UCA32 LPU Reference Manual

MIL-STD-1553 Products

abaco SYSTEMS

Copyrights

MIL-STD-1553 Enhanced Universal Core Architecture (UCA32) Local Processing Unit Reference Manual (1500-095)

Firmware Revision: 6.17

Document Revision: 1.36
Document Date: 23 October 2018

Abaco Systems, Inc.
26 Castilian Drive, Suite B
Goleta, CA 93117
Main +1 805-883-6101
Support +1 805-883-6097

support@abaco.com (email)

https://www.abaco.com/products/avionics

Additional Resources

For more information, please visit the Abaco Systems website at:

www.abaco.com

Contents

Chapter 1 Introduction 1

About this Manual 1
LPU Overview 1
Definitions 2

Chapter 2 Registers 3

Introduction 3
Register Summary 4
Common Registers 6
- LPU Version 6
- LPU Build 7
- Microcode Version 7
- Microcode Build 7
- Heartbeat Register 7
- 1553 Control 7
- MIL-STD-1553A Enables 17
- Programmable Response Time-Out and Late Response Time 18
- 1553 Firmware Control 18

Interrupt Registers 19
- IQ Start of Buffer Pointer 19
- IQ End of Buffer Pointer 20
- IQ Head Pointer 20

Bus Controller Registers 20
- BC Message Initial Word Pointer 20
- BC Interrupt Enables 21
- BC Minor Frame 22
- BC Retry On Status Word 22
- BC Retry On MSW 23
- BC Retry List 24
- BC High Priority Aperiodic Message Pointer 25
- BC Low Priority Aperiodic Message Pointer 25
- BC Flags 25
- BC High Word Count 25

Remote Terminal Registers 26
- Hardwired RT Address 26

RT Address for Single RT 27
RT Enables, Bus A 27
RT Enables, Bus B 27
RT Filter Buffer Base Address 28
RT Message Initial Word Pointer 28
RT Filter Buffer Base Address 28
Broadcast Command Received and Message Error 28
Busy 28
RT Address Buffer 28
Bus Monitor Registers 33
BM Start of Buffer Pointer 33
BM End of Buffer Pointer 33
BM Head Pointer 33
BM Tail Pointer 34
BM Filter Buffer Base Address 34
BM Trigger Buffer Pointer 34
BM Interrupt Enables 34
BM Overflow Counter 35
BM Control 36
Playback Registers 36
Playback Control/Status 36
Playback Start of Buffer Pointer 38
Playback End of Buffer Pointer 38
Playback Tail Pointer 38
Playback Head Pointer 38
Playback threshold word count 38
Playback interrupt status 39
Playback interrupt threshold count 39
playback current word Pointer 39

Chapter 3 Memory 40

Memory Structures 40
BC Memory Usage 41
Aperiodic Message Lists 41
BC Message Control Block 43
BC Data Buffer 52
RT Memory Usage 56
RT Filter Buffer 57
RT Control Buffer 57
RT Message Buffer 60
BM Memory Usage 65
BM Filter Buffer 66
BM Control Buffer 67
BM Message Buffer 69
BM Trigger Buffer 73

Error Injection Buffer 79
error_injection_word (except zero-crossing) 80
error_injection_word (zero-crossing) 83
Interrupt Queue Buffer 85
interrupter_mode 86
interrupting_message_pointer 89
Playback Memory Usage 89
Playback Buffer 89
1553 playback message packet format 92

Appendix A 1553 Message Status/Interrupt Enables 93

Introduction 93

CHAPTER 1

Introduction

About this Manual

This document describes Abaco Systems' MIL-STD-1553 Local Processing Unit (LPU) used with our Enhanced Universal Core Architecture (UCA32). It is provided as a supporting document for the *UCA32 Global Register Reference Manual*.

This document is divided into three chapters:

- **Chapter 1 I**ntroduction and LPU Overview.
- **Chapter 2** describes the register interface for an LPU.
- **Chapter 3** describes the memory interface for an LPU.

LPU Overview

Abaco Systems boards may have one or more MIL-STD-1553 channels. Each MIL-STD-1553 channel is comprised of an LPU. This document describes the programming interface for a single LPU. The complete board memory map is described in the *UCA32 Global Register Reference Manual*.

The programming interface to each LPU consists of two distinct regions, a register region and a memory region. The register region and the memory region do not overlap one another.

The register region size is 1024 bytes and can only be accessed with 32 bit transfers.

The memory region is determined by the application and by physical board constraints. It is normally at least 1 Mbyte in size (the default) and is generally accessed with 32 bit transfers. 16 bit elements within the memory region may be accessed with 16 bit transfers. The memory region may be shared among multiple LPUs.

All register bits and memory bits described in this document are active "1" unless otherwise specified. Take care to only write zeros to bits specified as "Reserved", "unused" or "Undefined"

Definitions

All addresses described in this document are *byte addresses*, unless otherwise specified. Word format is *Little Endian*.

All memory pointers are 32-bit addresses with the LSB representing the byte address, unless otherwise specified.

When referencing specific bit fields, "bit 0" refers to the LSB of a byte, 16-bit word or 32-bit word. For instance, "set bit 0 and clear bits 1 through 31" would result in a hex value of 0x00000001 in a 32-bit word.

The following two-letter mnemonics are used throughout this document:

- BC - Bus Controller
- RT - Remote Terminal
- BM - Bus Monitor
- IQ - Interrupt

CHAPTER 2

Registers

Introduction

The register region consists of 1024 bytes organized as 256 32-bit words and is often referred to as “File Registers” throughout this document. Access the registers using the register base address plus the offset. The register base address is unique for each LPU on each board.

The registers are used extensively by embedded microcode when running Bus Controller, Bus Monitor, Remote Terminal or Playback operations. Many registers need to be set up before running these operations.

Exercise care when writing to registers if the BC, RT, or BM, are currently running. Writing to the board at the wrong time could cause the BC, RT, or BM to fail, thus requiring registers and memory to be setup again.

The following table summarizes the list of registers along with the address offset, and whether you may read (R) or write (W) to the register. Some registers are also referred to by a mnemonic. Do not write to registers designated as "Reserved."

Register Summary

Register	R/W	Double Word Offset	Byte Offset	Mnemonic
Common Registers				
LPU Version	R	0x10	0x40	LPU_version
LPU Build	R	0x11	0x44	LPU_build
Microcode Version	R	0x12	0x48	Microcode_version
Microcode Build	R	0x13	0x4C	Microcode_build
Heartbeat	R	0x14	0x50	Heartbeat_REG
Reserved	R	0x15	0x54	
1553 Control	R/W	0x16	0x58	CR
Reserved	R	0x17	0x5C	
MIL-STD-1553A Enables	R/W	0x18	0x60	MS1553A_en_REG
Reserved	R	0x19	0x64	
Programmable Response Time-Out and Late Response Time	R/W	0x1A	0x68	Response_time_out_REG
Reserved	R	0x1B-0x2F	0x6C-0xBC	
1553 Firmware Control	R/W	0x30	0xC0	CRF
Reserved	R	0x31-0x39	0xC4-0xE4	
Interrupt Registers				
IQ Start of Buffer Pointer	R/W	0x3A	0xE8	IQ_start_ptr
IQ End of Buffer Pointer	R/W	0x3B	0xEC	IQ_end_ptr
IQ Head Pointer	R/W	0x3C	0xF0	IQ_head_ptr
Reserved	R	0x3D-0x3F	0xF8-0xFC	
Bus Controller Registers				
BC Message Initial Word Pointer	R/W	0x40	0x100	BC_msgpts
BC Interrupt Enables	R/W	0x41	0x104	BC_interrupt_enable
BC Minor Frame	R/W	0x42	0x108	BC_minor_frame_REG
Reserved	R	0x43	0x10C	
BC Retry Enable Bits	R/W	0x44	0x110	BC_retry_on_status_word
BC Retry Enable Bits	R/W	0x45	0x114	BC_retry_on_MSW
BC Retry List	R/W	0x46	0x118	BC_retry_list
Reserved	R	0x47-0x49	0x11C-0x127	
BC High Priority Aperiodic Message Pointer	R/W	0x4a	0x128	HP_msgpts

Register	R/W	Double Word Offset	Byte Offset	Mnemonic
BC Low Priority Aperiodic Message Pointer	R/W	0x4b	0x12C	LP_msgpts
Reserved	R	0x4c-0x5f	0x130-0x17F	
Remote Terminal Registers				
Hardwired RT Address	R/W	0x60	0x180-0x183	RT_hardwired_address
RT Address for Single RT	R/W	0x61	0x184-0x187	RT_single_address
RT Enables, Bus A	R/W	0x62	0x188-0x18B	RT_enablesA
RT Enables, Bus B	R/W	0x63	0x18C-0x18F	RT_enablesB
RT Filter Buffer Base Address	R/W	0x64	0x190-0x193	RT_filter_buf_base
Reserved	R	0x65	0x194-0x197	
RT Message Initial Word Pointer	R/W	0x66	0x198-0x19B	RT_msgpts
Reserved	R	0x67-0x73	0x19C-0x1CF	
Broadcast Command Received	R/W	0x74	0x1D0-0x1D3	BCR_Reg
Message Error	R/W	0x75	0x1D4-0x1D7	ME_Reg
Reserved	R	0x76-0x77	0x1D8-0x1DB	
Busy	R/W	0x78	0x1DC-0x1DD	Busy_Reg
Reserved	R	0x79-0x7F	0x1DE-0x1FF	
RT Address Buffer	R/W	0x80-0xBF	0x200-0x2FF	RT_Options, RT_Status, RT_last_command_word, RT_BIT_word
Bus Monitor Registers				
BM Start of Buffer Pointer	R/W	0xC0	0x300	BM_start_ptr
BM End of Buffer Pointer	R/W	0xC1	0x304	BM_end_ptr
BM Head Pointer	R/W	0xC2	0x308	BM_head_ptr
BM Tail Pointer	R/W	0xC3	0x30C	BM_tail_ptr
BM Filter Buffer Base Address	R/W	0xC4	0x310	BM_filter_buf_base
BM Trigger Buffer Pointer	R/W	0xC5	0x314	BM_trig_buffer_ptr
BM Interrupt Enables	R/W	0xC6	0x318	BM_interrupt_enable
Reserved	R	0xC7-0xD0	0x31C-0x340	
BM Overflow Counter	R/W	0xD1	0x344	BM_overflow_cnt
BM Control	R/W	0xD2	0x348	BM_control
Reserved	R	0xD3	0x34C	
Playback Registers				

Register	R/W	Double Word Offset	Byte Offset	Mnemonic
Playback Control/Status	R/W	0xD4	0x350	PB_REG
Reserved	R	0xD5-0xD7	0x354-0x35C	
Playback start Pointer	R/W	0xD8	0x360	PB_start_ptr
Playback end Pointer	R/W	0xD9	0x364	PB_end_ptr
Playback Tail Pointer	R/W	0xDA	0x368	PB_tail_ptr
Playback Head Pointer	R/W	0xDB	0x36C	PB_head_ptr
Playback threshold word count	R/W	0xDC	0x370	PB_int_threshold
Playback interrupt status	R/W	0xDD	0x374	PB_int_status
Playback interrupt threshold count	R/W	0xDE	0x378	PB_thrsh_cnt
Playback current word Pointer	R/W	0xDF	0x37C	PB_ptr
Reserved Registers				
Reserved	R	0xe0-0xFF	0x380-0xFC	

Common Registers

This section defines registers common to more than one of the following MIL-STD-1553 functions: Bus Controller (BC), Remote Terminal (RT), Bus Monitor (BM) or Playback. Playback allows prerecorded missions to be played back onto the 1553 bus for additional analysis.

LPU Version (Dbl Wd 0x10, Bytes 0x40 – 0x43)

The LPU Version and Microcode Version numbers are four-digit BCD numbers.

Read Only

Bit	Field	Definition
3-0	Minor_rev_ones	Minor revision number, ones digit
7-4	Minor_rev_tens	Minor revision number, tens digit
11-8	Major_rev_ones	Major revision number, ones digit
15-12	Major_rev_tens	Major revision number, tens digit
31-16	0x0000	

Revision numbers are generally written with a period between major and minor revision numbers. The most significant digit of the Major Revision is not indicated if it is zero. For instance, "LPU V6.09" would be programmed as 0x00000609.

LPU Build (Dbl Wd 0x11, Bytes 0x44 – 0x47)

The LPU Build and Microcode Build numbers are three-digit BCD numbers.

Read Only

Bit	Field	Definition
3-0	Build_ones	Build number, ones digit
7-4	Build_tens	Build number, tens digit
11-8	Build_hundreds	Build number, hundreds digit
31-12	0x0000	

The build number reflects a recompile of the firmware and is frozen for a specific version release of a board model. Valid build numbers are from 001 through 999.

Microcode Version (Dbl Wd 0x12, Bytes 0x48 – 0x4B)

The Microcode Version definition is the same as the LPU Version (Dbl Wd 0x10).

Microcode Build (Dbl Wds 0x13, Bytes 0x4C – 0x4F)

The Microcode Build definition is the same as the LPU Build (Dbl Wd 0x11).

Heartbeat Register (Dbl Wds 0x14, Bytes 0x50 – 0x53)

A 32-bit register for each 1553 channel is incremented at an undefined rate whenever the LPU microcode is running. Failure to increment indicates that the LPU is being held in reset or that the channel is not functioning properly.

1553 Control (Dbl Wd 0x16, Bytes 0x58 – 0x5B)

This register provides control and status for BC, RT and BM. The host programs this register during system setup. The control register bits are updated with a thread-safe mechanism which can either set or clear multiple bits with a single write operation:

Control Register (write data)

Bit	Field	Definition
0	CR bit 0	update bit 0
1	CR bit 1	update bit 1
.	.	.
29	CR bit 29	update bit 29
31-30	set_clr	00 = no change 01 = clear

		10 = set 11 = illegal

Examples:

Writing 0x80008001 sets control register bits 15 and 0.
Writing 0x48100000 clears bits 27 and 20.

Note: Typically control register bits are updated a single bit at a time.

Control Register (read)

Bit	Field	Definition	Valid Modes
0	imw	invalid mode warning	BC, BM, RT
1	BC_Run	BC_Run	BC
2	RT_Run	RT_Run	RT
3	BM_Run	BM_Run	BM
4	exto	external ttl output	BC, BM, RT
5	bcb	BC_busy	BC
6	MS1553A	MIL-STD-1553A mode	BC, BM, RT
7	bex	bc enable external sync trigger	BC
8	iw	internal wrap	BC, BM, RT
9	int	CPU interrupt	BC, BM, RT
10	RT31Bcst	RT 31 broadcast	BC, BM, RT
11	sa31MC	sub-address 31 mode code	BC, BM, RT
12		Unused	N/A
13	exs	RT enable external sync trigger	RT
14	ien	CPU interrupt enable	BC, BM, RT
15	cpl	1553 bus coupling	BC, BM, RT
16		reserved for factory use	BC, BM, RT
17		reserved for factory use	BC, BM, RT
18		reserved for factory use	BM
19		reserved for factory use	BM
20	eti	enable trigger interrupt	BC, BM, RT
21	bms	BC message scheduling	BC
22	fbct	fixed BC message timing	BC
23	mic	monitor invalid commands	BM
24	uil	undefined is illegal	BC, BM, RT
25	SC_Fail	security chip fail (read only)	BC, BM, RT
26	AMFO	allow minor frame overflow	BC
27	FST	frame start timing mode	BC
28		Unused	N/A
29	sRT	RT Validation mode (read only)	RT
31		Set/Clear bits 29:0	BC, BM, RT

invalid mode warning

The invalid mode warning bit is set under the following conditions:

- The BC_Run and the RT_Run bits are turned on simultaneously on a Dual mode board.
- A monitor-only board was purchased and either the BC_Run or the RT_Run bit was turned on.
- You program the BM_Only channel configuration option bit to be active and either the BC_Run or the RT_Run bit was turned on.
- More than one of the mutually exclusive Bus_Tester, BM_Only or sRT (Single RT) mode bits are set at the same time: see the descriptions of bits 29 through 31 of the 1553 Control register. This indicates that a flash configuration problem has occurred.

When the invalid mode warning bit goes active, it disables MIL-STD-1553 operation. The operation is enabled by writing a logic "1" to the warning bit after the offending BC_Run or RT_Run bit is turned off.

To turn off the warning and enable 1553 operation in the case of multiple mutually exclusive modes set, the configuration flash must be reprogrammed with no more than one mode bit.

The invalid mode warning bit is not set when running in internal mode except in the case when multiple mutually exclusive modes are set. This is provided to be able to support the BIT function.

BC_Run, RT_Run, BM_Run

Setting these three bits causes board operation as a Bus Controller, Remote Terminal(s), and Bus Monitor, respectively. These bits may be set once the on-board RAM and associated File Registers have been initialized. Multi-function boards support all three of these modes simultaneously. Dual-function boards simultaneously supports BM and either BC or RT modes.

Note: Avoid setting any of these bits prior to setting up the board.

external ttl output

Setting this bit forces the external TTL output to the active voltage level until this bit is cleared. Clearing this bit allows the output to be pulsed to the active level for 50 microseconds when the RT receives a sync mode command or when the Bus Monitor trigger point occurs, if either of these conditions is setup. To setup these capabilities, see the description of the fire_ext_trigg_if_sync_mode_code bit of the 1553 Firmware

Control or the description of the external_output_on_trigger bit of the BM_trigger_header word in chapter 3.

BC_busy

The BC_busy bit is used by the software to determine when it is allowed to shut off the BC run bit. The microcode sets the BC_busy bit at the start of a new minor frame and clears it when it is done processing the last message in a minor frame. It automatically clears the BC_busy bit on a Stop BC. BC_busy may also be cleared after processing a Noop or Conditional message block if the end of minor frame bit is set for that message block.

Using BC_busy is not foolproof, as the hardware might set BC_busy between the time when the software tests the bit and it clears the BC_Run bit. Therefore, the software should always program all BC registers prior to setting BC_Run in case it wasn't previously shut off during the allowed time. Turning off the BC_Run bit while BC_busy is set might cause a 1553 message to be aborted mid-message, which could produce bus errors.

Note: Care should be exercised if using BC_busy while enabling aperiodic messages. The BC_busy bit can be clear while processing the aperiodic messages.

MIL-STD-1553A mode

Setting this bit makes the channel MIL-STD-1553A, which has the following differences from MIL-STD-1553B.

The t_r bit field in the command word is ignored for mode commands. RT Filter pointers for both transmit and receive mode codes must be identical.

There are no mode codes with an associated data word. The MSB of the mode code/word count field is ignored. RT Filter pointers for each mode code with the MSB of the mode code/word count field either logic zero or logic one must be identical.

A value of 11111 (31) in the sub-address field of the command word doesn't indicate a mode command. The word count field determines the number of data words.

The MIL-STD-1553A Specification defines only the Dynamic Bus Control mode code, however, the Transmit Status Word mode code is also supported.

Other than the RT Address bits, the only bits in the status word that are defined are the terminal address, message error and terminal flag bits.

BC enable external sync trigger

Setting the BC enable external sync trigger bit allows the external trigger input to start the BC operation by setting the BC_Run bit. The board must be previously set up as a BC. After sensing an active input trigger, the firmware sets the BC_Run bit. An active input trigger has no effect if the BC_Run bit is already set.

internal wrap

A complete bus simulation is possible without affecting the MIL-STD-1553 bus itself. Setting the internal_wrap bit allows the encoder/decoder to function but inhibits the transceiver chip from transmitting or receiving data from the bus, thus running autonomously from the bus, even though physically connected. In order to guarantee correct operation don't toggle the internal_wrap bit while the BC_Run or RT_Run bits are set.

CPU interrupt

The CPU interrupt bit is set by firmware when an enabled event occurs. It is cleared by the software or when the board is reset.

The host may also set the CPU interrupt bit. This bit is output as the PCI INTA signal on PCI systems.

There is a time lag between reading the interrupt and clearing it. During this lag, the hardware might have set the bit again (for the next 1553 message) before the host has cleared it. If the host reads, clears, and then interrogates the Interrupt Queue Buffer, then it shouldn't miss anything. However, this allows the hardware to set the interrupt after it has been cleared but before the software has finished reading the buffer. Thus the interrupt routine is exited and the software responds to the new interrupt, which this time has an empty Interrupt Queue buffer. (You already read all the entries the last time you serviced it.) This is not a problem as long as you're aware of the “false” interrupt (and can handle the overhead). You could clear the interrupt bit a second time after you finished reading the buffer, but then you're back to possibly missing an interrupt.

If another message, which has interrupts set, comes in the software should get that interrupt, and a “time-out” interrupt occurs every now and again. The software should be clearing the interrupt register within fifty microseconds of the interrupt occurring. This should limit the software’s exposure to clearing the second interrupt.

RT 31 broadcast, sub-address 31 mode code

If RT31Bcst is set, then the 5-bit RT address field of a command word with a value of all ones (0x1f) is interpreted as a broadcast command rather than as a command to RT 31. If sa31MC is set, then the 5-bit sub-address field of a command word with a value of all ones (0x1f) is interpreted as a mode code instead of as sub-address 31.

To comply with MIL-STD-1553B, both the RT31Bcst and sa31MC bits must be turned on (logic one). To comply with MIL-STD-1553A, both of these bits must be turned off (logic zero). For mixed MIL-STD-1553A and MIL-STD-1553B systems, both bits must be turned on to comply with MIL-STD-1553B, however, the MIL-STD-1553A Enables Register bits bypass the RT31Bcst and sa31MC bits for each RT programmed for MIL-STD-1553A operation.

RT enable external sync trigger

Setting the RT enable external sync trigger bit allows the external trigger input to start the RT operation by setting the RT_Run bit. The board must be previously set up as a RT. After sensing an active input trigger, the firmware sets the RT_Run bit. An active input trigger has no effect if the RT_Run bit is already set.

CPU interrupt enable

This register bit is logically AND'ed with the interrupt bit to produce the interrupt request signal.

1553 bus coupling

Writing a logic zero to this bit makes the bus direct coupled (no coupling transformer required). A logic one makes the channel transformer coupled. This affects both bus A and bus B of the dual redundant channel.

Unused, reserved for factory use

These bits are for factory use only and must only be programmed with logic zeros by the application.

enable trigger interrupt

Setting this bit enables trigger interrupts. After sensing an active input trigger, the firmware sets the *trigger input interrupt* bit in the interrupt queue buffer and asserts a hardware interrupt. Care must be taken prior to setting this bit to ensure that the input is not left floating and is not so noisy as to cause spurious inputs that could cause multiple interrupts.

BC message scheduling

Setting the bms bit selects the BC message scheduling option. This option allows you to set start and repeat frame values for each message in the bus list, which provides additional flexibility to either standard BC message timing or to fixed message timing.

This option allows scheduling messages on particular frames by assigning a start frame and repeat rate for each message. The repeat rate assigns a frame count which determines the frequency that the message is transmitted and the start frame determines the specific frame offset that this message occurs on.

Setting this option bit is global; that is, it requires that every message block has values programmed for the start frame and repeat rate. The repeat_rate and start_frame are 16-bit words programmed in each BC Message Block. There is a one-shot mode where the repeat rate is programmed with 0x0 then the message only goes out once (determined by the start frame entry) and never again. Note that values programmed for the start frame and repeat rate of a message block in an aperiodic list have no meaning and are ignored.

BC message sequencing may be controlled by the application in real-time, based on external events, however, the start frame value

should always be programmed before the repeat rate or else you risk invoking a one-shot message rather than repetitive message.

Below is a table showing an example of BC Message Scheduling.

		Frame	1	2	3	4	5	6	7	8	9	10	11	12	13
MSG #	Start	Repeat													
0-RT1 SA1 WC1TX	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x
1-BCST SA9 WC9	1	4	x				x				x				x
2-RT9 SA9 WC9 TX	4	0				x									
3-RT10 SA10 WC10 TX	0	0													
4-RT11 SA11 WC11 RX	0	2													
5-RT2 SA2 WC2 RX	1	2	x		x		x		x		x		x		x
6-RT3 SA3 WC3 TX	2	2		x		x		x		x		x		x	
7-RTRT4/6 SA10/6 WC6	5	5					x					x			
8 RT4 SA0 MC17	3	3			x			x			x			x	

The message scheduled bus list contains nine messages and the table charts the frame in which each message would execute (indicated by an "x") based on the start frame and repeat rate. The bus list contains BC-to-RT, RT-to-BC, broadcast, mode code and RT-to-RT messages. The following start/repeat conditions are used in this example:

Start	Repeat	
0	0	Never
0	1	Never
n	0	Once on the nth frame
1	1	Always
1	n	Every nth frame start in frame 1
m	n	Every nth frame starting on frame m

Minimum inter-message gap times are increased when using message scheduling. Tests were performed using a 12 microsecond gap when one programmed message doesn't repeat and a 15 microsecond gap when two consecutive messages don't repeat. These minimum times are not guaranteed.

Warning: Asynchronously disabling a message (see Chapter 3) and later enabling it will preserve the frame count and thus it may lose sync with the other messages in the bus list.

fixed BC message timing

Setting the fixed BC message timing (fbct) bit causes the inter-message gap time to be a fixed message time which is measured from the start of the current command to the start of the next command in the bus list.

The fixed message time is programmed with each command with a minimum of 24 micoseconds and no practical upper limit. The time associated with the last command in a frame is ignored.

Retries do not reset the message timer. Retries are generally output at an inter-message gap time of 7.0 microseconds. A No-Response time-out requires additional time before the retry is output: the programmed response time-out value plus an additional 3 microseconds to detect that there is not a valid sync and first two data bits. It is your responsibility to ensure that the command, all retried commands and the response meet the programmed timing. If it overruns, the firmware posts a Minor frame overflow interrupt (if enabled) and does not transmit the remaining messages in the minor frame. See the AMFO bit in this register and the Minor frame overflow interrupt section for more information on Minor frame overflow behavior.

monitor invalid commands

Setting the monitor invalid commands (mic) bit allows the Bus Monitor to post an interrupt when expecting a command word but receiving an invalid command. In order to be recorded, a command sync followed by two valid Manchester II data bits must be received with proper timing. The message buffer will only consist of the command word, Time Tag and error status word. Care should be used when setting this option, as noise on the bus may sometimes be interpreted as invalid commands. This option will also affect the BC and should only be used when error injection is being implemented by the BC, in which case the BC will post an interrupt entry if enabled. The mic option should never be set in an operational environment.

Note: It is recommended that the customer not set the mic bit, as it was only intended as a troubleshooting aid in a lab environment. Invalid Commands are not considered part of MIL-STD-1553 messages and noise and injected errors may be detected, however, monitored traffic may be inconsistent, and Abaco Systems is deprecating this feature.

undefined is illegal

Setting the undefined is illegal (uil) bit causes the RT to treat undefined mode codes as illegal commands (the default for undefined mode codes is invalid). The difference is that the Remote Terminal must set the message error bit if illegal, while it completely ignores invalid commands.

Feature Overview:

Some T/R, sub-address, word count combinations are not defined by the MIL-STD-1553B Spec. These commands, if they appear on the bus with the sub-address field set to mode code, are called “undefined mode codes”. They should never appear on the bus

but the system should handle them consistently and appropriately should they occur. The following command-field values apply:

1) T/R=0, mode codes 0 through 15
2) T/R=0, mode codes 16, 18, 19
3) T/R=1, mode codes 17, 20, 21

The RT Validation Specification allows the RT to process an undefined mode code in any one of four ways to pass validation:

(1) INVALID, with no ME. The RT does not respond.

(2) INVALID, with ME. The RT does not respond but sets the ME bit.

(3) ILLEGAL, the RT responds but does not set the ME bit.

(4) ILLEGAL, the RT responds and sets the ME bit.

The table below shows the behavior for the different message formats for undefined mode codes. The example requires three messages to be transmitted (steps 1, 2 and 3). Refer to the RT Validation Specification for abbreviation usage.

Command (Hex)	Undefined Mode Code Type	Invalid, not monitored	Invalid, monitored	Illegal
0x800	Rx, no data	BC: CS-NR-CS BM: CS-NR-CS RT: CS-NR-CS	BC: CS-NR-ME BM: CS-NR-ME RT: CS-NR-ME	BC: CS-ME-ME BM: CS-ME-ME RT: CS-ME-ME
0x810	Rx, 1 data	BC: CS-NR-CS BM: CS-NR-CS RT: CS-NR-CS	BC: CS-NR-ME BM: CS-NR-ME RT: CS-NR-ME	BC: CS-ME-ME BM: CS-ME-ME RT: CS-ME-ME
0xc11	Tx, 1 data	BC: CS-NR-CS BM: CS-NR-CS RT: CS-NR-CS	BC: CS-NR-ME BM: CS-NR-ME RT: CS-NR-ME	BC: CS-ME-ME BM: CS-ME-ME RT: CS-ME-ME
0xf800	Bcst Rx, no data	BC: CS-NR-CS BM: CS-NR-CS RT: CS-NR-CS	BC: CS-NR-CS BM: CS-NR-CS RT: CS-NR-CS	BC: CS-NR-CS BM: CS-NR-CS RT: CS-NR-CS
0xf810	Bcst Rx, 1 data	BC: CS-NR-CS BM: CS-NR-CS RT: CS-NR-CS	BC: CS-NR-CS BM: CS-NR-CS RT: CS-NR-CS	BC: CS-NR-CS BM: CS-NR-CS RT: CS-NR-CS

Notes:

1. For CS-NR-CS, data word for the last command in Step 3 is the Command Word from Step 1; for CS-NR-ME and CS-ME-ME, data word for the last command in Step 3 is the Command Word from Step 2.
2. The board can respond to Undefined Mode Codes per RT Validation Specification 5.2.1.1.1 (2), (3) or (4).
3. Undefined Broadcast Mode Codes treated as Illegal should be CS-NR-ME/BCR; however this result meets (3) of RT Validation Specification 5.2.1.1.1 g.
4. Abbreviations used in the table: Rx (receive command) ; Tx (transmit command); CS (Command-Status response); NR (no response); ME (message error bit set in the status word).

security chip fail

This is a read-only bit and is set when there is a Security Chip failure. When the SC_Fail bit is active it disables MIL-STD-1553 operation. Please note that this bit should never be.

allow minor frame overflow (AMFO)

Setting the AMFO bit prevents the Minor Frame Overflow interrupt from occurring when the bus list overflows the frame.

For more information on minor frame overflows, refer to descriptions of the BC Minor Frame register and the four interrupter modes for BC overflow conditions

Warning: The minimum frame time verified by Abaco Systems is one millisecond (1kHz). If the application allows for sufficient message, response time-out, inter-message gap times and enabled retries and a small amount of firmware overhead time then a shorter frame time may be used.

frame start timing mode

Setting this bit puts the Bus Controller in Frame Start Timing mode. In this mode, the inter-message_gap_time word in the BC Message Block is an offset in microseconds from the start of the minor frame. The elapsed time since the start of the minor frame is compared to the programmed time and holds off the next command until equal.

RT_Validation mode

This is a read-only bit and reflects the state of the channel configuration RTVal bit (refer to the *Customer Installer*). When set, the channel is a single RT, Validated per MIL-STD-1553 Appendix A, in addition to a Bus Monitor. When the state of this bit is changed, the RT must be reinitialized before RT_Run bit is set.

Bus Monitor only mode

This is a read-only bit and reflects the state of the channel configuration BM_Only bit (refer to the *Customer Installer*). When set, the channel is a Bus Monitor only and cannot transmit onto the 1553 bus.

MIL-STD-1553A Enables (Dbl Wds 0x18, Bytes 0x60 – 0x63)

A 32-bit register allows each RT to be independently programmed to be Mil-Std-155A or MIL-STD-1553B. Setting the bit corresponding to the five-bit RT address field in the command word enables 1553A mode. The default is 1553B mode (logic zero). This bit is logically OR-ed with the channel-wide MS1553A bit in the 1553 Control register. See the description for the

MS1553A bit for a description of the differences between Mil-Std-155A and MIL-STD-1553B

31 - 0
MS1553A_enable for RT[31:0]

Programmable Response Time-Out and Late Response Time (Dbl Wd 0x1A, Bytes 0x68 – 0x6B)

The response register must be programmed with the RT response time-out and late response time. The response time-out may be programmed with 4 (001000) through 31.5 (111111) microseconds, in half-microsecond increments. The default is 14 microseconds for 1553B. The late response may be programmed with 4 (001000) through 31 (111110) microseconds, in half-microsecond increments. The default is 12 microseconds for 1553B. These registers should not be changed when the 1553 bus is active. The response time out is initialized to 16 microseconds and the late response is initialized to 14 microseconds upon board reset.

Read/Write

Bit	Definition
5 - 0	Late response
6	0
7	0
13 - 8	Response time out
14	0
15	0
31 - 16	0x0000

1553 Firmware Control (Dbl Wd 0x30, Bytes 0xC0 – 0xC3)

Miscellaneous control and status bits used for the RT or BC.

Read/Write

Bit	Field	Definition
0	MFO_Intr_Enable	Minor frame overflow interrupt enable (BC)
3-1	Unused	
4	rtts	Load Time Tag on sync (RT)
5	Unused	
6	fet	fire external trigger if sync mode code (RT)
31-7	Unused	

Minor frame overflow interrupt enable (BC)

Set this bit to enable the Minor frame overflow interrupts. The specified Minor frame overflow interrupts are:

- Minor frame overflow;
- BC_busy bit set on minor frame overflow;
- Low priority frame overflow;
- High priority frame overflow.

See the descriptions in the interrupter_mode section in Chapter 3.

load Time Tag on sync (RT)

This bit is valid only for the synchronize without data word, synchronize with data word, broadcast synchronize without data word and broadcast synchronize with data word mode commands. If set, the internal timer is loaded with the contents of the TTCL and TTCI registers when either mode command is received. If clear (the default), then these mode codes do not alter the timer.

Caution: Implementing this feature affects the Time Tag of all RT's (even though it is not necessarily a broadcast message); it also affects the Time Tag of the Bus Monitor, as the RT and BM use the same timer.

fire external trigger if sync mode code (RT)

Set this bit to cause the external trigger output TTL signal to pulse active when the RT receives a sync mode command.

Interrupt Registers

The Interrupt registers contain pointers used to manage interrupts. The software should initialize these registers prior to setting the BC_Run, BM_Run or RT_Run bits, but should not dynamically alter the registers during 1553 operation. Refer to the Interrupt Queue Message Buffer description in Chapter 3 for interrupt entry definitions.

IQ Start of Buffer Pointer (Dbl Wd 0x3A, Bytes 0xE8 – 0xEB)

The start addresses of the memory used for interrupt queuing must be specified during board initialization. This pointer along with the “IQ End of Buffer Pointer” determines the size of the IQ Buffer. This pointer must be aligned to 32-bit word boundary.

IQ End of Buffer Pointer (Dbl Wd 0x3B, Bytes 0xEC – 0xEF)

The end addresses of the memory used for interrupt queuing must be specified during board initialization. This pointer along with the "IQ Start of Buffer Pointer" determine the size of the IQ Buffer. This pointer must be aligned to 32-bit word boundary.

IQ Head Pointer (Dbl Wd 0x3C, Bytes 0xF0 – 0xF3)

This register tracks the posting of data to the IQ Buffer. It should be initialized before any 1553 run bit is set, generally set to the same value as the IQ Start of Buffer Pointer. When bus activity or a trigger occurs, the firmware posts any enabled interrupts and increment the IQ Head Pointer for each posting. The IQ Head Pointer points to the 32-bit word address that follows the last interrupt entry. When the IQ End of Buffer is reached the firmware reloads the IQ Start of Buffer into the IQ Head Pointer in order to maintain a cyclical IQ Buffer.

The software must allocate sufficient memory for the IQ buffer to avoid an Interrupt Queue overrun. The frequency of interrupt entries are determined by the frequency of messages received on the 1553 bus and by the number of interrupt entries generated for each message. Interrupting on multiple modes (BC, RT, BM), servicing RT-to-RT messages, and interrupting on triggers also have an effect on the IQ Buffer size required.

Bus Controller Registers

These registers contain pointers and discrete bits used by the hardware. The software should initialize them, if necessary, but should not write to them while the BC is running and there is bus activity. Refer to the BC_busy bit in the 1553 Control register for how to determine if there is bus activity.

BC Message Initial Word Pointer (Dbl Wd 0x40, Bytes 0x100 – 0x103)

This pointer addresses the first 32-bit word of the next BC Message Block to be transmitted onto the 1553 bus. It must be initialized to point to the first word of the first message to be transmitted, which is normally the first message of the first major frame.

Note: The BC Message Initial Word Pointer must be programmed to point to a valid BC Message Block prior to turning on the BC_Run bit or the board will not operate correctly.

After the command word is sent to the 1553 encoder, the firmware writes over the BC Message Initial Word Pointer with the

BC_NxtMsgPtr from the BC Message Block. This happens before the inter-message gap time is applied and before transmission of the command word onto the 1553 bus. If a retry occurs, then the firmware reloads the previous message pointer into the BC Message Initial Word Pointer.

BC Interrupt Enables (Dbl Wd 0x41, Bytes 0x104 – 0x107)

The board may interrupt the host to indicate the end of all messages or select messages, such as the last message of each minor frame. The board may also interrupt the host to indicate errors from the RT's. The host programs the enable bits in this register to enable the interrupt conditions of the BC. The bits are identical to the similarly named bits in the BC_message_status word in the BC Message Block.

The BC_interrupt_enables provides a mask to use with the BC_message_status generated at the end of the message. If the enable bit and the message status bit are both set for any of the 32 bits, then an interrupt is generated. All unused bit positions must be programmed with logic zeros by the software. The bit definitions are listed in the table below. Refer to Appendix A for a complete description of these bits.

Read/Write

Bit	Field	Definition
0	hw	high word
1	Iw	invalid word
2	lw	low word
3	Is*	inverted sync*
4	Mbe*	mid-bit error*
5	2bus*	two-bus*
6	Pe*	parity error*
7	Ncd*	non-contiguous data*
8	er	early response
9	lr	late response
10	ira	bad status address
11	bus	bus A or B
12	res	Reserved
13	res	Reserved
14	res	Reserved
15	nig	no/short inter-message gap
16	em	end of message
17	bcm	broadcast message
18	rtf	RT-to-RT message format
19	rrt	Reserved

Bit	Field	Definition
20	st	Reserved
21	mc	mode code
22	no cmd*	no command detected
23	iw2	RT-to-RT message format error
24	nres2	no-response on receive command of RT-to-RT
25	Rtry	retry occurred
26	nres	no response
27	me	message error bit
28	tb	trigger begin
29	te	trigger end
30	bmo	bus monitor overflow
31	res	reserved

Note: Bits with an asterisk "*" in the Field/Definition columns cannot be used reliably for the BC; therefore, these bits should be set to logic "0".

BC Minor Frame (Dbl Wd 0x42, Bytes 0x108 – 0x10B)

This 32-bit hardware register determines the time of each minor frame. The least significant bit represents one microsecond. Values from 40 (0x28) through 4,294,967,296 (0xffffffff) may be programmed into the register. This provides for a minor frame time of from 40 microseconds to over 71 minutes. A board reset loads 40,000 (40 ms) into the register. The timer is derived from the board internal clock oscillator. The minor frame time should be programmed prior to setting the BC_Run bit. The first minor frame begins within two or three microseconds after the BC_Run bit is set.

For information on minor frame overflows, refer to descriptions of the "Allow Minor Frame Overflow" bit in the 1553 Control register and the BC Minor Frame Overflow Interrupt.

BC Retry on Status Word (Dbl Wd 0x44, Bytes 0x110 – 0x113)

The BC may retry transmission of a message if certain bits in the status word are set. This register is programmed to enable bits that the application can retry on.

The 16-bit status word is bitwise-anded with the lower 16-bits of this register. Reserved bits are masked out by the firmware. The retry is armed if the result is non-zero. For RT-to-RT messages both transmit and receive status words are bitwise-anded with the low 16-bits of this register. If either result is non-zero, the retry is armed.

The error bits from the BC_message_status are bitwise-anded with the BC_Retry_on_MSW register, except that Reserved bits are masked out by the firmware. The retry is armed if the result is non-zero.

If the retry is armed either by a status word or by the error, then a retry occurs if the retry_enable bit is set in the BC Control word of the BC Message Control Block and if the retry count is non-zero. The number of retries to transmit and the bus to transmit on is determined by the BC Retry List register.

More information about BC retries may be found with the description of the retry_enable bit in the BC Control word in Chapter 3.

Read/Write

Bit	Field	Definition
0	tf	terminal flag bit
1	res	Reserved
2	ssf	subsystem flag
3	busy	busy bit
4	bcr	broadcast command received bit
7-4	res	Reserved
8	srq	service request bit
9	instr	instrumentation bit
10	me	message error bit
11-31	res	Reserved

BC Retry on MSW (Dbl Ws 0x45, Bytes 0x114 – 0x117)

The BC may retry transmission of a message if certain error bits in the *BC_message_*status in the BC Message Block are set. This register is programmed to enable bits that the application can retry on.

The error bits *from the BC_message_status* are bitwise-anded with this register. Reserved bits are masked out by the firmware. The retry is armed if the result is non-zero.

If the retry is armed either by a status word or by the error, then a retry occurs if the retry_enable bit is set in the BC Control word of the BC Message Control Block and if the retry count is non-zero. The number of retries to transmit and the bus to transmit on is determined by the BC Retry List register.

More information about BC retries may be found with the description of the retry_enable bit in the BC Control word in Chapter 3. Refer to Appendix A for a complete description of the message status bits of the BC_Retry_on_MSW register.

Read/Write

Bit	Field	Definition
0	Res	Reserved
1	Iw	invalid word
2	Res	Reserved
3	Res	Reserved
4	Res	Reserved
5	Res	Reserved
6	Res	Reserved
7	Res	Reserved
8	Er	early response
9	Lr	late response
10	Ira	bad status address
11-12	Res	Reserved
13	Res	Reserved
14	Res	Reserved
15	Nig	no/short inter-message gap
16-21	Res	Reserved
22	Res	Reserved
23	iw2	invalid command 2 word or CMD1/CMD2 combo
24	nres2	no-response on receive command of RT-to-RT
25	Res	Reserved
26	Nres	no response
27	Me	message error bit
28-31	Res	Reserved

BC Retry List (Dbl Wd 0x46, Bytes 0x118 – 0x11B)

The API can program up to eight retries and specify using the same or alternate bus for each retry. This register uses a two bit field for programming each successive retry.

Read/Write

Bit	Field	Definition
1-0	retry_num1	First retry
3-2	retry_num2	Retry #2
5-4	retry_num3	Retry #3
7-6	retry_num4	Retry #4
9-8	retry_num5	Retry #5
11-10	retry_num6	Retry #6
13-12	retry_num7	Retry #7
15-14	retry_num8	Retry #8

31-16	0x0000	Reserved

Each two bit retry field is defined as follows:

Definition
00 - last retry
01 - retry on same bus
10 - retry alternate bus
11 - Undefined (illegal)

The bus controller sequentially looks at each two bit field to determine if and on which bus it should retry a message which did not receive a response. When a retry field of “00” is encountered, the bus controller discontinues retrying the message. The last retry field must be programmed with the “last retry” code of “00”. Clearing bits 1:0 disables all retries.

Notes:

1. The firmware doesn't limit the number of retries: it just checks for a "00" code. Thus, a "00" code must be present in the BC_retry_control word, else it retries forever. The API limits you to eight retries.

2. A two-bit entry of "11" should not be entered.

BC High Priority Aperiodic Message Pointer (Dbl Wd 0x4A, Bytes 0x128 – 0x12B)

A description of the high priority aperiodic message pointer (HP_msgpts) may be found in the section, “Aperiodic Message Lists” in chapter 3, “Memory”.

BC Low Priority Aperiodic Message Pointer (Dbl Wd 0x4B, Bytes 0x12C – 0x12F)

A description of the low priority aperiodic message pointer (LP_msgpts) may be found in the section, “Aperiodic Message Lists” in chapter 3, “Memory”.

BC Flags (Dbl Wd 0x4D, Bytes 0x134 – 0x137)

The BC Flags are used by the firmware. The register *must* be cleared by software prior to running the Bus Controller.

Read/Write

Bit	Field	Definition
0	HP_active	high priority aperiodic list started
1	LP_ active	low priority aperiodic list started
2-31	res	Reserved for factory use

high priority aperiodic list started

The high priority aperiodic list started flag is used by the firmware while it is processing a high priority aperiodic message. The flag *must* be cleared by software prior to running the Bus Controller.

low priority aperiodic list started

The low priority aperiodic list started flag is used by the firmware while it is processing a low priority aperiodic message. The flag *must* be cleared by software prior to running the Bus Controller.

BC High Word Count (Dbl Wd 0x5C, Bytes 0x170 – 0x173)

The BC Flags are used by the firmware. The register is cleared on board power cycle and is used as a troubleshooting aide to count the total number of high words that the BC receives on Transmit commands. This counter is cumulative through multiple messages.

Remote Terminal Registers

These registers contain pointers and discrete bits used by the hardware. The software should initialize them, if necessary, but should never write to them while the RT is running (RT_Run bit of 1553 Control register is set) and there is bus activity.

Hardwired RT Address (Dbl Wd 0x60, Bytes 0x180 – 0x183)

This register contains the external RT address provided to the LPU. Depending on the product, this address can come from external RT address discrete inputs, from fixed values stored in flash memory, or from some other source. If external RT addressing is not used in a particular product, parity is set to an invalid state.

Read Only

Bit	Field	Definition
4-0	hwrt ad	Hardwired RT Address (read only)
5	hwrt adp	Hardwired RT Address Parity (read only) 0=Invalid 1=Valid
31-6		Unused

Hardwired RT Address, Hardwired RT Address Parity

When the address parity (hwrt adp) is a logic one, parity is valid and the RT responds to valid commands to the correct RT address

(as read in the “hwrt ad” field) that are received on the bus. This mechanism provides for RT status response with the BUSY bit set in the time period before host software has had a chance to initialize the RT.

RT Address for Single RT (Dbl Wd 0x61, Bytes 0x184 – 0x187)

This register provides the RT address when Single RT mode is used (see sRT bit of the 1553 Control register).

Read/Write

Bit	Field	Definition
4-0	srt ad	Single RT Address
31-5		Unused

RT Enables, Bus A (Dbl Wd 0x62, Bytes 0x188 – 0x18B)

The driver initializes this register to enable the RT’s that respond to commands received on Bus A. A logic zero enables the RT; a logic one disables the RT. RT31 is only valid if the RT31Bcst bit in the 1553 Control word is clear (logic zero).

Read/Write

Bit	Definition
0	Enable for RT0
1	Enable for RT1
2	Enable for RT2
. . .	. . .
30	Enable for RT30
31	Enable for RT31

Notes:

1. For RT Validated boards only a single RT may be enabled at any given time. The API does not allow more than one RT, but for designers writing directly to these registers, care must be taken to enable only a single RT.

RT Enables, Bus B (Dbl Wd 0x63, Bytes 0x18C – 0x18F)

The driver initializes this register to enable the RT’s that respond to commands received on Bus B. A logic zero enables the RT; a logic one disables the RT. RT31 is only valid if the RT31Bcst bit in the 1553 Control word is clear (logic zero). The 32-bit word format is identical to that for “RT Enables, Bus A” (Dbl Wd 0x62).

RT Filter Buffer Base Address (Dbl Wd 0x64, Bytes 0x190 – 0x193)

This pointer is programmed by the driver to point to the base address of the RT Filter Buffer.

RT Transmit Message Initial Word Pointer (Dbl Wd 0x66, Bytes 0x198 – 0x19B)

This register points to the first word of the RT Message Buffer currently being updated, or the first word of the next RT Message Buffer if no message is currently being updated. The microcode loads this register with the pointer in the RT Control Buffer when a command is input. The software may use this register to determine what messages are safe to update. Since host processors usually cannot respond in the millisecond range, it is imperative to chain RT Message Buffers in order to update each message before the hardware overwrites it.

RT Receive Message Initial Word Pointer (Dbl Wd 0x66, Bytes 0x198 – 0x19B)

This register points to the first word of the RT Message Buffer currently being updated, or the first word of the next RT Message Buffer if no message is currently being updated. The microcode loads this register with the pointer in the RT Control Buffer when a command is input. The software may use this register to determine what messages are safe to update. Since host processors usually cannot respond in the millisecond range, it is imperative to chain RT Message Buffers in order to update each message before the hardware overwrites it.

Broadcast Command Received (Dbl Wd 0x74, Bytes 0x1D0 – 0x1D3) and Message Error (Dbl Wd 0x75, Bytes 0x1D4 – 0x1D7)

These two registers are used internally by the microcode. They should be cleared during RT initialization to ensure that unwanted BCR and ME status bits are not set at the start of an operation. The format of each word is identical to that for “Busy” (Dbl Wd 0x78: see below).

Busy (Dbl Wd 0x78, Bytes 0x1DC – 0x1DD)

This register is used to shadow the busy bit for each of the 32 RT's. They should be set or cleared by the software whenever the busy status bits in the RT Address Buffer (see below) are set or cleared.

Bit	Field	Definition
0	RT0	RT #0
1	RT1	RT #1
2	RT2	RT #2
	. . .	
30	RT30	RT #30
31	RT31	RT #31

RT Address Buffer (Dbl Wds 0x80-0xBF, Bytes 0x200 – 0x2FF)

The RT Address Buffer contains sixty-four 32-bit registers, two registers per RT. The first 32-bit register for each RT is the RT_options and the RT_status_word. The second 32-bit register for each RT contains the RT_BIT_word. The two registers are consecutive for each RT as shown in the table below.

RT Address Register		
Double-word offset	**Bits 31:16**	**Bits 15:0**
0x80	■ RT_status for RT 0	■ RT_options for RT 0
0x81	■ RT_BIT_word for RT 0	Reserved
0x82	■ RT_status for RT 1	■ RT_options for RT 1
0x83	■ RT_BIT_word for RT 1	Reserved
...	■ ...	■ ...
0xBE	■ RT_status for RT 31	■ RT_options for RT 31
0xBF	■ RT_BIT_word for RT 31	Reserved

RT_options

This word contains control options that may be programmed for the RT. All unused bits are zero.

Bit	Field	Definition
0	RT mon	RT Monitor mode
1	DBC_Enabled	Dynamic Bus Control enabled
2	unused	
3	unused	
4	BIT_from_MBuf	Built-In-Test (BIT) Option
5-6	unused	
7	reserved	reserved for API use for RT Monitor mode
8	esw	Extended status word Option
14-9	unused	

Bit	Field	Definition
15	RT off	turn RT off with dynamic bus control
31-16	RT_status_word	RT Status Word

RT_monitor_mode

This bit is set by the API to monitor the RT without simulating the RT. The RT Monitor is useful for monitoring the operation of a specific RT that is simulated elsewhere.

An RT cannot be simultaneously monitored and simulated on the same channel. However, other RTs may be simulated, and for multi-function boards, the Bus Controller and/or Bus Monitor may be simulated.

The "RT Enables Bus A" and "RT Enables Bus B" registers must be programmed to determine which bus(es) to monitor.

Dynamic Bus Control enabled

This bit is set by the API to allow the RT to respond to a dynamic bus control mode command with the dynamic mode code acceptance bit set in its status word. The mode code must also be legalized in the RT Control Buffer for this to occur. It is the responsibility of the API to clear the acceptance bit in the RT address buffer and to transfer control to the RT.

Extended Status Word Option

- When the esw bit is set, the extended status word (16-bit Word 47 of the RT Message Buffer) is bitwise OR-ed with the RT_status_word in the RT_Address_Buffer.

Built-In-Test (BIT) Option

When this bit is set, the BIT word response is from data word 0 of the RT Message buffer. Otherwise it's from the The RT_BIT_word in the RT_address_buffer. This option is for MIL-STD-1553B only; not defined for MIL-STD-1553A.

turn RT off with dynamic bus control

This bit is used by the API to determine if the RT, which has been granted control of the bus, shall continue operating as an RT. It is not used or affected by the microcode.

RT_status_word

This is the status word returned by the RT. It is programmed by the host during initialization but can be altered at any time by the software. The software must initialize this word before running

the RT by setting the RT address and clearing all other bits. Two status bits are under the exclusive control of the hardware: busy and bcr. The hardware will NOT use what is programmed in this register for those two bits, but instead will autonomously determine what the correct values need to be per MIL-STD-1553 bus requirements.

Read/Write

Bit	Field	Definition
0	tf	terminal flag bit
1	res	Reserved
2	ssf	subsystem flag
3	busy	busy bit
4	bcr	broadcast command received bit (not used – determined by hardware)
7-4	res	Reserved
8	srq	service request bit
9	instr	instrumentation bit
10	me	message error bit (not used – determined by hardware)
11-15	RT_address	RT address bits

RT address bits

This five-bit field represents RT's at address 0 through 31. There is no RT-status_word for RT 31 if the RT_31_broadcast bit in the hardware register 1553 Control register is set.

message error bit

This bit is set by the hardware after receiving a valid receive command when an invalid data word is detected; there is a data contiguity error or improper word count; or after any command word with an illegal RT sub-address t/r bit combination is detected. The hardware clears this bit after receiving the next valid command, provided none of the above error conditions exist, before sending the next command's status response, except when the next valid command is a "transmit status" mode code (for MIL-STD-1553A or MIL-STD-1553B) or a "transmit last command" mode code (for MIL-STD-1553B only). The software should initialize this bit to logic "0", but should not try to alter this bit when the bus is running.

broadcast command received bit

This bit is set by the hardware after receiving of a valid broadcast command. The Bus Controller may continue to interrogate the broadcast command received bit by transmitting "transmit status"

or “transmit last command” mode codes, which do not affect the bit. Transmitting any other non-broadcast message to the RT causes the bit to be cleared, and it is not returned in the status word. The software should initialize this bit to logic “0”, but should not try to alter this bit when the bus is running.

busy bit

This bit is set by the RT to indicate that it is unable to transfer data. The RT returns its status word with this bit set and suppresses transmission or reception of all data. Setting of this bit is application specific.

Notes:

1. The RT logs an interrupt to the interrupt queue, even when the busy bit is set.
2. If the RT has its busy bit set when it receives a broadcast command, the RT clears the BCR bit.
3. If the RT has its busy bit set when it receives a non-broadcast command, and the command is not a “transmit last command word” or “transmit status word” mode code, the RT clears the BCR bit.
4. If the RT receives an illegal command when it’s sub-system is busy, it sets both the message error (ME) and the busy bit.
5. If the RT receives a command when it is busy, it transmits that command word if the next command is a “transmit last command word” mode code and it is no longer busy.

dynamic bus acceptance

This bit is set after receiving a valid dynamic bus control mode command provided that the DBC_Enabled bit of RT_Options is set.

RT_last_command_word

When the RT detects a command word on the 1553 bus, it is stored here for use in the “Transmit Last Command Word” mode command (MIL-STD-1553B only; not defined for MIL-STD-1553A). This is an internal word used by the hardware but may be initialized by the software prior to running the RT.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT Address					t/r	Subaddress/Mode					Word Count/Mode Code				

Field	**Definition**
RT Address	RT address bits
t/r	transmit if set, receive if not set
Subaddress/mode	subaddress or mode definition bits
Word Count/MC	word count or mode code bits

RT_BIT_word

This is the 16-bit word used by the Transmit Built-In-Test (BIT) mode command. The host normally clears this word during setup and updates it during reception of an initiate self-test mode command (MIL-STD-1553B only; not defined for MIL-STD-1553A). This capability requires the application code to read the self_test bit in the message_status word or set the self_test bit in the RT_interrupt_enables word and have a self-test routine to exercise the hardware.

Bus Monitor Registers

The Bus Monitor registers contain pointers used for Bus Monitor operation. The software should initialize the registers prior to setting the BM_Run bit, but should not alter them during Bus Monitor operation.

BM Start of Buffer Pointer (Dbl Wds 0xC0, Bytes 0x300 – 0x303)

The start addresses of the memory used to store messages must be specified during Bus Monitor initialization. This pointer along with the BM End of Buffer Pointer determines the size of the BM_Message_Buffer.

BM End of Buffer Pointer (Dbl Wds 0xC1, Bytes 0x304 – 0x307)

The end addresses of the memory used to store messages must be specified during Bus Monitor initialization. This pointer along with the BM Start of Buffer Pointer determines the size of the BM_Message_Buffer.

BM Head Pointer (Dbl Wds 0xC2, Bytes 0x308 – 0x30B)

The BM Head Pointer tracks the input of data to the BM Message Buffer. It should be initialized before the BM_Run bit is set: generally the pointer is set to the same value as the BM Start of Buffer Pointer. When the BM_Run bit is set, the firmware stores the words of the messages as they are received from the MIL-STD-1553 bus, including header information for each message. The firmware maintains the BM Head Pointer to point to the 32-bit word address that follows the last memory 32-bit word location that it has written data to. When the BM End of Buffer is reached the firmware reloads the BM Start of Buffer into the BM Head Pointer in order to maintain a cyclical BM Message Buffer.

Synchronization is maintained by reading the first header word of each message, which indicates the total number of 32-bit words in the message.

BM Tail Pointer (Dbl Wds 0xC3, Bytes 0x30C – 0x30F)

The BM Tail Pointer tracks the reading of data from the BM Message Buffer. It must be initialized before the BM_Run bit is set. Whenever the firmware increments the BM Head Pointer it checks to see if it has caught up with the tail pointer. If the pointers are equal, then a BM overflow has occurred and the firmware stops storing words in the buffer and does not update the BM Head Pointer. When the head and tail pointers are no longer equal, messages will again be stored in the BM Message Buffer.

If a BM overflow occurs, then the BM_overflow_cnt register is incremented for each message. The first message to overflow causes a BM overflow interrupt to occur.

BM Filter Buffer Base Address (Dbl Wds 0xC4, Bytes 0x310 – 0x313)

This pointer is programmed by the driver to point to the base address of the BM Filter Buffer.

BM Trigger Buffer Pointer (Dbl Wds 0xC5, Bytes 0x314 – 0x317)

This pointer is programmed by the driver to point to the base address of the BM Trigger Buffer.

BM Interrupt Enables (Dbl Wds 0xC6, Bytes 0x318 – 0x31B)

The BM may post an interrupt for every message or only for messages with certain error or status conditions. The host programs the enable bits in this register to enable the conditions that post a BM message interrupt or BM trigger interrupt in the interrupt queue. The bits are identical to the similarly named bits in the BM_message_status word in the BM Message Buffer register.

The BM interrupt enables provides a mask for the BM_message_status generated at the end of the message. If the enable bit and the message status bit are both set for any of the 32 bits, then the message generates an interrupt, provided that the command is enabled in the BM Control Buffer. All unused bit positions must be programmed with logic zeros by the software. The bit definitions are listed in the table below. If all messages are to be captured then bit 16, the "end of message" bit, must be set to

a logic "1" (default). Refer to Appendix A for a complete description of these bits.

Read/Write

Bit	Field	Definition
0	hw	high word
1	Iw	invalid word
2	lw	low word
3	Is*	inverted sync
4	Mbe*	mid-bit error
5	2bus*	two-bus
6	Pe*	parity error
7	Ncd*	non-contiguous data
8	er	early response
9	lr	late response
10	ira	bad status address
11	bus	bus A or B
12	res	Reserved
13	res	Reserved
14	res	Reserved
15	nig	no/short inter-message gap
16	em	end of message
17	bcm	broadcast message
18	rtf	RT-to-RT message format
19	rrt	Reserved
20	st	Reserved
21	mc	mode code
22	no cmd*	no command detected
23	iw2	RT-to-RT message format error
24	nres2	no-response on receive command of RT-to-RT
25	Rtry	retry occurred
26	nres	no response
27	me	message error bit
28	tb	trigger begin
29	te	trigger end
30	bmo	bus monitor overflow
31	res	reserved

BM Overflow Counter (Dbl Wd 0xD1, Bytes 0x344 – 0x347)

The BM Overflow Counter is incremented by one for each message occurring after a BM overflow occurs. The counter stops counting when the overflow condition no longer exists. The

counter must be initialized to zero before setting the BM_Run bit and must be initialized to zero when servicing a BM Overflow interrupt because the counter going from zero to one is what sets the interrupt.

BM Control (Dbl Wd 0xD2, Bytes 0x348 – 0x34B)

The BM Control register is used for optional BM control features.

Bit	Field	Definition
11-0	res	Reserved
12	No_HW_Int	no hardware interrupt
31-13	res	Reserved

no hardware interrupt

Clearing this bit (default) generates a hardware interrupt for the BC message block if the message generates an interrupt in the interrupt queue. Setting this bit disables the hardware interrupt for the BC message block.

Playback Registers

Playback Control/Status (Dbl Wds 0xD4, Bytes 0x350 – 0x353)

This register provides control and status for the MIL-STD-1553 playback operation. Playback allows prerecorded missions to be played back onto the 1553 bus for additional analysis.

The playback control/status register bits are updated with a special thread-safe mechanism which can either set or clear multiple bits with a single write operation:

Write

Bit	Field	Definition
0	bit 0	update bit 0
1	bit 1	update bit 1
.	.	.
29	bit 29	update bit 29
31-30	set_clr	00 = no change 01 = clear 10 = set 11 = illegal

Examples:

Writing 0x80000020 sets bit 5.

Writing 0x40000008 clears bit 3.

Read

Bit	Field	Definition
0	PB_start	software playback start
1	PB_inten	software playback interrupt enable
2	PB_run	hardware playback run
3	PB_err	hardware playback error
4	PB_empty	hardware playback buffer empty
5	PB_halt	software mid-playback halt
6	PB_abort	software mid-playback abort
31-7	res	Reserved

software playback start

Setting this bit indicates to the microcode that playback mode is desired. This bit is cleared by the microcode when an end of playback bit is encountered in the Message Code or the halt bit is asserted while playback is running.

software playback interrupt enable

Setting this bit enables playback hardware interrupts to be generated when the PB_thrshld_cnt File Register reaches a count of zero. Currently, the microcode only asserts the interrupt; there is no entry made for it in the interrupt queue.

hardware playback run

This bit is set by the microcode when it detects the playback start bit has been set by the host and branches to the playback routine for the first time. This bit is cleared by the microcode when an end of playback bit is encountered in the Message Code or the host asserts the halt bit while playback is running.

hardware playback error

This bit is set by the microcode when it is expecting to receive a playback gap time or a playback message code but receives neither. This bit is cleared by software.

hardware playback buffer empty

This bit is set by the microcode when the PB_tail_pointer File Register value is equal to PB_Head_pointer File Register value. This bit is cleared by software.

software mid-playback halt

This bit is asserted by the host to halt playback while it is running. Playback halts as soon as the current playback message is completed.

software mid-playback abort

This bit is asserted by the host to halt playback while it is running. Playback halts *immediately*, without waiting for the current playback message to be completed.

Playback Start of Buffer Pointer (Dbl Wds 0xD8, Bytes 0x360 – 0x363)

The start addresses of the memory used for playback must be specified during board initialization. This pointer along with the Playback End of Buffer Pointer determines the size of the Playback Message Buffer.

Playback End of Buffer Pointer (Dbl Wds 0xD9, Bytes 0x364 – 0x367)

The end addresses of the memory used for playback must be specified during board initialization. This pointer along with the Playback Start of Buffer Pointer determines the size of the Playback Message Buffer.

Playback Tail Pointer (Dbl Wds 0xDA, Bytes 0x368 – 0x36B)

This register contains the current location within the buffer where the firmware is extracting data to be processed with eight-word resolution. At the start of playback, it is initialized by the host at the same address as the PB_start_pointer.

Playback Head Pointer (Dbl Wds 0xDB, Bytes 0x36C – 0x36F)

This register contains the current location within the buffer where the host is adding data to the buffer to be processed.

Playback threshold word count (Dbl Wds 0xDC, Bytes 0x370 – 0x373)

This register contains a count of words. Each time the firmware processes this number of words, it signals the host and increments the PB_int_status register.

Playback interrupt status (Dbl Wds 0xDD, Bytes 0x374 – 0x377)

This register contains the count of the number of times the firmware has processed the “playback threshold word count” number of words.

Playback interrupt threshold count (Dbl Wds 0xDE, Bytes 0x378 – 0x37B)

This register is initialized with the value held in the playback threshold word count (PB_int_threshold) register. Each time the playback current word pointer (PB_ptr) is incremented, this value is decremented. Upon reaching a count of zero:

- Playback interrupt status (PB_int_status) has its value incremented by one.
- If the Playback Int Enable bit is set in the playback control register, a hardware interrupt is generated.
- This register is reinitialized with the value held in the playback threshold word count (PB_int_threshold) register.

playback current word Pointer (Dbl Wds 0xDF, Bytes 0x37C – 0x37F)

This register is used to point directly at the playback message buffer in RAM. At the beginning of playback, this value is initialized by the firmware as the playback start pointer. The host should not write to this register.

CHAPTER 3

Memory

Each board has a region of physical memory which is shared among all 1553 channels on the board. The application may use this RAM in any manner desired within the constraints outlined in the detailed description that follows.

RAM allocation is performed by the application. All structures may be allocated anywhere in memory. Refer to the BusTools/1553-API manual for the structures used by the API.

Note that some boards may access memory through a paging register method rather than directly.

Memory Structures

Below is a summary of each type of memory structure:

Structure	Size
BC Message Control Block	36 consecutive bytes per message
BC Data Buffer	92 consecutive bytes per message
RT Filter Buffer	8K bytes
RT Control Buffer	16K bytes maximum
RT Message Block	96 consecutive bytes per message
BM Filter Buffer	8K bytes
BM Control Buffer	16K bytes maximum
BM Message Buffer	variable depending on message size
BM Trigger Buffer	92 bytes
Error Injection Buffer	programmable
Interrupt Queue Buffer	programmable
Playback Buffer	programmable

These buffers may be located anywhere in memory that isn't allocated to other functions. They must begin on a 32-bit word boundary unless otherwise specified. The API takes care of memory allocation.

All memory pointers, whether stored in registers or RAM, are 32 bits and are byte offsets.

BC Memory Usage

Each BC message requires a BC Message Control Block and a BC Data Buffer.

Each BC Message Control Block contains sufficient information to either transmit a 1553 message, control the flow of subsequent messages or is a Noop. Each BC Message Control Block also contains a pointer to the next message block. This linked list approach provides significant flexibility in programming major and minor frames. A major frame is the slowest repetition rate for transmitting a sequence of messages to the bus and may contain numerous minor frames of a faster repetition rate. The bus may also be operated in a one shot mode by invoking a “Stop BC” message block. Conditional message sequencing and disabling messages is also provided.

Each BC Message Control Block also contains a pointer to a BC Data Buffer. For BC Messages, the data pointer may be updated by the firmware to enable a circular linked list of data values. The data buffer list may be made of sufficient length for the application to acquire 1553 transmit data before it is overwritten by the firmware or to update receive data after it is transmitted. Once the BC Message Blocks and BC Data Buffers are set up and the BC Message Pointer Register is set up, the BC Run bit may be set, which begins bus transmission. The hardware updates the BC Message Pointer Register at the start of each message.

Before setting the BC_Run bit in the 1553 Control register, the MS1553A, RT 31_broadcast and sub-address 31 mode code bits should be initialized; the Minor_Frame_REG, Response_Time_Out_REG, BC Interrupt Enable (BC_interrupt_enable) and interrupt queue pointers should be set up and the pertinent 1553 Firmware Control should be reset. If not using retries, then the BC Retry List register should be set to all zeros. If enabling retries, then the BC Retry List register should be programmed with valid retries, along with the BC_Retry_Bits and BC Flag Register. Refer to Chapter 2 for descriptions of each of these registers. Data is stored in the BC data buffer for RT-to-RT messages. This is not necessary, but prevents confusion over the fact that the BC data buffer is sometimes empty for real-time message buffer viewing.

Aperiodic Message Lists

The capability exists to insert an aperiodic list of messages in a regular frame sequence. The software sets up the list of messages, then programs a pointer to the first message in the list, which informs the hardware to process the list now. The

hardware sequences through the list and zeros the pointer out when done. A list may be as small as a single message.

There are two types of aperiodic message lists: high priority and low priority. High priority messages are sent to the bus immediately, whereas low priority messages must wait until the bus is not active until they are sent. There are separate File Registers for high and low priority messages to point to the first message of the list. High and low priority lists may not be programmed simultaneously.

Aperiodic message lists may contain linked BC Message Blocks that could be any combination of BC messages, Noops, conditional sequencing control, or a stop BC command, just like a regular BC message list.

High Priority

The high priority aperiodic message pointer (HP_msgpts) is loaded by the API with the address of the first message of a list of high priority aperiodic messages. The firmware, when detecting that this register contains a non-zero value, switches to processing the messages pointed to by this register. The current message and all retries of the current message are completed before the switch to the aperiodic list takes place. It returns to the regular message list when detecting a next message pointer value of zero. When all of the specified messages have been processed, the firmware loads a value of zero into the HP_msgpts register, which serves to terminate the mode and indicate to the API that the aperiodic messages have all been processed. The firmware uses this register to step through the list.

If there is not sufficient time to process the high priority list *and* the regular messages in a minor frame, one or more of the regular messages is not transmitted. Frame timing is thus preserved at the expense of possibly losing messages for the given frame. If the high priority list isn't finished by the end of a minor frame, the remaining messages are transmitted on the subsequent frame(s) until the list has been completely transmitted. The hardware compares the time required to transmit each message with the remaining time in the frame to determine if the message is transmitted in the current frame or the next frame.

Low Priority

The low priority aperiodic message pointer (LP_msgpts) is loaded by the API with the address of the first message of a list of low priority aperiodic messages. After processing the last message in the frame, determined by the presence of the "end minor frame" bit in the BC Control word, the firmware polls the LP_msgpts

register. If the value is non-zero, and the frame timer indicates that enough time remains in the minor frame, the firmware transmits each message in the list of messages addressed by the LP_msgpts register. If any messages in the low priority list remain that don't get processed in the minor frame, this process resumes at the end of the normal message sequence in the next minor frame. The firmware uses the LP_msgpts register to step through the list of messages, and sets its value to zero to indicate that the list of messages has been completed (e.g., when a next message pointer of zero is detected).

BC Message Control Block

The BC Message Control Block is a 36-byte buffer containing a 1553 command word (BC Message), message flow control for a BC message list (Condtional or Stop BC) or a Noop. Each BC Message Control Block contains a pointer to a BC Data Buffer and another pointer to the next BC Message Control Block in a circular linked list.

The BC_Control word determines if the BC Message Control Block is a BC Message, Conditional, Stop BC or Noop, with the format defined in the following table.

32-bit Word	BC Message	Conditional	Stop BC	Noop
0	messno	messno	messno	messno
1	Num_data_buffers	Num_data_buffers	Num_data_buffers	Num_data_buffers
2	BC_Control	BC_Control	BC_Control	BC_Control
3	BC_CmdWord1 & BC_CmdWord2	reserved	reserved	reserved
4	BC_EI_Ptr	reserved	reserved	reserved
5	BC_Data_Ptr	BC_Data_Ptr	BC_Data_Ptr	BC_Data_Ptr
6	BC_Inter-message_Gap	reserved	reserved	BC_Inter-message_Gap
7	BC_Msg_Scheduling	BC_BranchMsgPtr	reserved	reserved
8	BC_NxtMsgPtr	BC_NxtMsgPtr	BC_NxtMsgPtr	BC_NxtMsgPtr

messno

The message number is only for the API: the microcode never programs or uses this 32-bit word.

num_data_buffers

This 32-bit word is used by the API to store the number of data buffers allocated for this message: the microcode never programs or uses this 32-bit word.

BC Message

The message programmed here is output to the 1553 bus when the BC_Run bit is set and the message entry is sequenced to in the BC message list.

Conditional

Executing a Conditional causes the microcode to test a condition and load the BC_BranchMsgPtr into the BC message pointer register if the condition is true. If it is false, it loads the BC_NxtMsgPtr into the BC message pointer register. The condition tested is the exclusive-or of the data_pattern with the word to be tested (pointed to by the test_word_address), then logical-and'ed with the bit_mask (see the BC Data Buffer description). The equality status is interpreted as the *true* condition. For instance, if you wanted to branch on the busy bit (bit 4) set in the RT status word, then the test_word_address points to the RT's status word, bit 4 is set in the data pattern and the bit_mask contains 0x0010.

Note: Insertion of a Conditional BC Message Control Block may limit the inter-message gap time to at least 10 microseconds. When using the relative timing mode, for predictable timing, program the inter-message gap time to at least 15 microseconds in the message preceding the conditional.

Stop BC

Executing a Stop BC indicates that all messages are complete and the frame execution halts. The microcode turns off the BC_busy and BC_Run bits in the 1553 Control register. The BC_NxtMsgPtr is loaded into the BC_msg_pointer register by the firmware so that the BC may be turned on again without initializing this register.

Noop

Executing a Noop causes the microcode to load the next message pointer into the firmware message pointer register without processing a MIL-STD-1553 message.

An interrupt may be enabled for the Noop and the "start minor frame" and "end minor frame" bits are valid and must be set if the Noop is at the start or end of a frame.

A Noop may also be inserted to add to or to adjust message timing (see the description of the Timed Noop bit).

Note: A Noop may be converted to a message at any time, although timing may not preserved if the Noop is implemented as a Timed Noop or if there is insufficient time to process the message in the current frame. A message shall not be converted to a Noop when the BC_busy bit in the 1553 Control register is set.

A Noop not implemented as Timed Noop following a BC Message in the bus list may be executed prior to the completion of the message that precedes the Noop. An interrupt posted for the Noop will be posted prior to the interrupt for the message that preceded the

Noop in the bus list. This may be true for multiple consecutive Noops following a BC Message

BC_Control

The BC_Control word contains information passed between the API and the microcode on a message-per-message basis

Bit	Field	Definition
0	msg_disabled	Disabled Message Block
2-1	Msg_Ctrl [1:0]	message control bits
3	mnfr	start minor frame
4	Msg_Intr	interrupt on this message
5	RT_RT_msg	RT-to-RT message format
6	res	Reserved
7	emf	end minor frame
8	BC_bus	select bus
9	res	Reserved
10	Retry_enable	retry enable
11	res	Reserved
12	No_HW_Int	no hardware interrupt
13	BC_Halt	enable BC Halt
14	tnop	Timed Noop
15	rapi	reserved for use by the API (see BusTools API documentation)
31-16	BC_Buf_Num	BC Buffer Number

Disabled Message Block

This bit determines if the BC message, Stop BC, or Conditional is to be ignored. The bit is active low: clearing the bit turns the BC Message Block into a Noop (refer to the description of a Noop).

The purpose of this bit is to allow you to turn on or off a BC message, Stop BC, or Conditional by flipping a single bit. Words 2 through 20 of the BC Data Buffer must be programmed with appropriate values before changing a Noop back to either a BC Message or to a Conditional.

message control bits

These two bits determine the function of the BC Message Control Block (described earlier in this document). They are determined as follows:

bits 2-1:

00	Noop
01	BC Message
10	Stop BC or Halt BC
11	Conditional

start minor frame

The start minor frame bit is used to synchronize bus traffic to the minor frame timer, set the BC_busy bit and detect minor frame overflow errors.

If using the board's hardware timer for frame generation, the start minor frame bit *must* be set in the first message block of each minor frame. It *must* be cleared for all other message blocks in each minor frame. It must also be cleared for all aperiodic messages.

If generating frame timing externally by starting and stopping each frame, the start minor frame bit should not be set. Externally generated frame timing is more complicated to generate and is not the recommended method.

interrupt on this message

The interrupt_on_this_message bit allows user to select specific messages to post interrupts. This bit is normally a logic one (default), which enables the interrupt for this message.

RT-to-RT message format

This bit must be set by the software if the information transfer format for this BC Message is a remote terminal to remote terminal (RT to RT) transfer. The bit is ignored for Conditional, Stop BC, and Noop message blocks.

end minor frame

This bit is used by the microcode to clear the BC_busy bit once the BC Message Control Block has been processed. This bit must be set for the last message block of each minor frame in a regular message list (including a Noop) and must be cleared otherwise. It must not be set for aperiodic messages. This bit is ignored by Stop BC message blocks but should be set if the Stop BC is ever disabled (refer to the description of a Disabled Message Block).

select bus

This bit selects which bus (A or B) on which to send the message. Bus A is 0 and bus B is 1. This bit is ignored for Conditional, Stop BC, and Noop message blocks.

retry enable

The retry_enable bit allows the application to enable retries on a per message basis. The retry_enable bit must be set for a retry to occur. In addition, to retry on MIL-STD-1553 status word or on message error bits, the corresponding bits of the "BC Retry on Status Word" or "BC Retry on MSW" registers must be set.

When one or more retries occur, the "retry occurred" status bit is set in the BC_message_status word.

The number and type of retries are programmed in the BC_retry_cntrl register.

For Relative Timing mode, the inter-message gap preceding each retry is seven microseconds. For Fixed Message Timing and Frame Start Timing modes, the inter-message gap preceding each retry is determined by microcode execution time and is at least four microseconds but generally no more than eight microseconds.

Careful analysis must be performed at the system level to ensure that there is sufficient time to process retries before the minor frame ends and to provide appropriate error recovery should this situation occur.

The data that is stored in the BC Data Buffer will be that of the last retry.

The retry_enable bit must be clear for Conditional and Stop BC message blocks.

no hardware interrupt

Setting this bit disables the hardware interrupt for the BC message block. Clearing this bit sets a hardware interrupt for the BC message block if the message generates an interrupt in the interrupt queue.

enable BC Halt

This bit is only applicable when the message_control bits are "10" (Stop BC or Halt BC). If the BC_Halt bit is set then it clears the BC_run bit and the BC_trigger, allowing for a retrigger on an external pulse. In addition, the discrete output register bits are cleared.

tnop (Timed Noop)

Setting the tnop bit in conjunction with programming the message control bits as a Noop will implement a “Timed Noop.” A Timed Noop is used to extend the inter-message gap time following the preceding message before the next message in the bus list is transmitted. The inter-message gap time may be programmed from 50 microseconds to over 16 seconds.

Timed Noops are also used as "place holders" in case a message is not output on every frame, so that the timing can be preserved. It is recommended to use Fixed Message Timing if strict timing is required.

It is not recommended to use multiple Timed Noops between messages as a Timed Noop does not wait for the previous Timed Noop to finish: so the next message may be output earlier than expected. A single Timed Noop should provide sufficient time for any application.

The first message in a frame occurs a few microseconds after a frame tick. Placing a Timed Noop at the start of a frame adds approximately 5.0 microseconds more than the programmed inter-message gap time indicates, due to microcode execution time. This can be offset by programming 5 microseconds less than the desired time for a Timed Noop in the first message in a frame only.

There are two types of a Timed Noops, static and dynamic, and they have different usage rules. In the API, a static Timed Noop is enabled by programming BC_CONTROL_Timed_NOP for a message write, and a dynamic Timed Noop is programmed by programming the NoopFlag to Timed_Noop for a BC_MessageNoop instruction.

This table shows the BC timing modes where you can use the static TNOP and dynamic TNOP.

	Relative Time (default)	**Fixed Gap Timing**	**Frame Start Timing**
static TNOP	Yes	Yes	No
dynamic TNOP	No	Yes	No

In relative time, the programmed gap time is from the end of the message to the start of the next message, and therefore dynamically disabling the message with Timed Noop set, would produce a much shorter time delay than was intended.

In frame start timing mode, the time relative to the start of frame is already programmed into the gap time word, so there is no need for a Timed Noop; and, if programmed incorrectly, could alter the bus list.

We do not recommend using a Timed Noop with the message scheduling option. If you do, you must perform a thorough analysis for each unique frame in order to obtain the desired results.

BC Buffer Number

The API uses this 16-bit field as a BC Message Control Block index. Up to 65,536 BC Message Control Blocks may be generated if memory would support it.

BC_CmdWord1 and BC_CmdWord2

BC_CmdWord1 is the 16-bit command word, programmed by the software. In the case of an RT-to-RT transfer, this word is the receive command word.

BC_CmdWord2 is the 16-bit transmit command word of an RT-to-RT transfer, programmed by the software. It must have the same word count as command_word1 per MIL-STD-1553A/B specification. It is ignored by the firmware if it is not an RT-to-RT transfer.

31-27	**26**	**25-21**	**20-16**	**15-11**	**10**	**9-5**	**4-0**
BC_CmdWord*2*				BC_CmdWord1			
RT Address	t/r	Subaddress /Mode	Word Count/Mode Code	RT Address	t/r	Subaddress /Mode	Word Count/Mode Code

Field	**Definition**
Word Count/ Mode Code	word count or mode code bits
Subaddress/ Mode	subaddress or mode definition bits
t/r	transmit if set, receive if not set
RT Address	RT address bits

BC_EI_Ptr

This is a pointer to the BC error injection buffer used for this message. See the description for the RT_error_inject_pointer.

BC_Data_Ptr

The BC_Data_Ptr points to the first word of the next data buffer to be used by the BC message Control Block. For BC Message data buffers there is a circular linked list of data buffers. For Conditional, Noop and Stop BC Message Control Blocks, the pointer is always programmed to point to the current buffer. The maximum number of data buffers is only limited by the available memory.

For BC Message and Conditional BC message Control Blocks, the firmware updates BC_Data_Ptr with the BC_next_data pointer from the BC Data Buffer as it is being processed. The firmware does not alter the BC_Data_Ptr for Noop and Stop BC Message Control Blocks.

BC_Msg_Scheduling

These two 16-bit words, start_frame and repeat_rate, are only used if the BC Message Scheduling option is selected. They represent binary count values. Refer to the description for the BC Message Scheduling bit in the 1553 Control Word in Chapter 2.

Bit	Definition
15-0	repeat_rate
31-16	start_frame

BC_Inter-message_Gap

This 32-bit word is used to determine when to transmit the message.

Bit	Definition
23-0	Message time
31-24	reserved

The 24-bit inter-message gap time has a one-microsecond resolution: therefore, the maximum allowable value is greater than 16 seconds. The reserved bits should always be programmed with all zeros. The meaning of the inter-message_gap is determined by the timing mode.

There are three timing modes: relative (message-to-message) timing mode, start frame timing mode and fixed BC message timing mode. The start frame timing and fixed BC message timing modes are selected by setting bits in the 1553 Control register. These descriptions also apply to the BC message scheduling option, however, successive messages may differ on different occurrences of each frame depending on the repeat_rate and start_frame values assigned.

When determining the gap time, consider the possibility that the RT may respond late.

1. relative timing mode

For relative timing, the inter-message gap time is measured from the mid-parity bit time of the last word of the current message to the mid-sync bit of the command word of the next message.

A minimum gap time of 50.0 microseconds is validated by Abaco Systems, however, you may attain inter-message gap times as low as 4.0 microseconds under certain conditions. The bus topology may skew the selected gap time by hundreds of nanoseconds and you may need to measure actual gap times for their specific system.

Noop and conditional message blocks inserted between messages, may increase the minimum inter-message gap time due to firmware processing time. In this case, you may need to increase the minimum time programmed to generate deterministic inter-message gap times.

For a no-response time-out, allow time for the programmed response time-out time (default is 14 microseconds) plus an additional 3 microseconds before the gap timer starts counting for the next message. Using the default of 14 microseconds, this time is always less than if the RT responds with a valid status word.

For retries, a 7 microsecond inter-message gap time is inserted before each retried message.

Note: The default message timing mode for UCA uses an inter-message gap time that is not completely deterministic. The encoder uses a 2 MHz clock tick to output the command/data words on a half-bit interval. The inter-message gap timer counts down to zero after both the encoder is idle (BC is done) *and* the decoder is reset to await the next command (RT response is done). The response is a variable with respect to the 2 MHz clock tick and therefore limits the inter-message gap accuracy as it is not synchronized to the 2 MHz clock tick. If predictable inter-message gap timing is desired, then use the *Fixed Message Timing* mode.

2. start frame timing mode

In start frame timing mode, the inter-message gap is measured from the start of the current frame to the start of the command word in this BC Message Control Block. The first message in a frame cannot begin transmitting until about 2 or 3 microseconds after the frame tick and is not consistent from frame-to-frame. To ensure exact relative timing from frame-to-frame for the first message, enable transmission at ten microseconds. The message will start to be transmitted at the programmed “inter-message gap” time, or later if delayed by previous retries or high-priority aperiodic messages. Sufficient time should be budgeted for all possible retries.

3. fixed BC message timing mode

In fixed BC message timing mode, the inter-message gap is programmed to be the time measured from the start of the current command to the start of the next command in the bus list. The message will start to be transmitted at the programmed “inter-message gap” time, or later if delayed by previous retries or high-priority aperiodic messages. Sufficient time should be budgeted for all possible retries.

BC_NxtMsgPtr and BC_BranchMsgPtr

The BC_NxtMsgPtr contains the address of the next BC Message Block to be executed.

The BC_BranchMsgPtr is used in place of the BC_NxtMsgPtr for Conditional message blocks when the condition is true and the branch is to be taken.

BC Data Buffer

The BC Data Buffers for receive commands should be programmed during BC initialization. It may be useful to reset the BC Data Buffers for transmit commands during BC initialization, either to all zeros or to invalid values. It is up to the application to ensure that that firmware is not currently processing a receive message when reprogramming the data or updating the transmit data when the application is reading the same data buffer. The circular linked list of data buffers for each message may contain as many data buffers as required to accomplish this within physical memory limitations.

The BC Message, Conditional, Stop BC and Noop Data Buffer formats are depicted in the following four tables. The first two 32-bit words are pointers used for buffer management. The next two 32-bit words are the Time Tag and are identical for all four formats. The BC Message and Conditional formats require additional words. When a Noop is to be flipped to a BC Message or Conditional, the remaining words are also valid and need to be programmed the same as for the BC Message or Conditional BC Message.

BC Message		
32-bit Word	**Bits 31:16**	**Bits 15:0**
0	BC_msg_addr	
1	BC_next_data	
2	BC_TimeTagL	
3	BC_TimeTagH	
4	BC_message_status	
5	BC_StatWord2	BC_StatWord1
6	BC_DataWord2	BC_DataWord1
...	...	...
22	reserved	BC_Data_Word33

Conditional		
32-bit Word	**Bits 31:16**	**Bits 15:0**
0	BC_msg_addr	
1	BC_next_data	
2	BC_TimeTagL	
3	BC_TimeTagH	
4	test_word_address	
5	data_pattern	
6	bit_mask	
7	counter	
8	count value	
9	BC_NxtMsgPtr	
10 - 22	reserved	

Stop BC		
32-bit Word	**Bits 31:16**	**Bits 15:0**
0	BC_msg_addr	
1	BC_next_data	
2	BC_TimeTagL	
3	BC_TimeTagH	
4 - 8	reserved	
9	BC_NxtMsgPtr	
10 - 22	reserved	

Noop		
32-bit Word	**Bits 31:16**	**Bits 15:0**
0	BC_msg_addr	
1	BC_next_data	
2	BC_TimeTagL	
3	BC_TimeTagH	
4 - 22	Identical to BC Message or Conditional if it is turned into one; else, reserved	

BC_msg_addr and BC_next_data

These two 32-bit words are pointers used by the software for buffer management.

The BC_msg_addr is a pointer that points back to the first word of the BC message Control Block. This word is unaffected by the firmware.

The BC_next_data is a pointer to the next buffer in a circular linked list of data buffers. For BC Message data buffers there is a linked list from 1 to n. The maximum n, is only limited by available memory. Once the processing of the current data buffer begins, the firmware loads this value into the BC_Data_Ptr_Current in the BC Message Control Block to be used for the *next* data buffer. For Conditional, Noop, and Stop messages that pointer is always programmed to point to the current buffer.

BC_StatWord1 and BC_StatWord2

BC_StatWord1is the 16-bit status word input from the 1553 bus. For RT-to-RT transfers, this is the status word associated with the transmit command.

BC_StatWord2 is the 16-bit status word input from the 1553 bus associated with the receive command of an RT-to-RT transfer. It is not programmed by the firmware if it is not an RT-to-RT transfer. The software must check the RT-to-RT message format bit in the BC_message_status word to determine if it is valid.

31-27	**26**	**25-21**	**20-16**	**15-11**	**10**	**9-5**	**4-0**
BC_StatWord*2*				BC_StatWord1			

The 16 bits of each status word has the following format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT Address					me	in	sr	Reserved			br	by	sf	db	tf

Field	**Definition**
tf	terminal flag (bit 0)
db	dynamic bus acceptance
sf	subsystem flag
by	busy bit
br	broadcast command received
reserved	reserved bits
sr	service request
in	instrumentation
me	message error
RT_address	RT address bits

BC_message_status

This word contains discrete bits that are set by the firmware and used to post message status and to generate interrupts. The firmware may set more than one error status bit and does not attempt to prioritize errors. Bit definitions are shown in Appendix A.

BC_TimeTagL, BC_TimeTagM

The Time Tag Counter is incremented by a board-internal, free-running clock (or by an external 1PPS clock if the IRIG option has been selected). The 64-bit Time Tag is loaded into two 32-bit words of the BC message buffer at the reception of the mid-zero crossing of the parity bit of each enabled command word (the receive command for RT-to-RT transfers). BC_TimeTagL is the least significant 32 bits; BC_TimeTagM is the most significant 32 bits.

The firmware also stores the Time Tag of execution of Conditional, Stop and Timed Noop Control Blocks. For Noop Control Blocks that do not have the Timed Noop bit set, the most recently recorded time is stored. Recording the Time Tag in message flow control buffers could be helpful in post data analysis or engineering troubleshooting.

The Time Tag Counter is external to the LPU. Please refer to your specific board model's documentation for the Time Tag Counter word format.

Conditional test_word_address

This 32-bit word contains a pointer to a word in memory to be tested for conditional message flow.

The test_word_address may point to any 16 bit word in RAM, but is generally a command word, status word, data word, or message status word. The API masks the details from you. Refer to “BusTools/1553-API Software User’s Manual” for further information.

Conditional data_pattern and bit_mask

The data_pattern is bit-wise exclusive or with the word to be tested before the bit_mask is applied. Bits that don’t match contain logic ones, and if they are not masked out, cause the test to fail.

The bit mask determines which bits to test. If the bit is set in the bit_mask then include that bit position in the word to test; if zero, then exclude the bit position from the test.

Conditional counter and count_value

The conditional count value and associated counter are used for greater flexibility in message sequencing and to save RAM. It allows a condition to be true a specified number of times until the branch message sequence is taken. The API must program both these registers (normally to the same value) during set up. A default of zero is the normal branch on condition after one positive pass. The count is a 32-bit value, where a value of 0000 represents one pass, a value of 0001 represents two passes, etc.

If the conditional counter is zero and the condition tests true, the microcode reloads the count value into the counter and branches on the condition. If the count is non-zero, the microcode decrements the conditional counter every time the condition tests true. The condition can be forced true by clearing the bit mask.

BC_NxtMsgPtr

The BC_NxtMsgPtr contains the address of the next BC Message Block to be executed. It is included in the BC Data Buffer for only Conditional, Stop and Noop message control blocks for use in post data analysis or for engineering troubleshooting.

RT Memory Usage

The 1553 hardware may be set-up to simulate any or all RT's on a 1553 channel. The memory buffers described here and the registers described in the previous chapter must be programmed prior to setting the RT_Run bit.

Once the RT_Run bit is set, the channel enable bit is checked for the particular RT address when each command word is detected on the bus, and the command is either responded to or ignored.

When enabled, the RT address, transmit/receive bit and subaddress fields are used as an offset to point to the 2K-word (4K byte) RT Filter Buffer. The filter contains a pointer to the RT Control Buffer. The RT Control Buffer determines if this particular message is legal, based on the word count field of the command word. If it is legal, the status and data words are transmitted to the 1553 bus. If it is illegal, the message error bit is set in the status word, and the data words are suppressed. The message and associated status is stored in the RT Message Buffer. The message buffer pointer (register) is updated with the address of the next message, allowing for a programmable circular buffer. If the interrupt is enabled, then the interrupt queue is updated, and an interrupt is generated to the host upon message completion.

RT memory usage consists of three buffers, which may be located anywhere within memory:

- RT Filter Buffer
- RT Control Buffer
- RT Message Buffer

RT Filter Buffer

The RT Filter Buffer consists of 2,048 32-bit pointers to the RT Control Buffer. When a command word is received, the eleven bits comprising the RT address, transmit/receive bit and subaddress, are shifted left by 2 bit positions to align with a four-byte boundary; then the result is added to the RT Filter Buffer Base Address register to get a pointer to the RT Control Buffer. The 2,048 pointers are set-up by the software and may point to many different RT Control Buffers.

RT Control Buffer

The RT Control Buffer controls the message buffer sequence used by each RT for responding to commands for each specific sub-address and transmit/receive bit. It allows optimum flexibility in data sequencing by providing a pointer to the next message buffer to respond to a command to a specific sub-address and transmit/receive bit. As the message is being processed the microcode updates the RT message pointer from the next message pointer word of the message buffer, allowing for a circular linked-list of messages. Linking many message buffers allows the application to read or update the buffers less frequently so no message is overwritten before it is read.

The RT Control Buffer also contains a legal bit for each possible command word received. The non-broadcast commands have a different structure than the broadcast commands.

The RT Control Buffer may be as small as a single buffer or it may be many buffers as determined by the unique number of pointers in the RT Filter Buffer.

Non-Broadcast Commands

For each unique RT Control Buffer, a 32-bit word contains the legal bits for each of the 32 possible word count or mode code values. This scheme allows for each of the 65,536 command word combinations to be independently programmed as either legal or illegal. Setting the bit makes the command legal; clearing the bit makes it illegal. If every unique transmit/receive bit, sub-address and word count/mode code combination had its own RT Control Buffer, there would be 2,048 non-broadcast RT Control Buffers,

requiring 16 KB of memory. Most applications only require a few dozen entries. A single entry of all zeros may be used for all disabled or illegal combinations.

32-bit Word	Function (non-broadcast)
0	RT message pointer
1	legal word count/mode code

Broadcast Commands

For broadcast commands, the RT message pointer is followed by thirty-two 32-bit words, one word for each of the 32 word count/mode code values. This buffer always contains thirty-three 32-bit words. This buffer is not required for MIL-STD-1553A or if the RT31Bcst bit of the 1553 Control register is set.

32-bit Word	Function (broadcast)
0	RT message pointer
1	Broadcast legal, word count/mode code = 0
2	Broadcast legal, word count/mode code = 1
....	...
32	Broadcast legal, word count/mode code = 31

RT message pointer

This 32-bit word is a pointer to the first location of a message in the RT Message Buffer. It is initialized by the host. Buffering may be many messages deep since this pointer is updated by the firmware every time a new message is processed. The pointer doesn't change if the buffer is only one message deep.

legal word count/mode code

For non-broadcast commands, 32 bits are used to represent the legal word count for 1553 messages or to represent the legal mode commands for non-message transfers (mode commands are those which have their five-bit subaddress/mode field with "00000" or "11111" for MIL-STD-1553B). A logic "1" in the appropriate bit position means that the RT processes commands received with the corresponding word count/mode code field. Otherwise, it does not.

word count mapping

Bit	word count mapping	mode code mapping
0	word count 32	word count 32
1	word count 1	word count 1
2	word count 2	word count 2
. . .	. . .	. . .

30	word count 30	word count 30
31	word count 31	word count 31

mode code mapping

Bit	Field	Data word	Definition
0	dbc	yes	dynamic_bus_control
1	sync	yes	synchronize
2	tsw	yes	transmit_status_word
3	stst	yes	initiate_selftest
4	xsd	yes	transmitter_shutdown
5	oxsd	yes	override_xmit_shutdown
6	itf	yes	inhibit_terminal_flag_bit
7	oitf	yes	override_inhibit_tf_bit
8	rrt	yes	reset_remote_terminal
15-9	reserved	yes	reserved bits
16	tvw	no	transmit_vector_word
17	swo	no	synchronize
18	tlc	no	transmit_last_command
19	tbw	no	transmit_bit_word
20	sxsd	no	selected_xmitter_shutdown
21	osxs	no	override_sel_xmt_shutdown
31-22	reserved	no	reserved bits

The board responds to the reset_remote_terminal mode command, unless it is a broadcast command, and set the reset_RT bit in the message_status, and resets the RT disable bits and the inhibit terminal flag.

broadcast legal word count/mode code

For broadcast commands, these 32 bits are used to represent that the broadcast command was legal for each of the 31 RT's. A logic “1” in the appropriate bit position means that the RT processes commands received with the corresponding word count/mode code field. Otherwise, it does not. Bit 31 should always be programmed with a logic "0". Note that if an RT is not enabled, the state of the broadcast legal bits is not relevant because the RT will not process any commands including broadcast when not enabled.

RT mapping

Bit	RT mapping
0	RT 0
1	RT 1
2	RT 2
. . .	. . .

30	RT 30
31	0

RT Message Buffer

This twenty-four 32-bit word buffer is used to store MIL-STD-1553 messages and message status. Only one extra word is saved on high_word errors for transmit commands and none for receive commands, in order to limit the size of this buffer.

Note: Broadcast messages are stored once, regardless of the number of RT's that are enabled. The timing sequence of broadcast and non-broadcast messages can be determined by software by using the RT_time_tag words.

Each buffer contains the following data elements in the given order:

32-bit Word	Bits 31:16	Bits 15:0
0	RT_NxtMsgPtr	
1	RT_error_inject_pointer	
2	RT_interrupt_enable	
3	RT_message_status	
4	RT_TimeTagL	
5	RT_TimeTagH	
6	RT_status_word (reserved, if Broadcast)	RT_command_word
7	RT_data_word 1	RT_data_word 0
8	RT_data_word 3	RT_data_word 2
...	...	...
22	RT_data_word 31	RT_data_word 30
23	Extended Status Word	RT_data_word 32

Either one or two RT message buffers are updated for each message received on the bus, given the following message types and criteria:

Message Format	Criteria	Messages Stored
Broadcast BC to RT	Run	Broadcast RT message buffer
RT to RT	Run and transmitting RT enabled	Transmitting RT message buffer
RT to RT	Run and receiving RT enabled	Receiving RT message buffer
Broadcast RT to RT	Run	Broadcast RT message buffer

Broadcast RT to RT	Run and transmitting RT enabled	Transmitting RT message buffer
Other formats	Run and enabled	RT message buffer

RT_NxtMsgPtr

This pointer is loaded into the RT_message_pointer word of the RT Control Buffer location that pointed to it. This provides for multiple message processing before the host needs to be interrupted.

RT_error_inject_pointer

This pointer points to an error_inject_buffer to inject errors on selected words sent to the encoder. If no errors are selected, then it still points to an error_inject_pointer with all zeros. See the description on the error_inject_buffer later in this document.

RT_interrupt_enables

The RT may interrupt the host at the end of every message or only on specific message status. This 32-bit word determines the criteria to generate an interrupt, which is identical to the RT_message_status word. The interrupt enables include specific 1553 protocol status and various error conditions. The RT_interrupt_enables provides a mask for the RT_message_status generated at the end of the message. If the enable bit and the message status bit are both set for any of the 32 bits, then an interrupt is generated. All unused bit positions must be programmed with logic zeros by the software. The bit definitions are listed in the table below. Refer to Appendix A for a complete description of these bits .

Bit	Field	Definition
0	hw	high word
1	iw	invalid word
2	lw	low word
3	is	inverted sync
4	mbe	mid-bit error
5	2bus	two-bus
6	pe	parity error
7	ncd	non-contiguous data
8	er	early response
9	lr	late response
10	res	Reserved
11	bus	bus A or B
12	res	Reserved
13	res	Reserved

Bit	Field	Definition
14	res	Reserved
15	nig	no/short inter-message gap
16	em	end of message
17	bcm	broadcast message
18	RT f	RT-to-RT message format
19	rrt	reset RT
20	st	self test
21	mc	mode code
22	res	no command detected
23	iw2	RT-to-RT message format error
24	res	no-response on receive command of RT-to-RT
25	rtry	retry occurred
26	nres	no response
27	me	message error bit
28	tb	trigger begin
29	te	trigger end
30	bmo	bus monitor overflow
31	res	reserved

RT_message_status

The RT_message_status word contains discrete bits that are set by the hardware and used to generate interrupts (see RT_interrupt_enables in the above section). The host may read these words to determine the interrupt source or to obtain message status. Multiple bits may be set on the same message. The hardware does not attempt to prioritize errors. The structure of these two words are identical to the RT_interrupt_enables words and are described in the preceding paragraph.

RT_TimeTagL, RT_TimeTagM

The Time Tag Counter is incremented by a board-internal, free-running clock (or by an external 1PPS clock if the IRIG option has been selected). The 64-bit Time Tag is loaded into two 32-bit words of the RT message buffer at the reception of the mid-zero crossing of the parity bit of each enabled command word (the receive command for RT-to-RT transfers). RT_TimeTagL is the least significant 32 bits; RT_TimeTagM is the most significant 32 bits. The Time Tag Counter is external to the LPU. Please refer to your specific board model's documentation for the Time Tag Counter word format.

RT_command_word, RT_status_word, and RT_data_words

These are the command, status, and data words input from (or output to) the 1553 bus. The command word is followed by the status word for transmit commands or an ignored word for receive commands, followed by the first through 33rd data words. The last data word is followed by the status word for receive commands, which changes as the word count changes. No data words exist for mode commands of 00000 through 01111. For RT-to-RT commands, separate RT message buffers exist for transmitting and receiving RTs, if they are both enabled.

RT_command_word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT Address					t/r	Subaddress/Mode					Word Count/Mode Code				

Field	**Definition**
Word Count/ Mode Code	word count or mode code bits
Subaddress/Mode	subaddress or mode definition bits
t/r	transmit if set, receive if not set
RT Address	RT address bits

RT_status_word, extended status word

Bit	Field	Definition
0	tf	terminal flag bit
1	res	Reserved
2	ssf	subsystem flag
3	busy	busy bit
4	bcr	broadcast command received bit
7-4	res	Reserved
8	srq	service request bit
9	instr	instrumentation bit
10	me	message error bit
11-15	RT_address	RT address bits

For transmitting the status word, the extended status word is bit-wise OR'ed with the status word in the RT_Address_Buffer if the esw bit of RT_Options is set.

RT address bits

This five-bit field represents RT's at address 0 through 31. There is no RT-status_word for RT 31 if the RT_31_broadcast bit in the hardware register 1553 Control register is set.

message error bit

This bit is set by the hardware after receiving a valid receive command when an invalid data word is detected; there is a data contiguity error or improper word count; or after any command word with an illegal RT sub-address t/r bit combination is detected. The hardware clears this bit after receiving the next valid command, provided none of the above error conditions exist, before sending the next command's status response, except when the next valid command is a "transmit status" mode code (for MIL-STD-1553A or MIL-STD-1553B) or a "transmit last command" mode code (for MIL-STD-1553B only). The software should initialize this bit to logic "0", but should not try to alter this bit when the bus is running.

broadcast command received bit

This bit is set by the hardware after receiving of a valid broadcast command. The Bus Controller may continue to interrogate the broadcast command received bit by transmitting "transmit status" or "transmit last command" mode codes, which do not affect the bit. Transmitting any other non-broadcast message to the RT causes the bit to be cleared, and it is not returned in the status word. The software should initialize this bit to logic "0", but should not try to alter this bit when the bus is running.

busy bit

This bit is set by the RT to indicate that it is unable to transfer data. The RT returns its status word with this bit set and suppresses transmission or reception of all data. Setting of this bit is application specific.

Notes:

1. The RT logs an interrupt to the interrupt queue, even when the busy bit is set.
2. If the RT has its busy bit set when it receives a broadcast command, the RT clears the BCR bit.
3. If the RT has its busy bit set when it receives a non-broadcast command, and the command is not a "transmit last command word" or "transmit status word" mode code, the RT clears the BCR bit.
4. If the RT receives an illegal command when it's sub-system is busy, it sets both the message error (ME) and the busy bit.
5. If the RT receives a command when it is busy, it transmits that command word if the next command is a "transmit last command word" mode code and it is no longer busy.

dynamic bus acceptance

This bit is set after receiving a valid dynamic bus control mode command provided that the DBC_Enabled bit of RT_Options is set.

RT_data_words

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Words															

BM Memory Usage

When the Bus Monitor (BM) is run all unfiltered 1553 bus messages are stored in memory. The RT address, sub-address, and data direction fields of the command word are used as an eleven-bit offset to point to a location in the 2K double-word BM Filter Buffer, which in turn contains a pointer to the BM Control Buffer. The BM Control Buffer determines if this particular message should be captured (stored in memory) or ignored, based on the word count field of the command word. If the message is to be captured, the message and associated status is stored in the BM Message Buffer.

The software may further filter message acquisition by using interrupts. In order for the message to post an interrupt, the BM_interrupt_enable bit of the BM_trigger_header word must be set and the BM Interrupt Enables Register must be initialized.

Triggering enables the application to turn on and off interrupts. When interrupts are disabled, traffic continues to be captured to memory as long as the BM_Run bit is set. Messages may be

polled by software when interrupts are disabled, including those occurring before a start trigger or after a stop trigger.

Bus Monitor data may be uploaded in large blocks containing many messages or messages may be loaded immediately upon reception using interrupts, depending upon the application.

The BM buffer size and location are determined by the BM Start Pointer and BM End Pointer. Current message acquisition location in the buffer is determined by the BM Head Pointer.

The BM memory usage consists of four buffers:

- BM Filter Buffer
- BM Control Buffer
- BM Message Buffer
- BM Trigger Buffer

BM Filter Buffer

The BM Filter Buffer is used for filtering down to the sub-address level. It contains 2,048 pointers to locations in the BM Control Buffer. This allows for a unique pointer for each RT, sub-address and T/R combination of the command word. If such filtering is not required, then multiple pointers point to the same entry in the BM Control Buffer. For RT-RT messages, if either command is enabled then the message is captured.

The following examples depict how the BM Filter Buffer is addressed from the information obtained in the command word. The 11 bits are applied as an offset to the BM Filter Buffer's base address to determine the actual pointer address.

Most Significant 11 Bits of the Command Word

10	9	8	7	6	5	4	3	2	1	0
RT Address					t/r	Subaddress				

Example 1

0	0	0	0	0	0	0	0	0	0	0

RT Address 0
t/r receive
Subaddress 0
BM Filter Buffer base offset 0x000 (32-bit word address)

Address of BM Control Buffer Base Address + 0 (byte address)

Example 2

0	0	0	0	1	1	0	1	1	1	1

RT Address 1
t/r transmit
Subaddress 15 (0x0f)
BM Filter Buffer base offset 0x06f (32-bit word address)

Address of BM Control BufferBase Address + 0x01bc (byte address)

Example 3

0	1	1	0	1	0	1	0	0	0	0

RT Address 13 (0x0d)
t/r receive
Subaddress 16 (0x10)
BM Filter Buffer base offset 0x350 (32-bit word address)
Address of BM Control Buffer Base Address + 0x0d40 (byte address)

Example 4

1	1	1	1	1	1	1	1	1	1	1

RT Address 31 (0x1f)
t/r transmit
Subaddress 31 (0x1f)
BM Filter Buffer base offset 0x7ff (32-bit word address)Address of BM Control Buffer Base Address + 0x1ffc (byte address)

BM Control Buffer

This buffer contains two 32-bit words of information for each unique pointer in the BM Filter Buffer. A separate entry is required for a dummy message buffer, which is accessed for all RT, transmit/receive, sub-address combinations that are not enabled. The total buffer depth may be as little as two 32-bit words (for no filtering) to as large as 4K 32-bit words for filtering of all possible 32 RT's, transmit/receive bit, and subaddress combinations. The simplest method might be to create a two-word buffer for each of the 2K combinations, repeating the same pointer value many

times where filtering isn't required; but that would use the maximum amount of memory.

The first word of each entry is pointed to from the BM Filter Buffer and may be located anywhere within memory.

32-bit Word	Function
0	BM_enable_word_count or BM_enable_mc
1	BM_nth_occurrence

BM_enable_word_count and BM_enable_mc

This 32-bit word is used to filter down to the word level. Setting a bit enables the corresponding word count or mode code. For RT-to-RT messages, the receive command word must be enabled in order to capture the message.

word count mapping

Bit	Word Count Mapping	Mode code Mapping
0	word count 32	Mode code 32
1	word count 1	Mode code 1
2	word count 2	Mode code 2
. . .		
30	word count 30	Mode code 30
31	word count 31	Mode code 31

The MIL-STD-1553B specification defines the mode codes which apply when the sub-address field in the command word is 0 or 31. Below is a summary of the mode codes.

mode code mapping

Bit	Field	Data word	Definition
0	dbc	yes	dynamic_bus_control
1	sync	yes	synchronize
2	tsw	yes	transmit_status_word
3	stst	yes	initiate_selftest
4	xsd	yes	transmitter_shutdown
5	oxsd	yes	override_xmit_shutdown
6	itf	yes	inhibit_terminal_flag_bit
7	oitf	yes	override_inhibit_tf_bit
8	rrt	yes	reset_remote_terminal
15-9	reserved	yes	reserved bits
16	tvw	no	transmit_vector_word
17	swo	no	synchronize
18	tlc	no	transmit_last_command
19	tbw	no	transmit_bit_word

Bit	Field	Data word	Definition
20	sxsd	no	selected_xmitter_shutdown
21	osxs	no	override_sel_xmt_shutdown
31-22	reserved	no	reserved bits

For reset_remote_terminal mode code the board responds to the mode command, unless it is a broadcast command, and set the reset_RT bit in the message_status; it also resets the RT disable bits and the inhibit terminal flag.

The selected_xmitter_shutdown and override_sel_xmt_shutdown mode codes are not implemented. The board responds to the mode command, unless it is a broadcast command, but does not shut down any busses.

BM_nth_occurrence

Note: New designs should avoid using this feature as it is intended that future firmware revisions count a single nth occurrence for all unfiltered messages. Program with 0x00010001 if not using this feature.

This function provides the capability to capture every message on the bus or only one message for every Nth occurrence. This allows the buffer size to be reduced and can prevent a slow host from overrunning the buffer.

There is a separate Nth occurrence counter for each BM Filter entry. This is useful if one message is flooding the bus.

This function consists of two words: the first is the message count value "N" and the second is the counter itself. The software must initialize both words to the same value (default is 0x0001). The firmware decrements the counter each time the message is received and re-loads the message count value into the counter when the count reaches zero.

31-16	15-0
count value	counter

BM Message Buffer

The bus monitor function stores the 1553 messages in a sequential buffer. The first ten 32-bit words are present for every message, although 32-bit words 5 through 9 are not valid for all messages, such as a broadcast mode code without data. The ten words are followed by up to thirty-three 32-bit data words as required by the message type, in the order received from the MIL-STD-1553 receiver. A message therefore consists of between ten and forty-three 32-bit words, depending on the message format and the errors encountered.

The firmware sees the BM message buffer as an endless buffer and wraps messages from the 32-bit word pointed to by the BM_end_ptr and follow it by the 32-bit word pointed to by the BM_start_ptr. A message may cross a BM_end_ptr / BM_start_ptr boundary. The API uses the head pointer to determine the location of the most recent completed message.

32-bit Word	Bits 31:16	Bits 15:0
0	BM_message_header	
1	BM_message_status	
2	BM_TimeTagL	
3	BM_TimeTagM	
4	BM_word_quality	BM_command1_word
5	BM_word_quality	BM_command2_word
6	undefined	BM_response_time1
7	BM_word_quality	BM_status1_word
8	undefined	BM_response_time2
9	BM_word_quality	BM_status2_word
10 - 42 (opt.)	BM_word_quality	BM_data_word

BM_message_header

The 32-bit message header passes critical information about the message to the API. The upper word is programmed by the microcode at the reception of the command word to a known constant, which is used by the API to verify the start of a message block.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused							trig	BM_num_bytes							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xbeef															

Field	Definition
Num_Bytes	Number of bytes
trig	BM_trigger_occurred

Number of bytes

This is the actual number of bytes in the BM message buffer used for this message. This information is required since the API may be unable to use the 1553 command word to determine the actual number of bytes if an incorrect number of data words are received.

BM_trigger_occurred

This bit is set when the message triggers either start posting interrupts or stop posting interrupts.

BM_message_status

The 32-message status bits are set to inform the host of the message status. They are the logical OR of each 1553 word's BM_word_quality, plus additional message status bits which pertain to the whole message. Refer to Appendix A for bit descriptions.

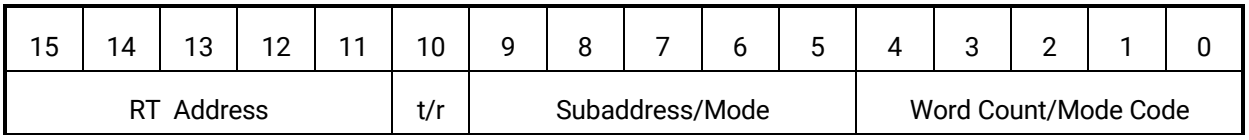
BM_TimeTagL, BM_TimeTagM

The Time Tag Counter is incremented by a board-internal, free-running clock (or by an external 1PPS clock if the IRIG option has been selected). The 64-bit Time Tag is loaded into two 32-bit words of the BM message buffer at the reception of the mid-zero crossing of the parity bit of each enabled command word (the receive command for RT-to-RT transfers). BM_TimeTagL is the least significant 32 bits; BM_TimeTagM is the most significant 32 bits.

The Time Tag Counter is external to the LPU. Please refer to your specific board model's documentation for the Time Tag Counter word format.

BM_command_word1 and BM_command_word2

BM_command_word1 is the 16-bit command word. In the case of an RT-to-RT transfer, this word is the receive command word. BM_command_word2 is the 16-bit transmit command word of an RT-to-RT transfer and is undefined for other message formats.



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT Address					t/r	Subaddress/Mode					Word Count/Mode Code				

Field	Definition
Word Count/ Mode Code	word count or mode code bits
Subaddress/ Mode	subaddress or mode definition bits
t/r	transmit if set, receive if not set
RT Address	RT address bits

BM_status1_word and BM_status2_word

BM_status_word1 is the 16-bit status word. In the case of an RT-to-RT transfer, this word is the transmit status word. BM_status_word2 is the 16-bit receive status word of an RT-to-RT

transfer and is undefined for other message formats. Note that the index (1, 2) for BM_command words is the reverse of the index for BM_status words.

The 16 bits of each status word has the following format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT Address					me	in	sr	Reserved			br	by	sf	db	tf

Field	**Definition**
tf	terminal flag
db	dynamic bus acceptance
sf	subsystem flag
by	busy bit
br	broadcast command received
reserved	reserved bits
sr	service request
in	instrumentation
me	message error
RT_address	RT address bits

BM_data_word

BM_data_word are the 16-bit MIL-STD-1553 data words and are stored in the order that they are received from the bus. A high word error is reported in the BM_message_status and the first extra word is stored in this buffer following the last correct data word. The maximum number of data words is stored 33.

BM_word_quality

Each 1553 command, status, and data word in the message is stored in the lower 16 bits of a 32-bit word and the associated word quality bits are stored in the upper 16 bits. These are the same 16 bits that are stored in the lower word of the BM_message_status as shown in the table below. Refer to Appendix A for complete bit descriptions.

Bit	Field	Definition
0	hw	high word
1	iw	invalid word
2	lw	low word
3	is	inverted sync
4	mbe	mid-bit error
5	2bus	two-bus
6	pe	parity error
7	ncd	non-contiguous data

Bit	Field	Definition
8	er	early response
9	lr	late response
10	ira	bad status address
11	bus	bus A or B
12	res	Reserved
13	res	Reserved
14	res	Reserved
15	nig	no/short inter-message gap

BM_response_time

The 1553 status response time is a 6-bit binary number with a half-microsecond resolution. BM_response_time1 is valid for all 1553 messages for which there are responses. BM_response_time1 is for the transmitting RT of RT-to-RT messages. BM_response_time2 is valid only for the receiving RT of an RT-to-RT transfer. Response times are measured at the receiver of the Bus Controller, referenced from the mid-parity bit time to mid-sync time of the status word. The "Unused" bits should be masked off as they are not guaranteed to be zeros.

31..6	5	4	3	2	1	0
Unused	BM_response_time1					
Unused	BM_response_time2					

BM Trigger Buffer

Bus Monitor Triggering Description

A Bus Monitor trigger may be set-up to indicate that a specific set of MIL-STD-1553 bus events have occurred or that an external input has occurred. Triggers are used to control the posting of BM interrupts, but have no direct effect on the actual storage (capture) of messages.

When the trigger event sequence occurs, the BM_trigger_occurred bit is set in the BM message header word. In the case of an external trigger input, the BM_trigger_occurred bit is set in the BM message header word of the next message to be captured. In addition, if the interrupt is enabled then the interrupter mode is recorded as "Bus Monitor Message and Trigger" rather than "Bus Monitor Message".

A trigger event sequence may be any selected command, status, or data word on the 1553 bus, a logical AND/OR of multiple bus

events, an ARMING-ARMED bus event combination, or an external pulse. Data words are armed by the command word.

There are three trigger sources: a start event sequence, a stop event sequence and an external input. The start event sequence, the stop event sequence and the external input enable may be set up simultaneously. The state of the BM_interrupt_enable (inte) bit determines if a start trigger or a stop trigger occurs. The start trigger and stop trigger alternate as many times as the respective trigger sources are satisfied. Start sequences can only occur while interrupts are disabled and stop sequences can only occur while interrupts are enabled. The following table shows the start and stop event sequences and the state of inte both before and after the trigger for both start and stop triggers.

	Start event sequence satisfied	Stop event sequence satisfied	External input (if enabled)	inte before trigger	inte after trigger
Start	TRUE	X	X	0	1
Start	X	X	TRUE	0	1
Stop	X	TRUE	X	1	0
Stop	X	X	TRUE	1	0

Other options provide the ability to trigger on errors, the ability to generate an external output pulse when the trigger event occurs and the ability to specify the number of times that you want the event to occur.

BM_trigger_buffer

The bus monitor trigger buffer consists of 23 double-words: a header word, eight double-words for start event control, eight double-words for stop event control and six unused double-words. This buffer must be programmed during board setup, but may be modified during real-time provided the appropriate enable bits are turned off in the BM_trigger_header.

32-bit Word	Description, bits 31 - 16	Description, bits 15 - 0
0	BM_trigger_header	
1	start event 1 BM Message Buffer byte offset	start_event_register
2	start event 1 bit_mask	start event 1 value_to_compare
3	start event 1 initial_count	start event 1 count
4	start event 2 value_to_compare	start event 2 BM Message Buffer byte offset

32-bit Word	Description, bits 31 - 16	Description, bits 15 - 0
5	start event 2 count	start event 2 bit_mask
6	start event 3 BM Message Buffer byte offset	start event 2 initial_count
7	start event 3 bit_mask	start event 3 value_to_compare
8	start event 3 initial_count	start event 3 count
9-11	unused	
12	stop event 1 BM Message Buffer byte offset	stop_event_register
13	stop event 1 bit_mask	stop event 1 value_to_compare
14	stop event 1 initial_count	stop event 1 count
15	stop event 2 value_to_compare	stop event 2 BM Message Buffer byte offset
16	stop event 2 count	stop event 2 bit_mask
17	stop event 3 BM Message Buffer byte offset	stop event 2 initial_count
18	stop event 3 bit_mask	stop event 3 value_to_compare
19	stop event 3 initial_count	stop event 3 count
20-22	unused	

BM_trigger_header

The header word is the first word of the buffer and applies to both start capture and stop capture triggers.

Bit	Field	Definition
0	inte	BM_interrupt_enable
1	senb	enable_start_trigger
2	henb	enable_stop_trigger
3	res	reserved
4	ext	enable external trigger
5	terr	trigger on errors
6	Res	reserved
7	Xot	external output on trigger
8-31	Res	reserved

BM_interrupt_enable

The application may select capture windows (chronological filters) for uploading 1553 bus traffic without stopping the Bus Monitor from storing continuous bus traffic. Setting the BM_interrupt_enable turns on the capture of messages; clearing it stops capture.

The software may directly program the BM_interrupt_enable bit or it may be set or cleared by firmware by setting up BM triggers.

If implementing a start trigger, BM_interrupt_enable is cleared by software, then set by the firmware when all of the trigger events occur in the given order. If implementing a stop trigger, BM_interrupt_enable is cleared by the firmware when the enable_stop_trigger bit is set and all of the events occur in the given order.

The BM_interrupt_enable bit may also be armed by an external hardwired discrete or differential input if the enable_external_trigger bit is set. It is then inverted (start or stop) when the next message has been input from the 1553 bus.

If triggering is not used, then set BM_interrupt_enable, clear the enable_start_trigger and enable_stop_trigger bits and program the trigger function to "automatically trigger"in both the start event register and the stop event register, prior to turning on the BM_Run bit.

enable_start_trigger

If the Bus Monitor is to be start triggered by a 1553 bus event, the software sets the enable_start_trigger bit, and if not already clear, it also clears the BM_interrupt_enable bit. Once the trigger event(s) occur, the firmware sets the BM_trigger status bit in the BM message header word and sets the BM_interrupt_enable bit.

enable_stop_trigger

If the Bus Monitor is to be stop triggered by a 1553 bus event, then the software sets the enable_stop_trigger bit, and if not already set, it also sets the BM_interrupt_enable bit. Once the trigger event(s) occur, the firmware sets the BM_trigger status bit in the BM message header word and clears the BM_interrupt_enable bit.

enable_external_trigger

This bit provides the capability to use a trigger input source hardwired externally to the board. The trigger occurs once the first 1553 message is input on the bus (any message) following the input pulse (an active high signal with a minimum 400 ns duration). The external trigger inverts the BM_interrupt_enable bit in the BM_trigger_header word and the set the BM_trigger status bit in the BM message header word. The registered input pulse is cleared after each message is captured (BM end of message) or if the BM is turned off.

trigger_on_errors

This enables triggering on an event (command, status, or data) even if the message is not valid (Manchester or parity error). If this bit is not set, triggering occurs only on valid 1553 messages.

external_output_on_trigger

Setting this bit causes a 50 microsecond pulse on the trigger_out port of the LPU when the Bus Monitor trigger occurs. The trigger_out port may be assigned to an available I/O port on your board (see the Hardware User manual for your board for a description of how to set up the output trigger).

start_event_register and stop_event_register

The event register determines what sequence of events must occur and keeps track of the events when they do occur. There are two event registers, one for the start_trigger function and one for the stop_trigger function. Software must set-up these registers prior to enabling the start or stop trigger bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								trigger function				0	ev3	ev2	ev1

Field	Definition
ev1	event 1
ev2	event 2
ev3	event 3
trigger function	trigger function

ev1, ev2, and ev3

The software clears these events on initialization of the trigger buffer, and the firmware sets them as each of (up to) three events occur. They are used by the firmware to determine if the trigger criteria have been met. The firmware then clears these bits when a trigger occurs.

trigger_function

These four bits determine the trigger function/sequence. A value of zero triggers upon the completion of any command. A value of one is specified for a trigger on a single event. Multiple events may be used to trigger by using AND, OR or ARM trigger functions.

Multiple events must be programmed for unique BM_message_buffer's except when specified that they must be specified to occur on the same message. To trigger on a status

word, data word, word quality or message status for a specific command word, specify a trigger function that links the word to the command word on the same message (enter a value of 2, 4, 5, 6 or 7).

Value	Trigger criteria
0	automatically trigger
1	single event (event 1)
2	event 1 AND event 2 on the same message
3	event 1 AND event 2
4	(event 1 AND event 2 on the same message) AND event 3
5	(event 1 AND event 2 on the same message) OR event 3
6	event 3 ARMED by (event 1 AND event 2 on the same message)
7	(event 1 AND event 2 on the same message) ARMED by event 3
8	event 1 AND event 2 AND event 3
9	event 1 OR event 2 OR event 3
10	event 2 ARMED by event 1
11	event 1 OR event 2
12-15	reserved

BM_trigger_event_buffer

There are four events available each for the start trigger and for the stop trigger. Five words are used for each event. The words are in the following sequence:

- byte_offset
- value_to_compare
- bit_mask
- initial_count
- event_count

BM Message Buffer offset

Enter the byte offset of the BM_message_buffer for the 16-bit word to be tested for the trigger event. If trigger events are enabled (either enable_trigger_start is set or enable_trigger_stop is set), unused byte offsets should all be programmed with 0x0. Here are some example offsets:

16-bit Word to be tested	Byte Offset
Message Status word	0x04
Command word	0x10
Status word	0x1c
Data word 1	0x28

16-bit Word to be tested	Byte Offset
Data word 32	0xa4

value_to_compare

This is the value of the word that is used to determine the trigger event. For instance, if the event is the reception of a command for RT 1, receive, sub-address 1 with one data word, enter 0x0821 into this location.

bit_mask

This word is logical ANDed with the value_to_compare to use only a single bit or a field (multiple bits) for the trigger value. For the bits that are set here, the corresponding bits in the event word are included in the comparison test. For bits that are clear here, the corresponding bits in the event word are ignored. If no mask is specified, this word contains 0xffff (initialization default) and the whole word is tested.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16 bit data mask word, ANDed with the event data value															

event initial_count and event count

Enter the number of times that you want the specified event to occur before making it cause a trigger. The firmware decrements the event count upon each occurrence. When it reaches zero, it sets the BM_event_occurred bit and reloads the event count with the event initial_count.

Notes:

When two events are programmed for the same message, both the initial_count and the event_count must be programmed with the same values for both messages.

Both the initial_count and the event_count must be programmed with 0x1 for the armed (second) event of an arming/armed event sequence. The first event must occur as many times as its event_count before the second event is counted. The second event must then occur once before trigger occurs.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16 bit event count															

Error Injection Buffer

The words in this buffer determine the errors to inject on each word sent to the encoder to be output on the 1553 bus. The hardware writes this information in parallel with the 1553 word sent to the

encoder. Each Error Injection Buffer consists of 34 consecutive 16-bit words. The number of Error Injection Buffers depends on the number of unique error sequences programmed by the application and is limited by the available RAM.

There always exists a 34-word buffer of all zeros for messages without error injection. The first word of each buffer corresponds to the command1, status1, or status2 word, depending on which mode (BC or RT) and if it's an RT-to-RT message that invokes the buffer. The second through 34th words correspond to the first through 33rd data word of the message, respectively, except in the case of an RT-to-RT message in the BC mode, where only the second word is used and it corresponds to the command2 word. The actual number of error injection words corresponding to the data words must equal or exceed the maximum number of data words of all the messages that invoke the buffer, but must not exceed 33.

The word format is defined separately for zero-crossing error injection than for all other error injection.

error_injection_word (except zero-crossing)

Thirty-four consecutive 16-bit words are required for each buffer. The 16-bit word format is as follows.

<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>5-0</td><td>Information associated with the error to be injected (error_info) (BC, RT)</td></tr>
<tr><td>6</td><td>bit count error (BC, RT)</td></tr>
<tr><td>7</td><td>inverted sync error (BC, RT)</td></tr>
<tr><td>8</td><td>parity error (BC, RT)</td></tr>
<tr><td>9</td><td>non-contiguous data (BC, RT)</td></tr>
<tr><td>10</td><td>word count error (BC, RT)</td></tr>
<tr><td>11</td><td>programmable response (RT)</td></tr>
<tr><td>12</td><td>respond to the wrong address (RT)</td></tr>
<tr><td>15-13</td><td>encoded error
<table>
<tr><td>000</td><td>none</td></tr>
<tr><td>011-001</td><td>reserved</td></tr>
<tr><td>100</td><td>transmit 1553 on the wrong bus (RT)</td></tr>
<tr><td>101</td><td>zero-crossing (BC, RT)</td></tr>
<tr><td>110</td><td>bi-phase error (BC, RT)</td></tr>
<tr><td>111</td><td>zero-crossing (BC, RT)</td></tr>
</table></td></tr>
</table>

Information associated with the error to be injected (error_info)

These six bits qualify some of the injected errors. They should always be zeros when not used.

bit count error

The hardware normally transmits a sync pulse followed by 16 data bits and ending with a parity bit for each word sent out on the 1553 bus. When the bit count error is selected, the five LSBs of the error_info bits are loaded into a counter representing the bit count in binary. The counter accepts values from zero to 31. The bit count error is programmed on a word-per-word basis.

For bit counts of 18 or greater, the decoder sets the Low Word error status bit and reset to the next message. If simulating the Bus Controller, sufficient inter-message gap time must be allowed for transmit commands to prevent both the RT and the BC from transmitting on the bus simultaneously.

inverted sync

Setting this bit modifies the sync bit of the associated 1553 word.

The six err_info bits are used to invert any combination of half-bit times (500 ns) for the six half-bit times of the sync bit. A logic low on the respective err_info bit inverts the sequential half-bit time of sync.

For the five invalid sync patterns injected into the *command sync* in the RT Validation Specification, the following err_info bits apply:

111100	111011 (0x3b)
110000	110111 (0x37)
111001	111110 (0x3e)
011000	011111 (0x1f)
000111	000000 (0x0) – This inverts a command sync

For the five invalid sync patterns injected into the *data sync* in the RT Validation Specification, the following err_info bits apply:

000011	111011 (0x3b)
001111	110111 (0x37)
000110	111110 (0x3e)
100111	011111 (0x1f)
111000	000000 (0x0) – This inverts a data sync

Patterns other than 111000 on data sync and 000111 on command sync (inverted data) are available only on V5.0 firmware and later.

parity error

This error transmits a word with an inverted parity bit.

non-contiguous data (data gap)

This bit set injects a gap following the associated word. The three LSB's of the err_info field are used to select the amount of the gap. Valid values are:

1	0.5 us.
2	1.0 us.
3	1.5 us.
4	2.0 us.
5	2.5 us.

word count error

The hardware accepts any word count from 1 to 33, programmed in binary in the error_info bits. You enter the actual word count, which overrides the word count in the command word. This error is entered on the word corresponding to the command word of a BC receive message or the status word of an RT transmit message only and pertains to the whole message.

Note: Word count error injection is limited to a single high word being transmitted: entering a word count which is two or more than in the word count field of the command word transmits a single high word.

programmable response

This RT selection allows the application to enter a programmable response time of up to 31½ microseconds, measured from the mid-parity time of the receiver to the mid-sync time of the status word, programmed as a binary number in the error_info bits, with the LSB equal to 500 ns. An entry in the error_info of less than 4-microseconds constitutes an *early response*. An entry greater than 14-microseconds constitutes a *late response* according to the MIL-STD-1553B Specification. Values of less than 8½ microseconds *are not guaranteed* to be attainable by the hardware.

respond to wrong address

Setting this bit causes the RT to respond with the status word using the address programmed in the five LSBs of the error_info bits. Otherwise, it responds with the correct address.

bi-phase error

A bi-phase bit error is when there is no zero-crossing for the entire bit time. This error injection inhibits a zero crossing for the selected bit. It may be programmed as a logic high or a logic low.

The error_info[3:0] bits determine the selected bit of the MIL-STD-1553 word: 0 through 15. Bit 0 is the LSB of the 16-bit word, however, it is the last bit to be transmitted in the serial stream. Similarly, bit 15 is the MSB and is the first bit to be transmitted. The error_info[4] bit must be programmed with zero.

The error_info[4] bit is used in conjunction with error_info[3:0] bits to inject a bi-phase error onto the parity bit time. If error_info[4:0] is programmed to binary 11111, then the parity bit time is selected for error injection.

The error_info[5] bit determines if the injected bit is at a logic high or a logic low state for the entire bit time. Setting err_info[5] forces the signal to a logic high state throughout the bit time; clearing err_info[5] forces the signal to a logic low state throughout the bit time.

It is not predictable how a 1553 decoder interprets a word when a bi-phase error is injected onto one of the first two bits, as it depends on timing and the states of the first two bits. When the bit doesn't cross zero, it may stay in one state for 1½ microseconds and the decoder might try to re-establish sync.

wrong bus

This bit set causes the BC to transmit on the alternate bus from that which it is programmed or the RT to transmit on the alternate bus from that which it receives the command word.

error_injection_word (zero-crossing)

Setting the three MSB's to 111 of the error injection word causes a time shift of the zero-crossing point of a bit in the corresponding 1553 word. The word format is as follows.

Bit	Field	Definition
5-0	ZC_HalfBit	Bit to time shift in the 1553 word
7-6	reserved	
11-8	ZC_Offset	Time offset of the transition
12	ZC_Early	Early transition (0 for late transition)
15-13	111	Selects zero-crossing time shifting

A single bit of the 1553 word must be selected to inject the zero-crossing shift upon. The six-bit ZC_HalfBit field is used to select the desired bit as follows.

Selected bit of 1553 Word	Value to use in ZC_HalfBit
Sync	4
Bit 1	8
Bit 2	10
Bit 3	12
Bit 4	14
Bit 5	16
Bit 6	18
Bit 7	20
Bit 8	22
Bit 9	24
Bit 10	26
Bit 11	28
Bit 12	30
Bit 13	32
Bit 14	34
Bit 15	36
Bit 16	38
Parity bit	40

The transition may be programmed to occur early of the expected transition by setting the ZC_Early bit, or late of the expected transition by clearing the ZC_Early bit. The four-bit ZC_Offset field determines the amount of time that the transition is offset, with a resolution of 25 ns.

Late Transition Offset (ns)	Value to use in ZC_Offset
0	0
25	4
50	8
75	12
100	16
125	20
150	24
175	28
200	32
225	36
250	40
275	44
300	48
325	52
350	56
375	60

Early Transition Offset (ns)	Value to use in ZC_Offset
25	13
50	9
75	5
100	1
125	60
150	56
175	52
200	48
225	44
250	40
275	36
300	32
325	28
350	24
375	20
400	16
425	12
450	8
475	4
500	0

Interrupt Queue Buffer

An interrupt queue (IQ) keeps a history of interrupt events. It is written to by the firmware when the condition causing an enabled interrupt occurs. When the host responds to the interrupt, it checks the queue to service each event. This queue allows the host to buffer interrupt events so that it need not respond to them immediately. The CPU interrupt bit in the 1553 Control register is set for each interrupt event, which may be enabled (by setting the CPU interrupt enable bit in the 1553 Control register) to assert a hardware interrupt line to the host.

Each interrupt event requires two 32-bit words:

- interrupter_mode
- interrupting_message_pointer

The Bus Controller has the ability to enable asserting the CPU interrupt bit (hardware interrupt) on a message per message basis.

IQ pointers for buffer size and head/tail servicing are described in Chapter 2.

interrupter_mode

Each type of interrupting source has a specific value coded into the interrupter_mode word. It is the responsibility of the software to clear the interrupter_mode to a value of "0" after reading it.

Value	Interrupting Source
0	Interrupt not present
1	Time Tag load (BC, BM, RT)
2	Trigger input (BC, RT)
3	Bus Controller message (BC)
4	Bus Controller message flow control block (BC)
5	Remote Terminal message (RT)
6	Bus Monitor message (BM)
7	Bus Monitor message and trigger (BM)
8	Minor frame overflow (BC)
9	BC_busy bit set on minor frame overflow (BC)
10	Low priority frame overflow (BC)
11	High priority frame overflow (BC)
12	Bus Monitor overflow (BM)
13 - 65,535	Unused

Interrupt not present

This value may be used to support interrupt management. An interrupt_mode value of zero should not be encountered.

Time Tag load interrupt

If this bit is set, the Time Tag has been loaded with the contents of the tag time counter load (TTCL) register. No interrupting_message_pointer is associated with this interrupter_mode.

trigger input interrupt

If this bit is set, a trigger input on the selected I/O port has occurred. The enabled_trigger_interrupt bit in the 1553 Control register must be set to enable this interrupt source. No interrupting_message_pointer is associted with this interrupter_mode.

Bus Controller message interrupt

If this bit is set, a 1553 message has been input to the BC buffer, the "interrupt on this message" bit in the BC Control word is set

and bitwise ANDing the BC Interrupt Enables Register with the BC_message_status word produces a result that is TRUE.

Bus Controller message flow control block interrupt

This bit is set when the "interrupt on this message" bit in the BC Control word is set for a Noop, BC Conditional Branch Block or BC Stop Block has been executed. It is set regardless of whether the branch is taken for a Conditional Branch Block.

Note that a Noop control block that follows a message block could interrupt before the message block. If this is undesirable than use a Timed Noop of at least twenty microse in place of the Noop. The interrupt for a Timed Noop that follows a message thus occurs in the order of the bus list.

Remote Terminal message interrupt

If this bit is set, a 1553 message has been input to the RT buffer and bitwise ANDing the RT_interrupt_enable word with the RT_message_status word produces a result that is TRUE.

Bus Monitor message interrupt

If this bit is set, a 1553 message has been input to the BM buffer and bitwise ANDing the BM Interrupt Enables Register with the BM_message_status word produces a result that is TRUE.

Bus Monitor message and trigger interrupt

If this bit is set, then two conditions must be true: 1) a Bus Monitor Trigger sequence is complete or an input trigger has occurred, if enabled, and 2) a 1553 message has been input to the BM buffer and bitwise ANDing the BM Interrupt Enables Register with the BM_message_status word produces a result that is TRUE.

It provides interrupt notification for, and is always set at the same time as the BM_Trigger_Occurred bit in the BM_Header word in the BM message buffer.

Minor frame overflow interrupt (BC)

The Minor frame overflow interrupt only occurs if the Minor frame overflow interrupt enable bit in the 1553 firmware control register is set.

If this bit is set, a 1553 message has exceeded the allotted frame time. The system designer must ensure that there is sufficient time within a frame to transmit and process all the programmed messages. The total minimum frame time is a summation of the time it takes to transmit a 1553 word multiplied by the number of words, plus all the response and inter-message gap times in a

frame. The analysis must also account for extra words associated with RT to RT messages, retried messages (if enabled), high word error injection, long programmable responses, response time-outs and a minimal amount of firmware processing time.

When a message is to be processed, frame predictor hardware determines if there is sufficient time to output the message before the next frame time. If the hardware detects insufficient time to process the message then the message and all subsequent messages in the minor frame are not transmitted. Frame timing is preserved and the new frame begins at the appropriate time. The frame predictor accuracy limits the bus B/W to 95% or greater. The AMFO bit in the 1553 Control Word may be set to ignore the frame predictor logic, but use of this bit does not guarantee proper frame timing but with proper analysis it may marginally increase bus bandwidth.

This interrupt type is not valid when Message Scheduling is used.

This interrupt type is not valid for aperiodic messages.

BC_busy set on minor frame overflow interrupt (BC)

The BC_busy minor frame overflow interrupt only occurs if the Minor frame overflow interrupt enable bit in the 1553 firmware control register is set.

The BC_busy minor frame overflow interrupt occurs when the BC_busy bit is set at the start of a minor frame. This indicates that the firmware has not completed the previous frame; which could mean that a message transmitted into the next frame or that the firmware is out of sync with the bus list. Careful bus list analysis must be performed when this bit is set. Possible causes include: misuse of the AMFO bit, erroneous conditional branching or incompatible firmware/software versions.

This interrupt type is not valid when Message Scheduling is used.

This interrupt type is not valid for aperiodic messages.

The interrupting_message_pointer points to the next message in the bus list which has not yet been processed.

Low priority overflow interrupt (BC)

The Low priority overflow interrupt only occurs if the Minor frame overflow interrupt enable bit in the 1553 firmware control register is set.

The firmware was attempting to process a low priority aperiodic message when the minor frame time ended. The message is transmitted on the first subsequent frame when there is sufficient time after the regular message list has been transmitted.

This interrupt does not indicate an error and is for informational purposes only.

High priority overflow interrupt (BC)

The High priority overflow interrupt only occurs if the Minor frame overflow interrupt enable bit in the 1553 firmware control register is set.

The firmware was attempting to process a high priority aperiodic message when the minor frame time ended. This message and subsequent messages in the high priority message list begin being transmitted at the start of the next minor frame.

This interrupt does not indicate an error and is for informational purposes only.

Bus Monitor overflow interrupt (BM)

When the firmware detects that the BM Head Pointer has "caught up" with the BM Tail Pointer (both pointers have the same value) then the interrupt is set. This indicates that the software did not keep up with the bus traffic. Messages that overflow are not stored in the message buffer. The interrupting_message_pointer contains the BM Head Pointer of the last complete message that is stored in the buffer.

interrupting_message_pointer

Points to the BC Data Buffer, RT Message buffer or BM Message buffer that caused the interrupt. This pointer is valid for all interrupt modes unless explicitly excluded in the descriptions above.

Playback Memory Usage

Playback Buffer

A playback file must be created using the format specified here in order to use playback operation.

The Playback Buffer consists of a series of 16-bit Playback Code words, each followed by a variable number of Mil_Std-1553 timing and message words. Playback Time Codes are used to specify the timing and Xmit Message Codes are used to specify the data to be transmitted onto the MIL-STD-1553 bus. The End of playback Code stops playback operation.

A Playback Time Code is always followed by two 16-bit words representing the playback time; a Xmit Message Code is followed by the number of 1553 words designated in the Word Count field, unless the Bit Count Error bit is set, in which case it is followed by 32 data bits (of which only 16-bits are implemented). The Playback Data Buffer uses the following message code format.

Playback Code Format		
Bit(s)	**Function**	**Description**
15	Bus	Valid only for Xmit Message Codes. 0 = 1553 bus A; 1 = 1553 bus B
14	Xmit Message Code	When set, this bit indicates that the following 16-bit words in the buffer are transmitted onto the 1553 bus. If the bit count error bit is cleared the hardware expects the number of 1553 words indicated in bits 5 through 0 to follow the message code. If the bit error count bit is set the hardware expects two 16-bit words to follow the message code as explained in the bit count error bit description
13	Playback Time Code	This bit is used to hold off transmission of the 1553 Word(s) associated with the following Xmit Message Code in order to generate the response time (preceding a status word) or an inter-message gap time (preceding a command word). When set, this bit indicates that the following two 16-bit words contain the time that the next 1553 word transmits: the first 16-bit word contains the lower 16 bits and the second 16-bit word contains the upper 16 bits of the 32-bit time. The timer has a resolution of 500 ns. This time is referenced from the beginning of the playback operation. **Note:** The application must calculate the time information using the word count, Time Tag, and response time information found in the bus monitor message buffer; then add it to the previous calculated time to program the buffer with the cumulative time since the Playback Run bit was set.
12	End of playback Code	When set this bit informs the hardware the end of the playback data has been reached. It also clears the Playback Run bit (bit 2) in the playback status register.
11-9	Not used	–
8	Parity error	Valid only for Xmit Message Codes. When set, parity is inverted from calculated value. Parity is calculated from total number of bits sent.
7	Sync polarity	Valid only for Xmit Message Codes. When clear this bit indicates the sync pattern preceding each 1553 word is a high followed by a low (nominally command and status sync). If this bit is set then the sync pattern preceding each 1553 word is a low followed high (nominally data sync).
6	Bit count error	Valid only for Xmit Message Codes. ▪ When the bit count error is set, it indicates that two 16-bit words follow the Xmit Message Code and that bits [5..0] of the Xmit Message Code indicate the number of significant bits contained in the two words. The first bit sent is the MSB of the first word while the last bit sent is the LSB of the second word. Only the bit values of the second word are transmitted: bits after bit 16 are played back but have undetermined values. ▪ When the bit error count bit is cleared bits [5..0] indicate the number of 16-bit 1553 words that follow the Xmit Message Code.
5-0	Word count/ Bit count	Valid only for Xmit Message Codes. ▪ When the bit count error (bit 6) is clear this field indicates number of words following the Xmit Message Code to be transmitted on the 1553 bus. Valid word counts are 1 through 63. ▪ When the bit count error (bit 6) is set this field indicates the number of significant bits contained in the two words following the Xmit Message Code (32 max). The first bits to be sent is the MSB of the first word while the last bit to be sent is the LSB of the second word.

When the software sets the Playback Start bit in the Playback Control/Status register, the firmware sets the Playback Run bit. Setting the Playback Run bit resets a 32-bit free-running Counter which is incremented every 500 ns. The Playback Time Code

determines playback timing of the 1553 words of the Xmit Message Codes that follows it. The Playback Time Code is therefore a cumulative time and it is up to the application or API driver to compute the time to wait until the next set of 1553 words are transmitted. If there is no gap between words, such as with the transmission of a receive command followed by 1553 data words, then two consecutive Xmit Message codes may be entered: the first with command sync and word count of one, the second with data sync and word count of the number of 1553 data words in the message. The Xmit Message Code for the data words would be followed by a Playback Time Code for the timing of the status word or for the next Command word if there were no response.

As playback continues, the free-running gap counter is compared to each Playback Time Code in turn and the 1553 words of the following Xmit Message Code are not transmitted until the Playback Time Code time is equal to the free running counter. The counter free runs through 32-bits binary at half-microsecond resolution until the Playback Run bit is turned off. This limits playback to approximately 33 minutes before timing wraps back to zero.

Each 1553 word takes 20 microseconds and response times are between 4 and twelve microseconds: typically about 8 microseconds. It should be apparent that the time values in the two words following the Playback Time Codes should be increasing as the Playback buffer is executed.

1553 playback message packet format

Following are examples of how the playback messages are constructed using the message code.

BC-RT Transfer

RECEIVE COMMAND		**DATA WORD 1**		…	**DATA WORD n**	*response time*			**STATUS WORD**		*inter-message gap*		
XMIT MESSAGE CODE (command)	DATA (command)	XMIT MESSAGE CODE (n data words)	DATA (word 1)	…	DATA (word n)	PLAYBACK TIME CODE (response time)	DATA (gap LSW)	DATA (gap MSW)	XMIT MESSAGE CODE (status)	DATA (status)	PLAYBACK TIME CODE (gap time)	DATA (gap LSW)	DATA (gap MSW)

RT-BC Transfer

TRANSMIT COMMAND		*response time*			**STATUS WORD**		**DATA WORD 1**		…	**DATA WORD n**	*inter-message gap*		
XMIT MESSAGE CODE (command)	DATA (command)	PLAYBACK TIME CODE (response time)	DATA (gap LSW)	DATA (gap MSW)	XMIT MESSAGE CODE (status)	DATA (status)	XMIT MESSAGE CODE (n data words)	DATA (word 1)	…	DATA (word n)	PLAYBACK TIME CODE (gap time)	DATA (gap LSW)	DATA (gap MSW)

RT-RT Transfer

RECEIVE COMMAND	**TRANSMIT COMMAND**	*response time*	**STATUS WORD**	**DATA WORD 1**	…	**DATA WORD n**

XMIT MESSAGE CODE (command)	DATA (receive command)	XMIT MESSAGE CODE (command)	DATA (transmit command)	PLAYBACK TIME CODE (response time)	DATA (gap lsw)	DATA (gap MSW)	XMIT MESSAGE CODE (status)	DATA (status)	XMIT MESSAGE CODE (n data words)	DATA (word 1)	…	DATA (word n)	→

response time			**STATUS WORD**		*inter-message gap*		
PLAYBACK TIME CODE (response time)	DATA (gap LSW)	DATA (gap MSW)	XMIT MESSAGE CODE (status)	DATA (status)	PLAYBACK TIME CODE (gap time)	DATA (gap LSW)	DATA (gap MSW)

The following example transmits a transmit command with two data words to RT 0. Wait 4 microseconds after the Playback Run bit is set before outputting the command word in order to ensure deterministic timing, as there may be up to 4 microseconds for the microcode to initialize operation.

0x2000	Playback Time Code
0x0008	Output 1st word 4 us after PB_start is set, bits[15:0]
0x0000	Output 1st word 4 us after PB_start is set, bits[31:16]
0x4001	Xmit Message Code: 1 word, command sync
0x0422	Command Word (RT=0, Tx, SA=1, WC=2)
0x4001	Xmit Message Code: 1 word, command sync
0x0000	Status Word (RT=0)
0x4082	Xmit Message Code: 2 words, data sync
0x1111	Data Word 1
0x2222	Data Word 2
0x1000	End of Playback Code

APPENDIX A

1553 Message Status/Interrupt Enables

Introduction

There are 32-bits of 1553 message status associated with every message. For each respective mode of operation, this word is located in the BC Message Block, the RT Message Buffer and the

BM Message Buffer. Bits 0 through 15 are also used for each 16-bit BM_word_quality word.

These bits are also used to post message interrupts in BC, BM and RT modes.

The table summarizes bit usage for each of the three modes by a check mark. A double-dash indicates that the bit does not apply to that mode. Note that bits with a double-dash have undetermined values and should be ignored.

Bit	Definition	BC	RT	BM
0	high word	√	√	√
1	invalid word	√	√	√
2	low word	√	√	√
3	inverted sync	√	√	√
4	mid-bit error	√	√	√
5	two-bus	√	√	√
6	parity error	√	√	√
7	non-contiguous data	√	√	√
8	early response	√	--	√
9	late response	√	--	√
10	bad status address	√	√	√
11	bus A or B	√	√	√
12	reserved	--	--	--
13	reserved	--	--	--
14	reserved	--	--	--
15	no intermessage gap	√	√	√
16	end of message	√	√	√
17	broadcast message	√	√	√
18	RT-to-RT message format	√	√	√
19	reset RT	--	√	--
20	self test	--	√	--
21	mode code	√	√	√
22	no command detected	√	--	--
23	RT-to-RT message format error	√	√	√
24	no-response on receive command of RT-to-RT	√	--	√
25	retry occurred	√	--	--
26	no response	√	--	√
27	message error bit	√	√	√
28	trigger begin	--	--	√
29	trigger end	--	--	√
30	bus monitor overflow	--	--	√
31	reserved	--	--	--

high word and low word

When the 1553 decoder receives a valid command word, it stores the word count and decrements it on receiving each data word. If the word count field doesn't agree with the actual number of data words received, either the high word bit or the low word bit is set.

Note: In order to satisfy the requirements of the RT Validation specification, high word status is not detected for a channel that is programmed in flash as a single RT (sRT mode).

invalid word

The invalid word bit is the logical "or" of the inverted sync, mid-bit error, parity error, non-contiguous data and high word count error.

inverted sync

This bit is set when the decoder is expecting a sync bit of the opposite polarity of that which is received.

mid-bit error

This bit is set if a Manchester II error is detected on bits 2 through 16 or on a parity bit. A low bit count is a mid-bit error.

two-bus

This bit is set when both bus A and bus B decoders are processing a 1553 word at the same time. If a valid sync plus two data-bits is present on one bus, and if a valid sync plus two data-bits becomes present on the other bus before a mid-parity occurs on the first, then two-bus becomes active. It remains active until cleared by the microcode at the end of the message. Since the encoder output feeds back to the decoder, two-bus detection may be used to test cable continuity by connecting a cable from bus A to bus B and transmitting on bus A.

parity error

This bit is set when the parity bit is even parity. It might not be set for a mid-bit error.

non-contiguous data

This bit is set when there is a gap between data words of ½ microsecond to 2 microseconds of bus dead time.

early response

A status response time of less than 4.0 microseconds is considered an early response. Response time is measured from the mid-parity bit crossing of the last word from the Bus Controller

to the mid-sync time of the status word. The BC time includes cable delays, however, the Bus Monitor does not.

late response , no response and no_response2

The late response and no response errors are determined by comparing the programmed time-out values in the hardware response_reg with the time elapsed since mid-parity when a status word is expected. A no-response time-out clears the late_response bit.

The response time is measured from the mid-parity bit crossing of the last word from the BC to the mid-sync time of the status word. The BC time includes cable delays, however, the Bus Monitor does not.

Normally, a response time between 12 and 14 microseconds is a late response. Anything after that is a no_response error, however, these times are programmable from 4 to 60 microseconds in the hardware response_reg for system flexibility.

To differentiate between a no-response of the transmitting RT from the receiving RT on an RT-to-RT transmission, the no_response2 bit is used. If the transmitting RT fails to respond, the no_response bit is set and the no_response2 bit is clear. If the transmitting RT responds but the receiving RT fails to respond, both the no_response and the no_response2 bits are set. This might seem counterintuitive, but was done this way for backward compatibility with existing software.

bad status address

This bit is set when the address bits in the status word do not match the address bits in the command word.

bus A or B

This bit is clear if the command word was received on bus A, and it is set if the command word is received on bus B.

no inter-message gap

This bit is set when the bus dead time before a command word is less than 2.0 microseconds, with ½ microsecond accuracy. This translates to an inter-message gap time of less than four microseconds, measured from the mid-zero crossing of the parity bit of the last word of the last message and the mid-sync time of the current command word. Resolution is ½ microsecond. Therefore, it is not guaranteed to get set if the gap is between 3.5 and 4.0 microseconds.

end of message

This bit is set after the end of the message has been transmitted or received on the 1553 bus and is used by the API.

broadcast message

This bit is set if the message is a broadcast message, indicated by the RT address bits in the command word of 11111 when the RT_31_broadcast bit in the 1553 Control register is set.

RT-to-RT message format

This bit is set if the message is an RT-to-RT message.

reset RT

This bit is set when a valid reset remote terminal mode command is received (MIL-STD-1553B only; not defined for MIL-STD-1553A). The RT must reset to a power up initialized state. Upon detection, it is up to the driver software to initialize the status word, clear the RT_last_command_word, and set the RT_message_pointers, and RT_message_status words to their initialized states. In sRT (RT Validation) mode, the reset remote terminal and broadcast reset remote terminal mode commands enable both Bus A and Bus B for the RT address selected for RT Validation in the sRT_Address register.

self test

This bit is set upon reception of a initiate self-test mode command (MIL-STD-1553B only; not defined for MIL-STD-1553A).

mode code

This bit is set if the message is a mode code.

no command detected

This bit has been deprecated and should no longer be used.

This bit is set if the BC is being simulated and has sent a command word to the encoder, yet the command is not detected by the decoder.

The logic detection mechanism for the missing command is as follows. When the 1553 encoder transmits a command word, it monitors a feedback line from the decoder that a valid sync plus two data bits have been detected. If this signal is not present by the end of the encoder's serial shifting out the command word, the no command detected bit is set and the message is complete. Also, an interrupt is posted in the interrupt queue, if enabled.

When the BC's receiver does not see the command word, the time before the next message is transmitted is 20 microseconds for command word transmission, plus the programmed response time (default 16 microseconds), plus the programmed inter-message gap time. For example, with an inter-message gap time programmed for 15 microseconds, the delta time between the failed command word and the next command word (which may be a retry on the same or alternate bus, or the next message in the minor frame), is 20 + 16 + 15, or 51 microseconds, with an accuracy of one microsecond. This method guarantees that there is not any frame overrun when the BC does not see its own command word.

RT-to-RT message format error

This bit is set if the transmit command word of an RT-to-RT message has an invalid Manchester or parity bit error or if the receive/transmit command word combo does not meet MIL-STD-1553 specifications.

retry occurred

This bit is set by the hardware when an automatic retry occurs. This bit does not reflect pass/fail of the retry. Failure of the retry causes a no_response error or message_error bit to be set in the status word.

message error bit

This bit is set if an error in the message is detected which causes the message error bit at bit 10 of the 1553 status word to be set.

trigger begin, trigger end, bus monitor overflow

These three bits are used by the API for Bus Monitor functions. The microcode clears these bits at the end of every message.

The API vbt_notify routine uses the bus_monitor_overflow bit to determine if the hardware writing new messages has caught up with its ability to read the messages before the buffer overflows. When processing the interrupt queue, it sets the bus_monitor_overflow bit for the last message entry in the queue. Sometime later, it checks to make sure that the bit is still set. If not, it determines that the hardware has wrapped around and filled the entire buffer. It then skips the whole buffer to catch up and reports a "BM Overflow" condition.