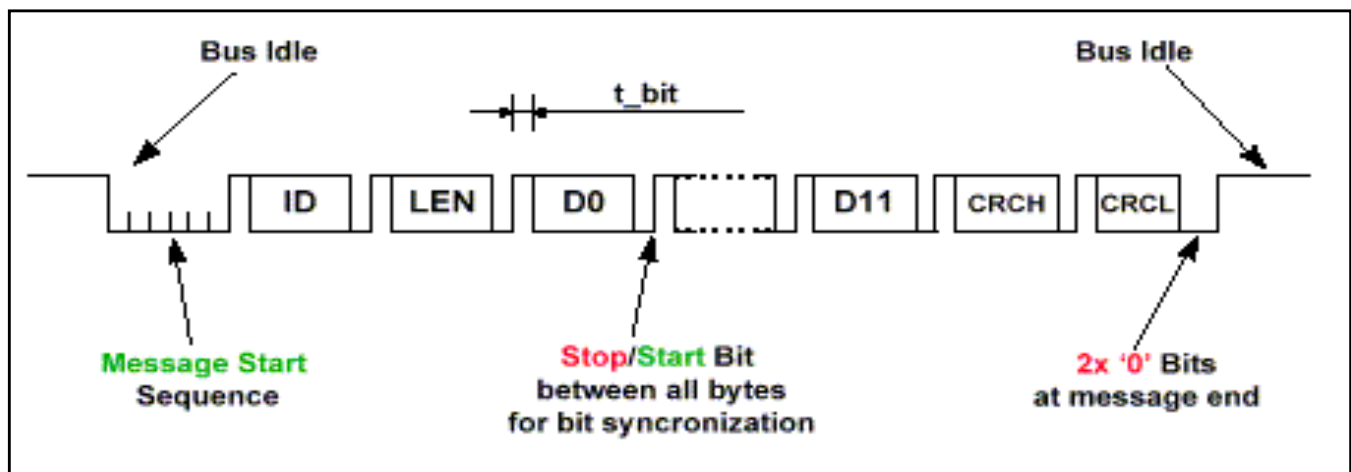
#### 4.1.1 Byteflight Format and Timing

The BMW Byteflight protocol defines the message format, as follows:



**Figure 4-1 Byteflight Message Format**

The complete message sequence is as shown below in **Figure 4-2**.



**Figure 4-2 Byteflight Complete Message Sequence**

The identifier (ID) byte is followed by the length (LEN) byte, which contains the number of data bytes (0..12) of the message. Following the data bytes (DATAn) are two CRC bytes (CRCH, CRCL), which contain the checksum. The value zero is not allowed for any identifier (ID). The priority of an identifier is inversely proportional to its value.

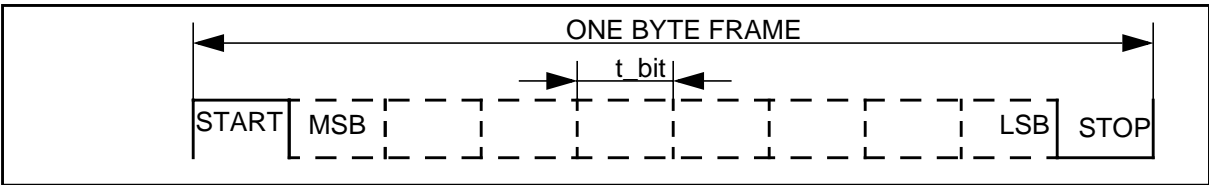


Figure 4-3 Byteflight Byte Format

One byte of a message is embedded into a start bit (“1”) and a stop bit (“0”), which are synchronization symbols and distinguish a message from a SYNC pulse. The transfer starts with the MSB first. The dominant state is represented by a logical “0” and the recessive state is represented by a logical “1”.

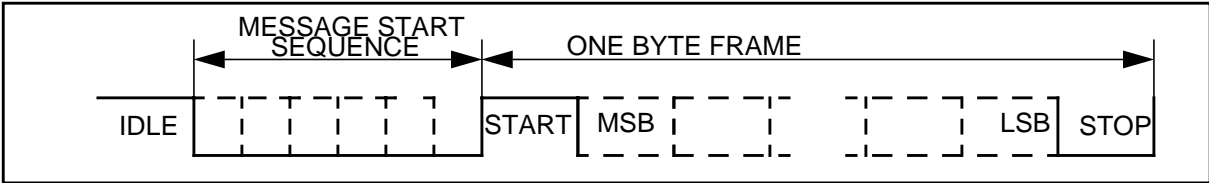


Figure 4-4 Byteflight Message Start Sequence

Every transmitted message contains an additional message start sequence of 6 bits logic “0” followed by the start bit (“1”) of the first byte. The receiver must recognize at least one bit (“0”) of the start sequence and the following start bit (“1”).

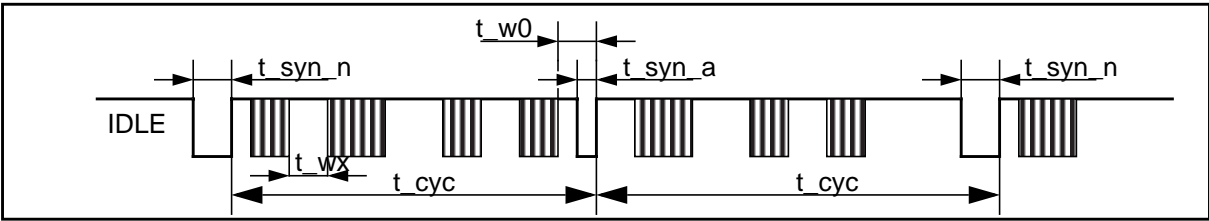


Figure 4-5 Byteflight Cycle Timing

The current bus master generates periodic synchronization (SYNC) pulses in the cycle of  $t_{cyc}$  (Table 4-1). There are two types of SYNC pulses, the normal SYNC pulse with a duration of  $t_{syn_n}$  (Table 4-1) and the ALARM pulse with the duration of  $t_{syn_a}$  (Table 4-1).

Every node can be configured as bus master by software. In the Byteflight system only one bus master should exist. All other nodes are bus slaves and use the SYNC pulses for their internal synchronization. The bus master and the bus slaves can send and receive messages in the communication cycle, which is the time between the SYNC pulses.

The order of the messages is determined by their priorities which are given by their identifiers ID (highest priority = lowest identifier). Every identifier can appear only once during a cycle. This warrants that a certain amount of high priority messages can be transferred, the bus cannot be occupied by the message with the highest priority.

Based on BFTCR1, BFTCR2, BFTCR3 (**Table 4-1**) a slot counter is running after a sync pulse was transmitted (master) or received (slave).

- Master:
  - From sync pulse:  $\text{BFTCR1} + \text{BFTCR3}$ : slot counter becomes 1
  - From sync pulse:  $\text{BFTCR1} + 2 \cdot \text{BFTCR3}$ : slot counter becomes 2
  - Form sync pulse:  $\text{BFTCR1} + 3 \cdot \text{BFTCR3}$ : slot counter becomes 3
- Slave:
  - From sync pulse:  $\text{BFTCR2} + \text{BFTCR3}$ : slot counter becomes 1
  - From sync pulse:  $\text{BFTCR2} + 2 \cdot \text{BFTCR3}$ : slot counter becomes 2
  - Form sync pulse:  $\text{BFTCR2} + 3 \cdot \text{BFTCR3}$ : slot counter becomes 3

When a node wants to transmit e.g. ID=2 and the slot counter reaches the value 2, the node starts to transmit and stops the slot counter. All other nodes receive that message in time slot=2 and also stop their slot counters as soon as they detect bus activity. When a rising edge (e.g. end of message) on the bus is detected the wait time is calculated again, i.e. the slot counter is incremented again after  $\text{BFTCR1} + \text{BFTCR3}$  (last bus activity was transmitted) or  $\text{BFTCR2} + \text{BFTCR3}$  (last bus activity was received):

Assuming that a message with ID=2 was received or transmitted in the time slot=2 the slot counter is restarted in the following way:

- Last bus activity was transmitted:
  - Waiting from rising edge:  $\text{BFTCR1} + \text{BFTCR3}$ : slot counter is incremented by one.
  - $\text{BFTCR3}$  later: slot counter is incremented again.
- Last bus activity was received:
  - Waiting from rising edge:  $\text{BFTCR2} + \text{BFTCR3}$ : slot counter is incremented by one.
  - $\text{BFTCR3}$  later: slot counter is incremented again.

### 4.1.2 Cyclic Redundancy Check (CRC)

Every message transmitted onto a Byteflight network contains two CRC bytes for error detection. These two bytes are produced by shifting the ID, LEN and DATA bytes through a preset series of feedback shift registers (*For more details refer to BMW – Lastenheft Byteflight*). The 15-bit CRC result is appended to the message following the data portion. The byte CRCH contains the upper 8 bits of the CRC result, the byte CRCL contains the lower 7 bits, the LSB of CRCL is set to “0”.

Any node which receives the message recalculates the CRC value with the same hardware and compares bitwise the result against the received CRC value. In the case of a mismatch a CRC Error is detected.

### 4.1.3 Byteflight Timing Parameters

**Table 4-1 Parameters of the Byteflight System**

Symbol	Characteristic	Description	Typ
t_bit	bit time	time for the transmission of one bit, defined by the physical limitations of the system	100 ns
t_cyc	cycle time <sup>2</sup>	time distance between the end of the SYNC pulses	250 us <sup>1</sup>
t_cyc_min	min cycle time	min time distance between the end of the SYNC pulses	249.725 us
t_cyc_max	max cycle time	max time distance between the end of the SYNC pulses	250.275 us
t_max	maximum run time of a signal in the bus system	max run time of a signal between a transmit module and the recognition by a receive module	tbd, ca 400 ns
t_tolerance	tolerance of the run time of a signal in the bus system	is the sum of all tolerances	tbd, ca 300 ns
t_idle_min	min bus idle time	min bus idle time between two messages or the SYNC pulse and a message	11 * t_bit
t_syn_a	Sync pulse ALARM	duration of the Sync pulse in the case of ALARM	2 us <sup>2</sup>
t_syn_a_min	Sync pulse ALARM, min pulse width	min duration of the Sync pulse in the case of ALARM	1.85 us
t_syn_a_max	Sync pulse ALARM, max pulse width	max duration of the Sync pulse in the case of ALARM	2.15 us
t_syn_n	Sync pulse NORMAL	duration of the Sync pulse in the normal case	3 us <sup>3</sup>
t_syn_n_min	Sync pulse NORMAL, min pulse width	min duration of the Sync pulse in the normal case	2.85 us
t_syn_n_max	Sync pulse NORMAL, max pulse width	max duration of the Sync pulse in the normal case	3.15 us
t_w0	Sync pulse wait time	minimum time between end of a message and end of SYNC pulse, $t_{w0} \geq t_{idle\_min} + t_{syn\_n\_max}$	< 6000ns
ID	Identifier	allowed values between 1 .. 255	
ID <sub>x-1</sub>	previous Identifier	Identifier of the previously received or transmitted message. When the Sync pulse was the last activity on the bus, then ID <sub>x-1</sub> = 0	
t_wx	message idle time	time between the end of the Sync pulse or end of the previous message and the start of the transmitted message. $t_{wx} \geq t_{idle\_min}$  $t_{wx} = t_{wx0\_tx} + t_{wx\_delta} * (ID - ID_{x-1})$ , when the node sent the last activity on the bus  $t_{wx} = t_{wx0\_rx} + t_{wx\_delta} * (ID - ID_{x-1})$ , when the node received the last activity on the bus	

**Table 4-1 Parameters of the Byteflight System**

Symbol	Characteristic	Description	Typ
t_wx0_tx	constant part of t_wx when last bus activity was transmitted	t_wx0_tx can be programmed individually for each node in the register BFTCR1	<1900 ns
t_wx0_rx	constant part of t_wx when last bus activity was received	t_wx0_rx can be programmed individually for each node in the register BFTCR2	<1900 ns
t_latest_tx	latest start of transmit	t_latest_tx is the maximum time for message transmission after sync pulse.	228.1 us
t_latest_rx	latest start of Receive	t_latest_rx is the maximum time for message reception after sync pulse. Every incoming message will be stopped and a message format error will be generated.	246.50 us – 246.75 us
t_start_seq_tx	length of tx start sequence	t_start_seq_tx is the maximum length of start sequence to be transmitted after message idle time.	600 ns
t_start_seq_rx	length of rx start sequence (max)	t_start_seq_rx_max is the maximum length of start sequence recognized by the receiver after message idle time.	975 ns
t_start_seq_rx	length of rx start sequence (min)	t_start_seq_rx_min is the minimum length of start sequence recognized by the receiver after message idle time.	100 ns
t_wx_delta	multiplier part of t_wx	this value must be assigned identically for each node using register BFTCR3. $t_{wx\_delta} = t_{max} + t_{tolerance}$	< 2000 ns > 200 ns
t_recognize	recognition time for bus activity	time required for recognition of bus activity: time from falling edge of receive pin until the bus controller can stop a waiting transmit process	<125 ns
t_wake_up	length of wake up pulse	a sequence of dominant and recessive pulses of length t_wake_up is sent out if WPULSE bit is set in BFMCR register	6.4 us
t_alarm_rst	Alarm Reset	Alarm bit in BFMCR register is reset after t_alarm_rst if it has not been set again by the CPU within the period.	255-256 ms

NOTES:

1.  $249.75\text{ us} \leq t_{cyc} \leq 250.25\text{ us}$  in 100% of cases there is no error,  $t_{cyc} < 249.7\text{ us}$  or  $t_{cyc} > 250.3\text{ us}$  in 100% of cases sync too early/ lost
2.  $1.875\text{ us} \leq t_{syn\_a} \leq 2.125\text{ us}$  in 100% of cases sync alarm is accepted,  $t_{syn\_a} < 1.825\text{ us}$  or  $t_{syn\_a} > 2.175\text{ us}$  in 100% of cases sync alarm is not accepted, but illegal pulse is detected
3.  $2.875\text{ us} \leq t_{syn\_n} \leq 3.125\text{ us}$  in 100% of cases sync normal is accepted,  $t_{syn\_n} < 2.825\text{ us}$  or  $t_{syn\_n} > 3.175\text{ us}$  in 100% of cases sync normal is not accepted, but illegal pulse is detected

## 4.1.4 Tolerances

The receiver is designed to handle signals on the Byteflight with the tolerances shown in **Figure 4-6** and in **Table 4-2**.

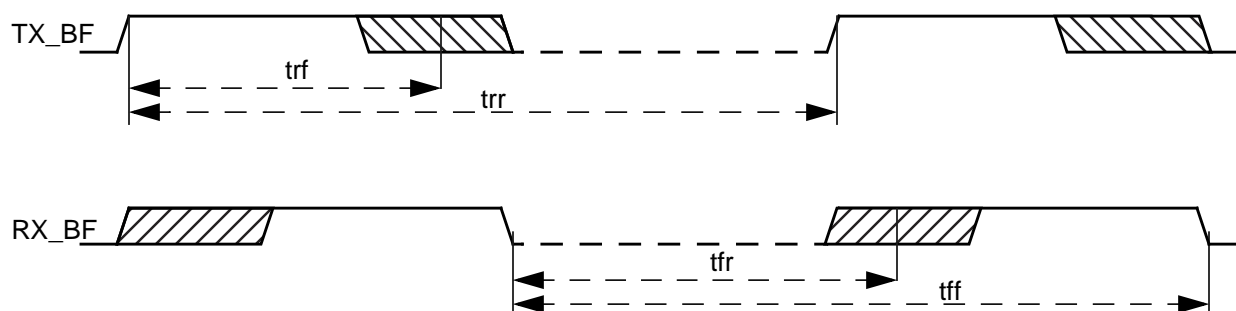


Figure 4-6 Pulse Time Tolerance

Table 4-2 Tolerances

Tolerance		Range
ttr	$n * t_{bit} \pm 5ns$	$2 \leq n \leq 10$
tff	$n * t_{bit} \pm 5ns$	$2 \leq n \leq 10$
trf	$n * t_{bit} \pm 35ns$	$1 \leq n \leq 10$
tfr	$n * t_{bit} \pm 35ns$	$1 \leq n \leq 10$

### 4.1.5 Glitch filtering

Glitches of a duration less than 25ns are filtered out.

## 4.2 Functional Overview

### 4.2.1 Receive Process

A part or all of the message buffers can be configured as a mailbox system consisting of dedicated receive buffers. The mailbox must be contiguous in the register map starting after the last FIFO buffer and ending before the first transmit buffer. The Byteflight transfers received messages from the serial message buffer to the mailbox message buffer with matching ID.

Follow the procedure described in **5.1 Initialization** to set up the buffers. In order to change the receive buffer IDs the following steps are required:

1. Enter the initialization mode by setting the INITRQ bit (BFMCR).
2. Wait until INITAK bit becomes 1.
3. Lock the message buffer (BFBUFCTL15..0) in order to appear in the Active Receive Buffer window in the memory map, LOCK bit.
4. Wait for lock acknowledge.

5. Write the matching ID to this Active Receive Buffer.
6. Clear the IFLG (buffer empty) by writing a “1” to it.
7. Unlock the message buffer.
8. Repeat from step 3 until all IDs are updated.
9. Exit the initialization mode by resetting the INTRQ bit (BFMCR).
10. Wait until INITAK bit is deasserted.

**NOTE:** *Clearing the IFLG and unlocking the message buffer must be split into two separate instructions, i.e. clear the IFLG first and then unlock the buffer.*

Once these steps are performed, the message buffer functions as an active receive buffer and participates in the internal matching process, which takes place every time the Byteflight receives a message. In this process, all active receive buffers compare their ID value to the newly received one. The buffer matching the received ID will be updated with the new message at the end of reception if the buffer is not locked. The matching buffer will be overwritten if the buffer was already full (IFLG=1). The matching buffer will not be updated if it is still locked. The buffer will be updated as soon as it is unlocked. If the buffer is locked for more than one cycle and if the same ID is received twice during this period, the buffer will be updated with the newer message as soon as it is unlocked.

The corresponding IFLG flag (buffer full) is set every time the buffer is updated, and if enabled a receive interrupt is generated. Erroneous messages will be ignored and will not overwrite unlocked full buffers. Only unique IDs must be used. ID “0” should be used to deactivate a receive buffer.

To read a receive message buffer the following steps are required, presuming the module has been initialized as described in **5.1 Initialization**:

1. Lock the corresponding message buffer (BFBUFCTL15..0) in order to appear in the Active Receive Buffer window in the memory map.
2. Wait for lock acknowledge.
3. Read the Active Receive Buffer.
4. Clear the IFLG (buffer empty) by writing a “1” to it.
5. Unlock the message buffer.

**NOTE:** *Bit manipulation instructions (BSET or BCLR) shall not be used to clear interrupt flags.*

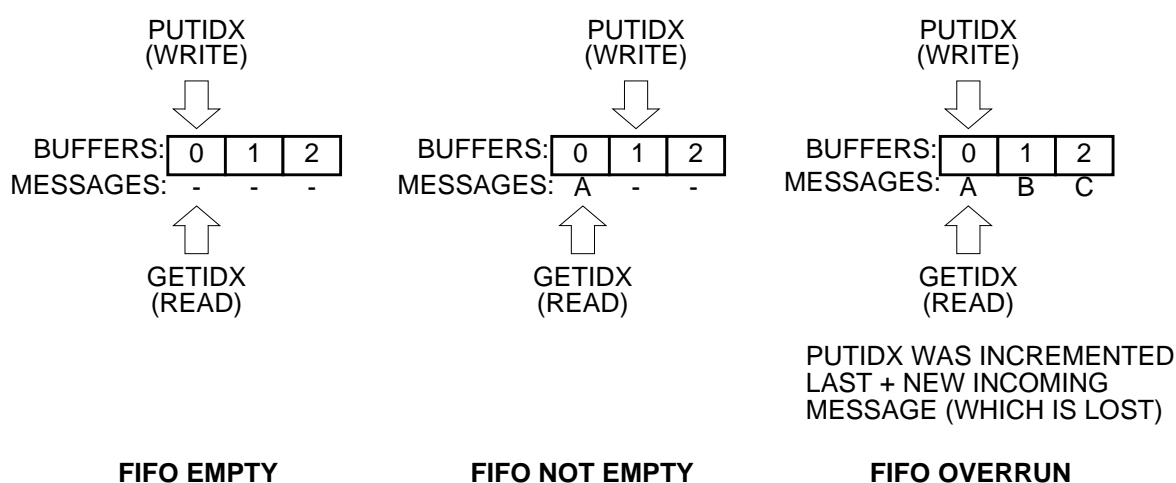
#### 4.2.2 Receive FIFO Function

A part or all of the message buffers can be configured to a Receive First-In-First-Out (FIFO) System, which can be used as logic analyzer buffers or for receiving consecutive data streams.

The FIFO starts with message buffer 0 and can be configured to the maximum of all 16 message buffers. Every incoming message not matching with any receive identifier but matching the programmable FIFO filter is stored into the FIFO buffer system. A status bit shows that the Receive FIFO is not empty, another status bit shows that a Receive FIFO Overrun has been detected. If enabled, interrupts are generated.

There are two index registers associated with the FIFO. The PUT Index Register (PUTIDX) is used as an index to the next available location in the FIFO buffer system. When a new message has been received it is written into the buffer addressed by the PUTIDX; the PUTIDX is then incremented so it addresses the next buffer. If the PUTIDX is incremented past the highest FIFO message buffer the PUTIDX is reset to 0. The GET Index Register (GETIDX) is used to address the next FIFO buffer to be read. The GETIDX is incremented when unlocking the Receive FIFO buffer. The FIFO buffer system is completely filled, when the PUT Pointer (PUTIDX) has reached again the value of the GET Pointer (GETIDX). A new incoming messages cannot be stored into the FIFO. The FIFO overrun flag is set at the end of this message if no error has occurred. A Receive FIFO non empty status is detected when the PUTIDX differs from the GETIDX. This indicates there is at least one received message in the FIFO buffer system. The PUTIDX and the GETIDX are not addressable by the CPU.

The FIFO empty, FIFO not empty and the FIFO overrun situations are explained in **Figure 4-7** below for a three buffer FIFO.



**Figure 4-7 FIFO Status (empty, not empty, overrun) – Example with 3 buffers**

To read a Receive FIFO buffer the FIFO must be locked by setting the LOCK bit of buffer 0. The message buffer addressed by GETIDX appears in the Active Receive FIFO Buffer window in the memory map. After reading the FIFO must be unlocked and the GETIDX will be incremented.

There is a programmable identifier acceptance filter for the Receive FIFO system. The FIFO identifier acceptance register (BFFIDAC) defines the pattern of the identifier to be received. The FIFO identifier mask register (BFFIDMR) specifies which of the corresponding bits are marked 'don't care' for acceptance filtering.

There is also a programmable identifier rejection filter for the Receive FIFO system. The FIFO identifier rejection register (BFFIDRJ) defines the pattern of the identifier to be rejected. The FIFO identifier rejection mask register (BFFIDMR) specifies which of the corresponding bits are marked 'don't care' for rejection filtering.



If acceptance and rejection filter are configured to match the same identifier, the message will be rejected.

**Table 4-3 Acceptance/Rejection filter mechanism – Examples**

	Example 1 <sup>1</sup>	Example 2 <sup>2</sup>	Example 3 <sup>3</sup>	Example 4 <sup>4</sup>
BFFIDAC	\$00	\$55	\$xx	%0xxxxxxx
BFFIDMR	\$00	\$00	\$FF	%01111111
BFFIDRJ	\$xx	\$55	\$0x	%xx11xxxx
BFFIDMR	\$FF	\$00	\$0F	%11001111
Stage I: Accept ID	\$00	\$55	\$00...\$FF	\$00...\$7F
Stage II: Reject ID	none	\$55	\$00...\$0F	\$30...\$3F \$70...\$7F
Receive ID	\$00	none	\$10...\$FF	\$00...\$2F \$40...\$6F

NOTES:

1. All registers in reset state: Do not accept any valid system ID (\$01...\$FF).
2. Acceptance and rejection filter set to match same ID: Do not accept ID's other than \$55, then reject \$55.
3. Accept all ID's (\$00 no valid system ID), then reject all ID's with high nibble=0.
4. Do not accept ID's \$80 to \$FF, accept all \$00 to \$7F (\$00 no valid system ID), then reject all ID's with bit 4 and 5 set.

To read a FIFO message buffer refer to the procedure in **5.2 FIFO Usage**, presuming the module has been initialized as described in **5.1 Initialization**.

### 4.2.3 Transmit Process

A part or all of the message buffers can be configured as transmit buffers. The set of transmit buffers must be contiguous in the register map starting at buffer 15 and ending before the first receive buffer. A message is submitted for transmission by unlocking the corresponding non-empty (IFLG=0) transmit buffer. The transmit buffer remains active until the message is transmitted or aborted.

If there are several messages pending for transmission within the interface, the message with the highest priority (lowest identifier) is selected next. Only transmit buffers which were unlocked and full before a sync pulse are transmitted in the following cycle. If there are two or more transmit buffers with the same identifier, the buffer with the lowest address wins the internal arbitration and will be transmitted, the remaining buffer(s) with the same ID will be removed from the arbitration process until the next communication cycle. A full buffer can be aborted by setting the corresponding abort request bit. At the next sync pulse the interrupt status flag (buffer is empty) and the abort acknowledge flag are set if the buffer has been aborted. Only the interrupt status flag is set if the buffer has been sent out.

To prepare a transmit message buffer for transmission the following steps are required, presuming the module has been initialized as described in **5.1 Initialization**:

1. Lock the corresponding message buffer in order to appear in the Active Transmit Buffer window in the memory map.
2. Wait for lock acknowledge.
3. Write to the Active Transmit Buffer (ID, LEN, DATA). If no update is needed, write is not required.

4. Unlock the message buffer and clear the IFLG (buffer full) by writing a “1” to it.

**NOTE:** *Bit manipulation instructions (BSET or BCLR) shall not be used to clear interrupt flags.*

*Use one single instruction to clear the IFLG and to unlock the buffer.*

## 4.3 Synchronization Process

The node can be configured as bus master. In this case the module generates the periodic synchronization (SYNC) pulses. If the ALARM bit is set, the master generates the ALARM pulses as long as this bit is asserted. The ALARM bit is reset after  $t_{\text{alarm\_rst}}$  (Table 4-1) if it has not been set again by the CPU within that period.

### 4.3.1 Error Handling

The Byteflight module detects and handles the following types of errors:

**Table 4-4 Error Handling**

Error Type	Error Description	Error Handling
Message Format Error	Incorrect CRC, incorrect START bit or STOP bit (Frame), missing or corrupted message start sequence, bus-activity during the bus idle time or before expiry of $(t_{\text{cyc\_min}} - t_{\text{syn\_n}})$ since the last correct sync pulse: dominant pulse of length $t$ : $t_{\text{start\_seq\_rx\_max}} < t < t_{\text{syn\_a\_min}}$	The node is still synchronized. Transmission and reception is possible after $t_{\text{idle\_min}}$ . Set flag and generate interrupt if enabled.
SYNC Pulse Too Early Error	A pulse has not the required position of an ALARM or NORMAL pulse, the pulse appears too early	SYNC pulse is used for synchronization. Set flag and generate interrupt if enabled.
SYNC Pulse Lost Error	No valid SYNC pulse detected until end of cycle $t_{\text{cyc\_max}}$	No transmission or reception until the next correct SYNC pulse. Set flag and generate interrupt if enabled.
Illegal Pulse Error	Before expiry of $(t_{\text{cyc\_min}} - t_{\text{syn\_n}})$ since the last correct sync pulse: dominant pulse of length $t$ detected on the bus: $t_{\text{syn\_a\_max}} < t < t_{\text{syn\_n\_min}}$ or $t > t_{\text{syn\_n\_max}}$ after expiry of $(t_{\text{cyc\_min}} - t_{\text{syn\_n}})$ since the last correct sync pulse: all dominant pulses which are not correct sync-normal or sync-alarm pulses lead to this error	No transmission or reception until the next correct SYNC pulse. Set flag and generate interrupt if enabled.

The receive pin of the Byteflight module is internally connected to its own transmit pin during transmitter activity and for  $8 \cdot t_{\text{bit}}$  after last transmitter activity (to avoid receiving of echoes). Therefore receiver is disabled during transmit process.

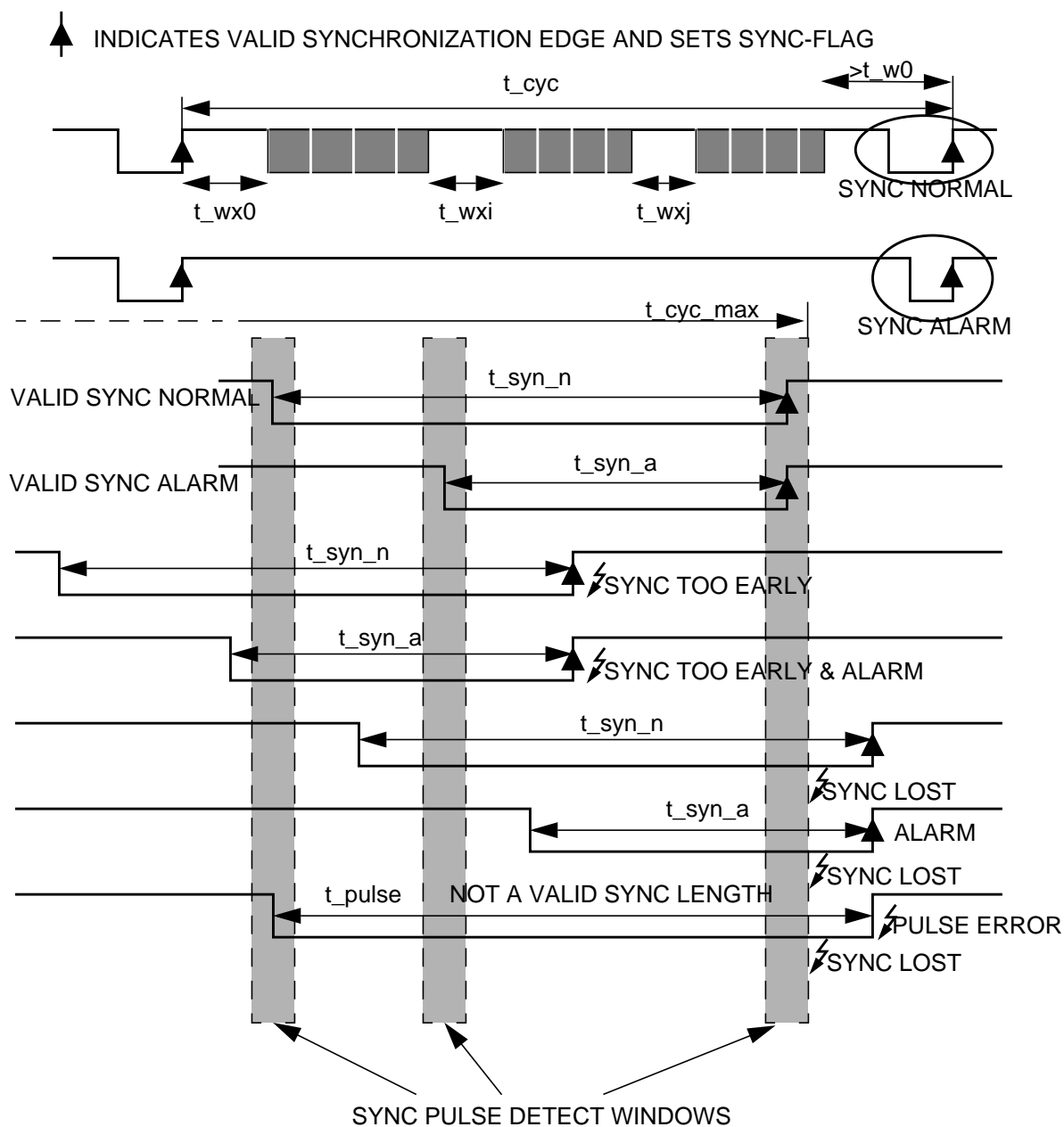
### 4.3.2 SYNC Pulse Detection

A SYNC Pulse is defined as a continuous dominant pulse with:

$$t_{\text{syn\_a\_min}} < t_{\text{syn\_a}} < t_{\text{syn\_a\_max}} \text{ or}$$

$$t_{\text{syn\_n\_min}} < t_{\text{syn\_n}} < t_{\text{syn\_n\_max}}$$

Any dominant pulse within the wait times  $t_{\text{wx}}$ ,  $t_{\text{w0}} - t_{\text{syn\_n\_max}}$  (**Table 4-1**) is considered as an error. If this pulse has a valid SYNC length the cycle is restarted and a SYNC too early error is set. If the pulse has not a valid length an illegal pulse error or a message format error is issued (see **Table 4-4**). A valid synchronization edge is detected if the pulse length is correct independent of the pulse position. If the pulse position is not correct either a pulse too early or a sync lost error condition is set. ALARM and NORMAL SYNC are distinguished only by the bits SYNAIF and SYNINF.



### Figure 4-8 SYNC Pulse Errors Detection

## 4.4 Reset Initialization

All registers reset to a particular value as indicated in **Section 3.3 Register Descriptions** of this document.

## 4.5 Interrupts

### 4.5.1 General

This section describe interrupts originated or handled by the Byteflight module. **Table 4-5** lists all interrupts generated by the Byteflight module to communicate with the CPU.

**Table 4-5 Interrupt Summary**

Interrupt	Offset <sup>1</sup>	Vector <sup>1</sup>	Priority	Source	Description
Valid Sync Pulse Interrupt			HG <sup>3</sup>	XSYNIF	Highest priority interrupt. It indicates valid SYNC pulse (ALARM or NORMAL) has been received.
Receiver FIFO Interrupt			MI <sup>2</sup>	RCVFIF	Active high level detect. It indicates FIFO not empty interrupt.
Receive Data Interrupt			MI <sup>2</sup>	IFLG[15..0] (RXIF)	Active high level detect. It indicates successfully Receive data interrupt.
Sync Pulse Interrupt			MI <sup>2</sup>	SYNAIF SYNNIF	Active high level detect. It indicates Synchronization (ALARM or NORMAL) pulse detected interrupt.
General Interrupt			MI <sup>2</sup>	IFLG[15..0] (TXIF) OVRNIF ERRIF SYNEIF SYNLIF ILLPIF WAKEIF LOCKIF SLMMIF	Active High Level detect. Can be generated asynchronously when all clocks are stopped to wake-up the CPU and transceiver.

**NOTES:**

1. Chip Dependent.
2. Maskable Interrupt.
3. High Priority Interrupt.

### 4.5.2 Description of Interrupt Operation

The Byteflight module is capable of generating five different interrupt requests (vectors) mapped onto different interrupt sources:

1. SYNC pulse interrupt at high priority (SYNC pulse NORMAL or ALARM received)
2. Receive FIFO not empty
3. Receive Interrupt
4. Synchronization Interrupt (SYNC pulse NORMAL or ALARM received)
5. General Interrupt (all errors, transmit and other interrupt sources)
  - a. Transmit Interrupt
  - b. Receive FIFO overrun

- c. Message Format Error (CRC Error, Frame Error)
- d. SYNC Pulse Too Early Error
- e. SYNC Pulse Lost Error
- f. Illegal Pulse Error
- g. Wake-Up Interrupt (can be generated asynchronously)
- h. Locking Error Interrupt
- i. Slot Mismatch Interrupt

**NOTE:** *Bit manipulation instructions (BSET or BCLR) shall not be used to clear interrupt flags.*

## Section 5 Initialization/Application Information

### 5.1 Initialization

To ensure correct operation, use the following initialization procedure:

1. Enter the initialization mode by setting the INITRQ bit (BFMCR).
2. Wait until INITAK bit becomes 1.
3. Set the Byteflight Enable bit in BFPCTLBF, BFEN bit.
4. Desired setting of module configuration register (BFMCR) – Master or Slave select, MASTER bit.
5. Desired setting of the FIFO Size register (BFFSIZR) – configure FIFO size, FSIZ4:FSIZ0 bits.
6. Desired setting of time configuration registers (BFTCR1-BFTCR3).
7. Desired setting of the message buffer control registers (BFBUFCTL15..0) – configure the buffers as transmit or receive buffers, CFG bits.

Enable/disable the corresponding interrupt, IENA bits.

8. Set FIFO acceptance/rejection register (BFFIDAC/BFFIDRJ) and corresponding mask register (BFFIDMR/BFFIDRMR).
9. Setup receive buffers:
  - a. Lock the corresponding message buffer (BFBUFCTL15..0) in order to appear in the Active Receive Buffer window in the memory map, LOCK bit.
  - b. Wait for lock acknowledge.
  - c. Write the matching ID to this Active Receive Buffer.
  - d. Clear the IFLG (buffer empty) by writing a “1” to it.
  - e. Unlock the message buffer.
10. Setup transmit buffers:
  - a. Lock the corresponding message buffer (BFBUFCTL15..0) in order to appear in the Active Transmit Buffer window in the memory map, LOCK bit.
  - b. Wait for lock acknowledge.
  - c. Write to the Active Transmit Buffer (ID, LEN, DATA).
  - d. Unlock the message buffer and clear the IFLG (buffer full) by writing a “1” to it.
11. Exit the initialization mode by resetting the INITRQ bit (BFMCR).
12. Wait until INITAK bit is deasserted.

### 5.2 FIFO Usage

The user's receive FIFO handler has to run the following procedure, assuming the FIFO is not empty:

1. Lock the FIFO buffer by setting the LOCK bit of buffer 0, the buffer with the oldest unaccessed message, addressed by GETIDX, appears in the memory map.
2. Copy over the message.
3. Unlock the FIFO buffer, the GETIDX will be incremented.

Unlocking an empty FIFO does not increment the GETIDX and does not set the FIFO empty flag.



# Index

## –I–

Initialization/application information 63



# Block Guide End Sheet

**FINAL PAGE OF  
68  
PAGES**