

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

CONDOR ENGINEERING

MIL-STD-1553

Universal Core Architecture Reference Manual



Supporting Products:

- PCI-1553
- PMC-1553
- cPCI-1553
- ISA-1553
- QPMC-1553
- PCCARD-1553
- IP-1553
- PC-1553-IP
- VME-1553
- IP-D1553
- PCI-1553-IP
- cPCI-1553-IP
- VXI-1553-IP

Copyrights

Software Copyright © 1994-2002 Condor Engineering, Inc.
MIL-STD-1553 Universal Core Architecture Reference Manual Copyright © 2001 Condor Engineering, Inc.

This software product is copyrighted and all rights are reserved. The distribution and sale of this product are intended for the use of the original purchaser only per the terms of the License Agreement.

This MIL-STD-1553 Hardware Reference Manual is copyrighted and all rights are reserved.

This document may not, in whole or part, be; copied; photocopied; reproduced; translated; reduced or transferred to any electronic medium or machine-readable form without prior consent in writing from Condor Engineering, Inc.

Microsoft is a registered trademark of Microsoft Corporation
Windows is a registered trademark of Microsoft Corporation

MIL-STD-1553 Universal Core Architecture Reference Manual (1600100-0001)

IP-1553 Revision:	07
PMC-1553 revision:	04
PCI-1553 Revision:	04
cPCI-1553 Revision:	04
ISA-1553 Revision:	01
VME-1553 Revision:	03
PCCARD-1553 Revision:	3.40
QPMC-1553 Revision:	3.40
IP-D1553 Revision:	3.40
Document Revision:	3.40
Document Date:	17 June 2002

Condor Engineering, Inc.
101 W. Anapamu Street
Santa Barbara, CA 93101
(805) 965-8000
(805) 963-9630 (fax)
support@condoreng.com (email)

http://www.condoreng.com

Contents

Chapter 1 Introduction

PCI-1553 / cPCI-1553 / PMC-1553 / ISA-1553 / PCCARD-1553/ Interface Description 2
VME/VXI-1553 Interface Description 4
IP-1553 Interface Description 5
QPMC-1553 and IP-D1553 Interface Description 6

Chapter 2 Host Interface Registers

Host Interface Registers 9
- control_status_10K_configuration Register (CSC) (byte offset 0x00-0x03) 10
- 10K_configuration_data (byte offset 0x04-0x05) 15
- Bus1_page_register (byte offset 0x08-0x09) and bus2_page_register (byte offset 0x0e-0x0f) 16
- Jumpers/revision_id (byte offset 0x0a-0x0b) 16
- irq_enable_register (byte offset 0x0c-0x0d) 18
- Undefined 18

Chapter 3 VME/VXI-1553 A16 Configuration Register

A16 Configuration Registers 19
- ID Register (byte A16 offset 0x00-0x01) read only 20
- Logical Address Register (byte A16 offset 0x00-0x01) write only 21
- Device Type Register (byte A16 offset 0x02-0x03) read only 21
- Status Register (byte A16 offset 0x04-0x05) read/write only 22
- Offset Register (byte A16 offset 0x06-0x07) read/write 23
- HIU Control Register (byte A16 offset 0x08-0x09) read/write 23
- APEX Configuration Register (byte A16 offset 0x0a-0x0b) read/write 25
- Config-Data-Write (byte A16 offset 0x0c-0x0d) read/write 26
- Status/ID Ch1 (byte A16 offset 0x0e-0x0f) read/write 27
- Status/ID Ch2 (byte A16 offset 0x10-0x11) read/write 27
- Status/ID Ch3 (byte A16 offset 0x12-0x13) read/write 27
- Status/ID Ch4 (byte A16 offset 0x14-0x15) read/write 28

Host I/F Version (byte A16 0x16-0x17) read 28
VME-1553 Jumpers 28

Chapter 4 IP Module I/O Register and Interrupt Vector

I/O Register 31
Interrupt Vector 32

Chapter 5 Writeable Control Store

Writeable Control Store (WCS) 33

Chapter 6 Hardware Register

Hardware Register Summary 36
1553_control_reg (bytes 0x00-0x01) 36
time tag counter control reg (bytes 0x02-0x03) - write only 41
clear_tag (bytes 0x04-0x05) - write only 43
Playback Control/status Register (bytes 0x06-0x07) 43
response_reg (byte offset 0x08-0x09), reserved (byte offset 0x0a-0x0b) - write only 45
bc_enable_external_sync (byte offset 0x0c-0x0d) - write only 45
Clear Playback Status (byte offset 0x0e - 0x0f) 46
variable_volt (byte offset 0x10 - 0x11) 46
LPU Revision Register (byte offset 0x12 - 0x13) 47
interrupt_vector (byte offset 0x14) 47
write_interrupt_bit (byte offset 0x16-0x17) 48
Time Tag Counter (bytes 0x18-0x19) 48
minor_frame_timer (byte offset 0x1c-0x1d) 48
Time Tag Counter Load Register (bytes 0x20-0x25) 49
Time Tag Counter Increment Register (bytes 0x26-0x29) 49
Reserved Registers 50

Chapter 7 Register File Usage

Register File Summary (0x10 - 0xFF) 52
Bus Controller Registers 58
high priority (0x54 - 0x55) and low priority (0x56 - 0x57) aperiodic message pointers 58
high priority (0x58 - 0x59) and low priority (0x5A - 0x5B) aperiodic temporary message pointers 58
high priority (0x5C - 0x5D) and low priority (0x5E - 0x5F) aperiodic message list flags 58
high priority list bump normal flag (0x60 - 0x61) 59
bc_msg_pointer (0x80 - 0x81), bc_msg_pointer_save (0x82 - 0x83) and bc_msg_pointer_temp (0xA4 - 0xA5) 59

bc_data_pointer (0x84 - 0x85), bc_data_data_pointer_temp (0x62 – 0x63) 59
bc_interrupt_enables (0x86 - 0x89) 60
bc_retry_info (0x8A - 0x8B) 61
out_word_count (0x9C - 0x9D) 62
bc_flags (0xF0 - 0xF1) 62
bc_control_save (0xF4 - 0xF5) 64
bc_error_inject_pointer_save (0xF6 - 0xF7) 64
Remote Terminal Registers 64
Rt_disables (0x68 - 0x6F) 64
broadcast command save (0x4C - 0x4D) 64
rt_last_status_word_pointer (0x50 - 0x51) 65
rt_status_word_pointer (0x52 - 0x53) 65
rt_message_pointer_save (0x94 - 0x95) 65
rt_message_pointer (0x92 - 0x93) 65
rt_message_pointer1 (0x44 - 0x45) 66
rt_message_pointer2 (0x96 - 0x97) 66
rt_control_pointer (0x90 - 0x91) 66
owcnt (0x9E - 0x9F) 66
rt_error_inject_pointer_save1 (0xA0 – 0xA1) and rt_error_inject_pointer_save2 (0xA2 - 0xA3) 66
rt_address_buffer_base_address (0xA8 - 0xA9) 67
rt_filter_base_address (0xAA - 0xAB) 67
rt_enables (0x68 - 0x6F) 67
rtmonitr (0x78 - 0x79), rtmonitt (0x7E – 0x7F) 67
Bus Monitor Registers 68
bm trigger event type (0x48 - 0x49) 68
bm_message_pointer (0x9A - 9B), bm_message_pointer_save (0x98 - 0x99) 68
bm_filter_base_address (0xAC - 0xAD) 69
endflags (0xF2 - 0xF3) 69
bm_trigger_event_pointer (0xAE - 0xAF) 69
Playback Registers 70
playback start pointer (0x20 - 0x21) 70
playback end pointer (0x22 - 0x23) 70
playback tail pointer (0x24 - 0x25) 70
playback head pointer (0x26 - 0x27) 70
playback threshold word count (0x28 - 0x29) 70
playback interrupt status (0x2A - 0x2B) 70
playback interrupt threshold count (0x2C - 0x2D) 71
playback current word pointer (0x2E - 0x2F) 71
out_word_count (0x9C - 0x9D) 71
pb_error_inject_pointer_save (0xF6 - 0xF7) 71

Multi-Purpose Registers 72
command_word1 (0x40 - 0x41) and command_word2 (0x42 - 0x43) 72
interrupt_queue_pointer (0xA6 - 0xA7) 72
message_status1 and message_status2 (0xB2 - 0xB5) 72
orphaned_hardware_bits (0xB6 - 0xB7) 73
undefined_mode_code_table[3..0] (0xB8 - 0xBF) 75
Temporary registers (0x4E - 0x4F, 0xEC – 0xED and 0xEE - EF) 76
eom_flag (0x46 - 0x47) 76
word sequencer bits (0xF8 - 0xF9) 76
decoder word (0xFA - 0xFB) 77
temporary message status (0xFC - 0xFD) 77
constants 77

Chapter 8 RAM Usage

19-Bit RAM Pointers 80
RT Memory Usage 82
rt_address_buffer 83
rt_filter_buffer 87
rt_control_buffer 88
rt_message_buffer 92
BC Memory Usage 101
Aperiodic Message Lists 102
bc_message_block 104
bc_retry_buf 114
BM Memory Usage 115
bm_filter_buffer 116
bm_control_buffer 117
bm_message_buffer 120
bm_trigger_buffer 126
Playback 130
Message Code Format 130
1553 playback message packet format 131
Error Injection Buffer 132
error_injection_word 133
Interrupt Queue 135
interrupter_mode 136
message_pointer 137
next_interrupt_pointer 137

Chapter 9 PCI Configuration Space

PCI Configuration Space 139
Vendor ID 139
Device ID 139
Command Register 140
Status Register 140
Revision ID 140
Class Code 140
Subsystem ID 141
Subsystem Vendor ID 141
Base Address Registers 141
Interrupt Pin 141
Interrupt Line Register 141

Chapter 10 IP-1553 and IP-D1553 ID PROM

ID PROM 143

Chapter 11 IRIG

IRIG Memory Space 145
Time Tag Counter Control Register 145
IRIG DAC 145
IRIG Control Register 146
IRIG Time of Year (TOY) 146

CHAPTER 1

Introduction

This document describes Condor Engineering's powerful Universal Core Architecture for MIL-STD-1553. It is written for those who don't wish to use the API library and intend to write their own low-level driver instead. This Core Architecture is the low-level hardware interface that is common to many of Condor's MIL-STD-1553 interfaces including the PCI-1553, cPCI-1553, PMC-1553, ISA-1553, PCCARD-1553, VME-1553, IP-1553, QPMC-1553, IP-D1553 and derivative products. Provided with these products is an easy-to-use, high-level programmer's library called BusTools/1553-API. This library is a software layer written on top of the Core Architecture. For a description of the API usage and hardware installation instructions, see "BusTools/1553-API User's Manual and Hardware Installation Guide".

The following information can be found in this manual:

- Chapter 1 contains a separate interface description for the PCI-1553, cPCI-1553, PMC-1553, VME-1553, ISA-1553, PCCARD-1553, QPMC-1553, IP-D1553 and for the IP-1553 model boards.
- Chapter 2 describes the host interface registers for PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards.
- Chapter 3 describes using VME/VXI-1553 A16 Configuration Register with VME-1553 and VXI-1553 model boards.
- Chapter 4 describes the I/O register and Interrupt Vector for the IP-D1553 and the IP-1553 model boards.
- The remainder of the document applies to all models.

All addresses described in this document are BYTE addresses. All boards accept 16-bit words as well as byte accesses. Since all BYTE addresses must be EVEN, you can convert from a BYTE to a WORD address by dividing the BYTE address by two (2). Word format is Little Endian.

PCI-1553 / cPCI-1553 / PMC-1553 / ISA-1553 / PCCARD-1553/ Interface Description

The PCI-1553 is sold in four configurations, all of which are documented in this manual:

- PCI-1553-M Single-Channel multi-function 1553 module
- PCI-1553-MM Dual-Channel multi-function 1553 module
- PCI-1553-S Single-Channel single-function 1553 module
- PCI-1553-SS Dual-Channel single-function 1553 module

The cPCI-1553 is sold in four configurations, all of which are documented in this manual:

- cPCI-1553-M Single-Channel multi-function 1553 module
- cPCI-1553-MM Dual-Channel multi-function 1553 module
- cPCI-1553-S Single-Channel single-function 1553 module
- cPCI-1553-SS Dual-Channel single-function 1553 module

The PMC-1553 is sold in two configurations, both of which are documented in this manual:

- PMC-1553-M Single-Channel multi-function 1553 module
- PMC-1553-S Single-Channel single-function 1553 module

The ISA-1553 is sold in four configurations, all of which are documented in this manual:

- ISA-1553-M Single-Channel multi-function 1553 module
- ISA-1553-MM Dual-Channel multi-function 1553 module
- ISA-1553-S Single-Channel single-function 1553 module
- ISA-1553-SS Dual-Channel single-function 1553 module

The PCCARD-1553 is sold in two configurations, both of which are documented in this manual:

- PCCARD-1553-M Single-Channel multi-function 1553 module
- PCCARD-1553-S Single-Channel single-function 1553 module

The multifunction modules can run as the Bus Controller, Bus Monitor, *and* up to 31 Remote Terminals at the same time. The single function module can run only one of these functions at a time.

The PCI-1553, cPCI-1553, and PMC-1553 boards use a 256-byte PCI configuration space and a 32M byte PCI bus 32-bit memory address space. No I/O address space is used. PCI address line *a24* determines the 1553

bus number; *a23* is not used and should be set to zero when accessing the board; *a[22..20]* determine one of eight 1M byte functions (host interface registers, RAM, LPU hardware registers, LPU file registers, WCS), and *a[19..0]* address specific bytes within the function's memory space.

Note: The hardware derives *a[1..0]* from the PCI *CBE#[3..0]* byte enable signals. Upper address lines are ignored by the hardware.

The ISA-1553 board uses 16K bytes of the "high memory" area of the first 1M byte of ISA memory address space. No I/O ports are used. ISA address lines *a[23..14]* are used to access the board: *a13* determines the 1553 bus number; *a[12..11]* determine one of four, 2K byte functions (host interface registers, RAM, LPU hardware/file registers, WCS), and lines *a[10..0]* are used in conjunction with a page register to address specific bytes within the function's memory space.

ISA address line *a10* is also used to determine whether a hardware register or a file register is being accessed. The page register supplies the upper address lines *[19..11]* to RAM and WCS so that up to 1M byte may be accessed for each function. Host Interface Registers, WCS, LPU Hardware Registers, File Registers and RAM usage are described in separate chapters.

The PCCARD-1553 product occupies 8K bytes of PCCARD memory space. Memory mapping and functionality are the same as the ISA-1553 board with the following differences:

- It has only one channel.
- The FPGA configuration is accomplished on the card; it is not loaded by the host.
- The Writeable Control Store (WCS) is part of the FPGA configuration data and does not need to be loaded by the host.

PCI Byte Address a[24..0]	ISA Byte Address a[13..0]	PCCARD Byte Address a[13..0]	Function
0000000-00FFFFF, 1000000-10FFFFF	0000-07FF, 2000-27FF	0000-07FF, 2000-27FF	Host Interface Registers
0100000-01FFFFF, 1100000-11FFFFF	N/A	N/A	Undefined
0200000-02FFFFF	0800-0FFF (PAGED)	0800-0FFF (PAGED)	Bus 1 RAM (512K words)
0300000-03FFFFF	N/A	N/A	Bus 1 RAM (reserved for expansion)
0400000-04FFFFF	1000-13FF	1000-13FF	Bus 1 LPU Hardware Registers
0500000-05FFFFF	1400-17FF	1400-17FF	Bus 1 Register File
0600000-06FFFFF	1800-1FFF	N/A	Bus 1 WCS (1st 64K words used;

PCI Byte Address a[24..0]	ISA Byte Address a[13..0]	PCCARD Byte Address a[13..0]	Function
	(PAGED)		rest repeats)
0700000-07FFFFF	N/A	N/A	Bus 1 Undefined
1200000-12FFFFF	2800-2FFF (PAGED)	N/A	Bus 2 RAM (512K words)
1300000-13FFFFF	N/A	N/A	Bus 2 RAM (reserved for expansion)
1400000-14FFFFF	3000-33FF	N/A	Bus 2 LPU Hardware Registers
1500000-15FFFFF	3400-37FF	N/A	Bus 2 Register File
1600000-16FFFFF	3800-3FFF (PAGED)	N/A	Bus 2 WCS (1st 64K words used; rest repeats)
1700000-17FFFFF	N/A	N/A	Bus 2 Undefined

PCI accesses to undefined address locations are valid PCI transactions. The memory base address register (BAR) and the Memory Access Enable bit in the Command Word in PCI configuration space must be programmed before accessing memory. Memory addresses are 32M byte offsets of 32-bit PCI address space, as programmed in the BAR.

ISA and PCCARD accesses to undefined address locations will be valid ISA/PCCARD transactions. The page register must be set-up prior to accessing RAM and WCS.

VME/VXI-1553 Interface Description

The VME-1553 and VXI-1553 are sold in six configurations, all of which are documented in this manual.

- VME/VXI-1553-M Single-Channel multi-function 1553 module
- VME/VXI-1553-MM Dual-Channel Multi-function 1553 module
- VME/VXI -1553-MMMM Quad-Channel Multi-function 1553 module
- VME/VXI -1553-S Single-Channel single-function 1553 module
- VME/VXI -1553-SS Dual-Channel single-function 1553 module
- VME/VXI -1553-SSSS Quad-Channel single-function 1553 module

The multifunction modules can run as the Bus Controller, Bus Monitor, *and* up to 31 Remote Terminals at the same time. The single function module can only run one of these functions at a time.

The VME/VXI-1553 uses 8M bytes of A24 or A32 memory. There are also 64-bytes of A16 configuration registers. On board jumpers determine the address of the A16 registers. The factory default address for the A16

base address is C3C0. The starting A16 base address is (ID * 0x40) + 0xC000. ID is the Identification number of "this" device (0x00 through 0xFF) set by the jumpers.

VME/VXI-1553 Byte Address Offset	Function
000000-0007FF	Not Used
000800-000FFF	Bus 1 LPU Hardware Registers
001000-001FFF	Bus 1 Register File
100000-1FFFFF	Bus 1 RAM (1 Mbyte is currently used)
200800-200FFF	Bus 2 LPU Hardware Registers
201000-201FFF	Bus 2 Register File
300000-3FFFFF	Bus 2 RAM (1 Mbyte is currently used)
400800-400FFF	Bus 3 LPU Hardware Registers
401000-401FFF	Bus 3 Register File
500000-5FFFFF	Bus 3 RAM (1 Mbyte is currently used)
600800-600FFF	Bus 4 LPU Hardware Registers
601000-601FFF	Bus 4 Register File
700000-7FFFFF	Bus 4 RAM (1 Mbyte is currently used)

IP-1553 Interface Description

The IP-1553 is sold in two configurations, both of which are documented in this manual:

- IP-1553-M Multi-function 1553 module
- IP-1553-S Single-function 1553 module

The only difference between the configurations is that the multifunction module can simulate the Bus Controller, Bus Monitor, and up to 31 Remote Terminals at the same time. The Playback function is not implemented on the IP-1553. The single function module can support either Bus Controller, Bus Monitor, or up to 31 Remote Terminals. The modules are programmed identically except that the BC, BM, and RT enable bits in the 1553_control_reg register can be set for only one mode at a time if you have the IP-1553-S module. All modes can be started concurrently if you have purchased the IP-1553-M module.

The IP-1553 board requires 2K-byte contiguous address space in the host bus memory map, interrupt vector space and a single 16-bit I/O port.

Memory consists of write-able control store (WCS), RAM, file registers and hardware registers. All memory is paged into 2K byte (1K word) blocks. Upper address lines are ignored by the hardware. The WCS consists of 16K 48-bit words, which are downloaded by the software. The RAM is 64K 16-bit words deep. Hardware registers are mapped into the

lower 16-word address space of RAM, making these 16 words of RAM unusable. The file registers are mapped into the next 112 words of RAM and are physically part of RAM, but are treated separately because they have fixed addresses in hardware.

Memory Accessed	I/O Register bits 14-13	I/O Register bits 10..4	Byte Address	Comments
WCS				
48-bit word 0000	1	0	000-007	least significant byte at 000; 7th and 8th bytes reserved and are read only
48-bit word 0001	1	0	008-00f	"
48-bit word 3FFF	1	3F	7F8-7FF	"
Hardware Registers				
1553_control_reg	0	0	000-001	Read back at address 0000-000F
response timer	0	0	008-009	Write only
tag timer	0	0	010-012	Write to either address clears timer; cannot Read or Write timer value.
File Registers	0	0	020-0FF	Fixed RAM locations.
RAM				
Page 0 (1st K)	0	0	100-7FF	First 128 addresses not available.
Pages 1-3F	0	1-3F	000-7FF	All 63K available

QPMC-1553 and IP-D1553 Interface Description

The QPMC-1553 is sold in six configurations, all of which are documented in this manual:

- QPMC-1553-1M Single-Channel multi-function 1553 module, fixed voltage PMC interface
- QPMC-1553-2M Dual-Channel multi-function 1553 module, fixed voltage PMC interface
- QPMC-1553-4M Quad-Channel multi-function 1553 module, fixed voltage PMC interface
- QPMC-1553-1S Single-Channel single-function 1553 module, fixed voltage PMC interface
- QPMC-1553-2M Dual-Channel single-function 1553 module, fixed voltage PMC interface
- QPMC-1553-4S Quad-Channel single-function 1553 module, fixed voltage PMC interface

In addition, the above configurations may have an optional "-D" suffix for Rear I/O (P14) configuration and/or a "-W" suffix for IRIG Receiver/Transmitter. Other configurations are also available by special order.

The IP-D1553 is sold in four configurations, all of which are documented in this manual:

- IP-D1553-1M Single-Channel multi-function 1553 module
- IP-D1553-2M Dual-Channel multi-function 1553 module
- IP-D1553-1S Single-Channel single-function 1553 module
- IP-D1553-2M Dual-Channel single-function 1553 module

The multifunction modules can run as the Bus Controller, Bus Monitor, *and* up to 31 Remote Terminals at the same time. The single function module can run only one of these functions at a time.

The memory maps for the modules are shown in the following table, but the software accesses the modules through a carrier board; you need to reference the appropriate manual for the carrier board for proper memory access. For further information, refer to the BusTools1553-API Software User Manual and Hardware Installation Guide.

QPMC/IP-D1553 Byte Address Offset	Function
000000-0007FF	Host Interface Registers
000800-000FFF	Bus 1 LPU Hardware Registers
001000-001FFF	Bus 1 Register File
002000-002FFF	ID ROM (shadow)**
100000-1FFFFF	Bus 1 RAM (1 Mbyte is currently used)
200800-200FFF	Bus 2 LPU Hardware Registers
201000-201FFF	Bus 2 Register File
300000-3FFFFF	Bus 2 RAM (1 Mbyte is currently used)
400800-400FFF	Bus 3 LPU Hardware Registers*
401000-401FFF	Bus 3 Register File*
500000-5FFFFF	Bus 3 RAM (1 Mbyte is currently used)*
600800-600FFF	Bus 4 LPU Hardware Registers*
601000-601FFF	Bus 4 Register File*
700000-7FFFFF	Bus 4 RAM (1 Mbyte is currently used)*

* Not applicable to the IP-D1553 model boards.

** Not applicable to the QPMC-1553 model boards. This allows reading ID PROM from memory space. Also readable at 202000-202FFF, 402000-402FFF and 602000-602FFF.

In addition to memory space, the IP-D1553 also provides PROM ID space and Interrupt Vector Space, both of which may be mapped into memory space by the carrier board. IO Space is not used but may be assigned by the carrier board.

CHAPTER 2

Host Interface Registers

This chapter applies to all model boards except the IP-1553 models, which do not contain Host Interface Registers.

Host Interface Registers

Host Interface Registers are the memory address space registers that physically reside in the host interface PLD chip. This chip is loaded from Flash EPROM when power is first applied. These registers are board-level and affect all channels. They consist of the CSC Register, 10K Configuration Data, Page Registers (ISA-1553/PCCARD-1553 only), Jumpers/Revision Register (ISA-1553 only), and Interrupt Request Enable Register (ISA-1553/PCCARD-1553 only). The PCI-1553, cPCI-1553, and PMC-1553 boards contain PCI configuration space that determines the model and revision number. See Chapter 8, “PCI Configuration Space“ for further details.

The QPMC-1553 and IP-D1553 boards contain only a single host interface register, the CSC Register, which is not part of a separate PLD as it is in other model boards. In addition, the QPMC-1553 contains IRIG DAC and Control Register Space (see Chapter 11, “IRIG”).

PCI Byte Address a[24..0]	ISA Byte Address a[13..0]	PCCARD Byte Address a[13..0]	Read	Write	Function
0000000-0000003, 1000000-1000003	0000-0003, 2000-2003	0000-0001	Yes (part)	Yes (part)	CSC Register
0000004-0000007, 1000004-1000007	0004-0007, 2004-2007	N/A	No	Yes	10K Configuration Data***
N/A	0008-0009, 2008-2009	0008-0009	Yes	Yes (part)	Bus 1 Page Register*
N/A	000a-000b, 200a-200b	N/A	Yes	No	Jumpers/Rev. # **
N/A	000c-000d, 200c-200d	N/A	Yes	Yes	IRQ Enable Reg.*
N/A	000e-000f, 200e-200f	N/A	No	Yes	Bus 2 Page Register*
0000008-00fffff, 1000008-10fffff	0010-07ff, 2010-27ff	0010-07ff	Undefined	Undefined	Undefined

* Applies to ISA-1553/PCCARD-1553 model boards only.

** Applies to ISA-1553 model boards only.

*** Applies to PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards only.

control_status_10K_configuration Register (CSC) (byte offset 0x00-0x03)

The CSC Register is used by the application to reset the bus, determine board ID and configuration status, and configure the 10K programmable chips (local processing unit) used for each 1553 bus. All bits are active "1" unless otherwise stated. This register powers up with all zeros (except board ID bits). The application must set some of these bits to access the rest of memory space. PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards have a different format than QPMC-1553 and IP-D1553 model boards.

PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 Model Boards

Word 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
bt[1..0]		pr2	pr1	f2	f1	m2	m1	sb2	sb1	su2	su1	rsvd	rst2	rst1	rst0

Byte 0: Reset/Power Control

rst0	7K reset (write only)
rst1	reset bus 1 LPU (write only)
rst2	reset bus 2 LPU (write only)
rsvd	reserved
su1	bus 1 WCS set-up complete
su2	bus 2 WCS set-up complete
sb1	bus 1 in standby power mode
sb2	bus 2 in standby power mode

Byte 1: Board ID bits

m1	bus 1 multiple mode (read only)
m2	bus 2 multiple mode (read only)
f1	bus 1 1553/1773 (read only)
f2	bus 2 1553/1773 (read only)
pr1	bus 1 present (read only)
pr2	bus 2 present (read only)
bt[1..0]	host bus type ID (read only)

7K reset

Writing a logic one to this bit resets the local (on-board) bus controller (LBC) chip and all of the registers and latches in the host interface chip. This affects the control logic for both 1553 buses. The reset LPU bits (see below) should be set simultaneously. The WCS set-up complete bits should be cleared simultaneously for a full board reset.

WCS and local memory should not be affected by a reset, but it is recommended that all memory be reprogrammed after a board reset. The hardware reset remains active only for 50 nanoseconds, so this bit can't be read back by the host (logic "0" is returned). The standby power mode bit for the corresponding bus should always be clear prior to setting this bit.

This bit is not applicable to the PCCARD-1553.

Reset LPU

Writing a logic one to this bit resets the LPU (encoder/decoder logic of the 1553 bus). There is a bit for each 1553 bus. The LPU for one bus can be reset if one bus is in an unknown state but the other bus is to remain running. Normally, both reset LPU bits are set simultaneously, with the 7K reset bit to put the board in a reset state.

WCS and local memory should not be affected by this reset, but it is recommended that you reprogram all memory after a board reset. The LPU may be reset in standby power mode or in regular mode. The hardware reset only remains active for 50 nanoseconds so this bit cannot be read back by the host (logic "0" is returned).

This bit is not applicable to the PCCARD-1553. Any reset to the card should be accomplished through the PC Card Services.

WCS set-up complete

The application may program the WCS only with this bit off and must set this bit after the WCS programming is complete. It isn't permissible to set this bit prior to programming the WCS. The bc_run, rt_run, and bm_run bits in the 1553_control_reg should all be turned off prior to setting or clearing this bit. There is a bit for each 1553 bus.

This bit is not applicable to the PCCARD-1553 since the WCS is part of the on board FPGA configuration PROM data.

standby power mode

An active "0" on this bit turns off the clock to the 1553 local processor (LPU), thus placing the chip in standby power mode (10K chip consumes only 1 mA). A logic "0" is the default on power-on (or host bus hardware reset). To access the LPU registers and WCS, this bit must be set. There is a bit for each 1553 bus.

This bit is not applicable to the PCCARD-1553.

multiple mode

These read-only bits inform the application if the 1553 bus has multiple-mode capability (logic "1") or single mode capability (logic "0"). Boards purchased with multiple-mode capability have "M" suffix(es) on their part number; those with single mode have "S" suffix(es). Multiple mode boards can act as BC, BM, and up to 32 RT's simultaneously. Single mode boards can function only as a BC, BM, or up to 32 RT's. For single mode boards, if two or more of the three run bits in the 1553_control_reg are set, then all three run bits are cleared and the board can't talk to the 1553 bus.

MIL-STD-1553/1773

These read-only bits inform the application if the bus is a MIL-STD-1553 (logic "1") or a MIL-STD-1773 (logic "0") interface. The application may use this information for GUI text, such as for supplying information when you select "About BusTools".

bus present

These read-only bits inform the application which buses are physically present (logic “1”). Single MIL-STD-1553 bus boards have only the “bus 1 present” bit set.

host bus type ID

These read-only bits allow hardware identification of the host bus model board installed in the system.

The following values for bt[1..0] are assigned:

00	PCI / PMC / cPCI 32-bit bus
01	ISA 16-bit bus
10	PCCARD 16-bit bus
n	VME-1553

Word 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
und											sdat	cfr	cfd	cfs	cfm

Byte 2: 10K Configuration control/status

cfm	10K configuration mode
cfs	10K configuration status bit (read only)
cfd	10K configuration done (read only)
cfr	10K configuration ready (read only)
sdat	Serial data for 10K configuration (read only)
und	Undefined (hardware returns 0’s on read)

Byte 3: Undefined

und	Undefined (hardware returns 0’s on read)

It is the responsibility of the application to configure the Altera FLEX 10K chips used for the LPU’s. The hardware uses a “passive serial (PS) configuration” generated by the host interface chip. The application controls configuration by downloading a raw binary file (lpu.rbf) to the board on the LSB of the “10K configuration data” address. 10K configuration is required prior to programming the WCS, accessing the LPU registers and running the 1553 bus. The process is as follows:

1. The application clears the “10K configuration mode” bit and then sets it not less than two microseconds later.
2. The application writes each bit of the lpu.rbf file, LSB first, to the “10K configuration data” address. The hardware guarantees that the maximum clock rate of 10 MHz is not exceeded.

3. The application continues writing serial bits until “10K configuration done” goes active. This should happen after the last bit in lpu.rbf is written as the file should not have any header information in it.
4. The application needs to do 10 more “dummy” writes to the “10K configuration data” address to satisfy a hardware requirement. The data is a “don’t care” for these 10 writes.
5. The application reads the “10K configuration status bit” to ensure that the LPU’s were programmed without errors. The bit should be high. If not, the application should make a second attempt.

10K configuration mode

A low resets the 10K device. A low to high begins configuration. The bit should be held low a minimum of two microseconds prior to configuration.

10K configuration status

This read-only bit goes low immediately after power-up and is set high within 100 ms. If an error occurs during LPU configuration, this bit goes low.

10K configuration done

This read-only bit goes low during LPU configuration. After all configuration data has been received without errors, the bit goes high.

10K configuration ready

Each configuration bit written by the application clears the “10K configuration ready” bit. The hardware sets the 10K configuration ready bit once the configuration bit has been clocked into the LPU. Generally, this bit does not need to be tested by the application, as the hardware is faster than the host bus.

Serial data for 10K configuration

This is the read-back of bit 0 written to the “10K Configuration data” address.

QPMC-1553 and IP-D1553 model boards

Word 0:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Undefined				mod	channel_num					board_type					rst0

reset

Writing a logic one to this bit will reset all registers on the board. This is to be used for board initialization only and the bit cannot be read back.

Board_type

These five read-only bits determine the board type. The following types have been assigned:

1	QPMC-1553
2	IP-D1553
0 or > 2	Invalid board type

channel_num

These five read-only bits determine the number of 1553 channels on the board. The following are possible values:

1	One 1553 channel
2	Two 1553 channels
4	Four 1553 channels

mode

This read-only bit determines if the board is single-mode (0) or multi-mode (1).

10K_configuration_data (byte offset 0x04-0x05)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
und															dat0

dat0	Serial data for 10K configuration
und	Undefined (not readable)

The application writes configuration data to bit 0. It can read the data back at bit 12 of the CSC register. Word 1 (bits 31- 16) is not implemented.

Bus1_page_register (byte offset 0x08-0x09) and bus2_page_register (byte offset 0x0e-0x0f)

The high memory space of the ISA bus is of limited size and therefore ISA-1553/PCCARD-1553 model boards must use paged memory. The page register is not used for PCI-1553, cPCI-1553 and PMC-1553 model boards. Refer to “IP Module I/O Register and Interrupt Vector” chapter for the IP-1553 model board’s page register description.

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

page

These bits determine the page of memory to be accessed. Model ISA-1553/PCCARD-1553 uses bits 8..0 to page down to 2K byte blocks of memory pages in order to access a 1M byte address space. On the ISA-1553 board bits 8..0 map to RAM and WCS address lines 19..11. Bits 15..9 are not used and should be programmed with all zeros. Bit 15 is the most significant bit.

Jumpers/revision_id (byte offset 0x0a-0x0b)

This read-only word allows the application to determine the ISA memory address window, MEM16 qualifying address and the board revision ID for ISA-1553 model boards.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
mem_addr					usa	rsvd		revision_id							

revision_id

These bits are the 8-bit revision ID beginning with 00 hex (prototype) and 01 hex (first production).

Jumpers

The most significant byte controls the ISA memory address window and the MEM16 qualifying address.

Bits 8 and 9 are reserved for future use.

Bit 10 is assigned to “usa” which forces hardware to decode the ISA sa[16..14] signals along with the la[23..17] signals as qualifiers for the MEM16 signal. MEM16 is the asynchronous handshake that informs the initiator that it can accept a 16-bit word transfer (PC/AT 16-bit board versus 8-bit XT board). The EISA/ISA specification does not allow sa[] signals to be used to qualify MEM16, as the adapters within a 128K boundary must all be 16-bit or all be 8-bit boards. In reality, many platforms do not follow the specification and we have found that qualifying MEM16 with sa[16..14] works best on the vast majority of PC’s. However, a very few PC’s delay the sa[] signals too long to qualify them for MEM16. Thus, the user may insert a jumper to disable the use of sa[16..14] qualifiers. When the bit is set, it will use sa[16..14] for MEM16.

Bits 15..11 select the memory address as shown in the table. A “1” represents an open; a “0” represents a jumper inserted. Schematic designators j[18..14] are read into bits 15..11, which represent ISA address lines 18..14, respectively.

ISA 16K-bit segment address range, a[19..4]	Mem_addr bits 15..11
a000-a3ff	01000
a400-a7ff	01001
a800-abff	01010
ac00-afff	01011
b000-b3ff	01100
b400-b7ff	01101
b800-bbff	01110
bc00-bfff	01111
c000-c3ff	10000
c400-c7ff	10001
c800-cbff	10010
cc00-cfff	10011
d000-d3ff	10100
d400-d7ff	10101
d800-dbff	10110
dc00-dfff	10111
e000-e3ff	11000
e400-e7ff	11001

irq_enable_register (byte offset 0x0c-0x0d)

This register is used in the ISA-1553 and PCCARD-1553 boards only.

This register must be programmed to allow the interrupt event to physically assert one of the ISA bus interrupt request lines (IRQ's). There are a number of events on which the application can enable the hardware to interrupt. These are all discussed in the hardware register, file register, and/or RAM memory usage sections.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
irq15	irq14	0	irq12	irq11	irq10	irq9	0	irq7	irq6	irq5	irq4	irq3	0	0	0

Setting a bit in this register, in addition to setting the "cpu interrupt enable" bit in the "1553 control register", enables the interrupt onto the ISA IRQ line. Eleven IRQ lines are hardwired to the board and any one may be used by the application. When this register is zero, all eleven signals are tri-stated (open-collector). The application may set any one bit in this register to allow the board to drive the ISA IRQ line. The interrupt is a low to high transition, and the interrupts for both 1553 buses should be cleared prior to enabling this register to flush out any residual interrupt request.

To clear the interrupt, write a zero to bit 9 of the hardware "write interrupt bit" register of the appropriate 1553 bus. The interrupt may apply to either 1553 bus; it is up to the application to interrogate the board to determine the interrupt source. This register may be read back and written to by the application. Enabling an IRQ, which is already in use by another resource, causes an IRQ conflict, which may "hang" your PC. Check system IRQ usage before enabling the IRQ here.

Undefined

Accessing undefined host interface registers results in a normal bus cycle, but the data is meaningless. Writes have no effect on the board. For the PCI-1553, cPCI-1553, and PMC-1553 boards, reading any double word in this memory space displays the CSC Register, but there is no guarantee of this except for reading from the assigned CSC Register address.

CHAPTER 3

VME/VXI-1553 A16 Configuration Register

This chapter applies to VME-1553 and VXI-1553 model boards. These two products are functionally identical, except for form factor. Although only the VME-1553 board is mentioned in this chapter, all of the information provided applies to the VXI-1553 board as well.

A16 Configuration Registers

The VME-1553 A16 configuration registers are the memory address space registers that physically reside in the host interface PLD chip. These registers allow you to:

- Provide a method to configure the other FPGA's found on the board
- Specify where the board is mapped in A32 or A24 space
- Control the resetting of the local bus
- Enable hardware interrupts and their status
- Provide hardware/version information
- Provide the necessary information for VXI dynamic configuration to take place.

A16 Address offset from Base Address	Read/Write	Register Function
0 – 1	R/W	ID/Logical Address
2 – 3	R	Device Type
4 – 5	R/W	Status/Control
6 – 7	R/W	Offset
8 – 9	R/W	HIU Control
0xa –0xb	R/W	Configuration
0xc – 0xd	R/W	ConfigurationData
0xe – 0xf	R/W	Status/ID Ch1
0x10 – 0x1`	R/W	Status/ID Ch2
0x12 – 0x13	R/W	Status/ID Ch3
0x14 – 0x15	R/W	Status/ID Ch4
0x16 – 0x17	R/W	HIU Version

ID Register (byte A16 offset 0x00-0x01) read only

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dc[1..0]		as[1..0]		Manufacturer's ID											

dc[1..0] = 11b

Device Class bits. Both bits are set to one indicating the board is VXI Register based device. These bits are read only and may not be changed.

As[1..0] = 01b

Address Space bits. These bits are set to indicate that the addressing modes of the board operational registers are A16/A32. These bits are read only and may not be changed.

Manufacturer's ID = E94h

This is Condor Engineering's VXI Manufacturer's ID number as assigned by the VXI Bus Consortium. These bits are read only and may not be changed.

Logical Address Register (byte A16 offset 0x00-0x01) write only

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
not used								Logical Address [7..0]							

Logical Address [7..0]

After a system reset (when the plug-and-play option is selected), the Logical Address byte is set internally to 0xFF. This means that ID/Logical Address register responds at the A16 address of 0xFFC0. However, since this is a VXI function, the MODID backplane signal for the particular slot (there is a separate MODID signal for each card position within a VXI backplane) must also be active to be able to access the ID/Logical Address register.

Since all VXI plug-and-play cards respond to the same address, the normal procedure is to address each one in turn at 0xFFC0 (the MODID is the card discriminator) and write an 8-bit value that becomes the new base address. From that write forward, the new address is where this and the following registers appear in VME address space.

If the plug-and-play option is not selected refer to the "VME-1553 Jumpers" section at the end of this chapter for information regarding mapping the board via shorting jumpers.

Device Type Register (byte A16 offset 0x02-0x03) read only

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Required Memory[3..0]				Model Code[11..0]											

Required Memory[3..0]=08h

This is a read only register for use by VXI dynamic configuration. It indicates 8MB of A16/A32 space is required for a four-channel board (2MB/channel).

Model Code[11..0]= 15*n*h

These bits identify how many 1553 channels and whether the board supports single or multi-function 1553 operation. The "15" portion of the Model Code is short for "1553". The remaining nibble "*n*" is described below:

n = 0000b for single function, single channel (model code = 150h)
n = 1100b for multi function, single channel (model code = 15Ch)
n = 0001b for single function, two channel (model code = 151h)
n = 1101b for multi function, two channel (model code = 15Dh)
n = 0011b for single function, four channel (model code = 153h)
n = 1111b for multi function, four channel (model code = 15Fh)

Status Register (byte A16 offset 0x04-0x05) read/write only

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
a32	md	not used										rdy	psd	sfi	rst

Byte 0: Reset/Power Control

rst	local bus reset (1 = reset)
sfi	SYSFAIL* Inhibit (1= disable)
psd	Passed (read only)
rdy	Ready (read only)
md	MODID* status (0 = active, read only)
a32	A24/A32 Active bit status (1 = enabled)

rst

Local bus reset – A one (1) forces a local bus reset, resetting all of the 1553 channels. A zero (0) makes local bus reset inactive.

Sfi

SYSFAIL* Inhibit – A one (1) written in this field disables the device from driving the SYSFAIL* line

psd

Passed – Indicates the board has complete it's built-in self test. Since the VME/VXI1553 boards do not execute a built-in self-test, this bit is set to one once the host interface configuration is complete.

Rdy

Ready – A one in this bit indicates the board is ready to accept full operational commands. This bit is set to one once the host interface configuration is complete.

Md

MODID status – A one (1) indicates the device is not selected through the P2 MODID line. A zero (0) indicates the device is selected by a high state on the P2 MODID line.

A32

A24/A32 active – Setting this bit indicates the boards A24 or A32 registers can be accessed. Clearing this bit grants access to only the A16 registers.

Offset Register (byte A16 offset 0x06-0x07) read/write

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset[8..0]									not used						

Offset[8..0]

Defines the base address of the A32 VXI operational registers with dynamic configuration enabled. The msb's of this register become the msb's of A32 addresses. If operating in A24 VME space, Offset bit 0 (bit 7 of the register) becomes the msb of the A24 address. Sysreset* sets theses bits to zero (0).

HIU Control Register (byte A16 offset 0x08-0x09) read/write

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
User-Out[7..0]								hiur	LIRQ [2..0]			Mod	Int_level		

User Out[7..0]

Eight discrete general-purpose outputs. These LVTTL (3.3V) ouputs are current limited with 330Ω series resistors. They are accessible on the connector J9 as follows:

- User Out0 J9 pin-11
- User Out1 J9 pin-10
- User Out2 J9 pin-9
- User Out3 J9 pin-8
- User Out4 J9 pin-7
- User Out5 J9 pin-6
- User Out6 J9 pin-5
- User Out7 J9 pin-4

J9 pins 8 through 11 are used for factory test by default. The msel bit in Apex Configuration Register (A16 byte offset 0x0A-0x0B must be set to enable User_out[3..0] at J9).

The current status of User Out[7..0] is returned through a read of this register.

Hiur/LIRQ3

This bit has two separate functions:

- hiur (write only) – Host interface logic reset bit. Writing a zero to this bit re-initializes the HIU interface logic.
- LIRQ3 (read only) – The current state of the LIRQ_3 interrupt request from the local bus logic for 1553 channel 4. This bit is active high.

LIRQ[2..0]

Local bus IRQ bits 2 ..0 active (read only). These bits are don't-cares during writes. LIRQ3 status is read in bit[7] as described above. These bits are active high.

Mod

This bit selects whether the board responds to VME bus cycles in A24 or A32 space. Writing a 0 selects A24, and writing a 1 selects A32.

Int-Level

These bits determine which VME interrupt is generated by this particular card. Writing a value from 1 to 7 selects the interrupt level. Writing a 0 to these three Control register bits prevents the card from generating any VME interrupts.

APEX Configuration Register (byte A16 offset 0x0a-0x0b) read/write

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
not used						msel	n.u.	pcd[3..0]				n.u.	cfd	cfs	cfm

It is the responsibility of the application to configure the Altera APEX FPGA's used for the LPU's. The hardware uses a "passive serial (PS) configuration" generated by the host interface chip. The application controls configuration by downloading a raw binary file (lpu.rbf) to the board on the LSB of the "APEX configuration data" address. On the VME-1553 the WCS load is contained in the configuration data for the FPGA. The FPGA configuration process is as follows:

1. The application clears configuration mode bit and then sets it not less than two microseconds later.
2. The application writes each bit of the lpu.rbf file, LSB first, to the pld[3..0] bits (all four bits hold the same data
3. The application continues writing serial bits until configuration done goes active. This should happen after the last bit in lpu.rbf is written, as the file should not have any header information in it.
4. The application needs to do forty more "dummy" writes to the pld[3..0] address to satisfy a hardware requirement. The data is a "don't care" for these 40 writes.

The application reads the configuration status bit to ensure that the LPU's were programmed without errors. The bit should be high. If not, the application should make a second attempt.

Cfm

Configuration mode – A low resets the APEX device. A low to high begins configuration. The bit should be held low a minimum of two microseconds prior to configuration.

Cfs

Configuration status – This read-only bit goes low immediately after power-up and is set high within 100 ms. If an error occurs during LPU configuration, this bit goes low.

Cfd

Configuration done – This read-only bit goes low during LPU configuration. After all configuration data has been received without errors, the bit goes high.

Pcd[3..0]

Previous configuration data – Readback of the last APEX configuration data bits written.

Msel

Mux select – Must be set to enable user_out[3..0] at J9.

Config-Data-Write (byte A16 offset 0x0c-0x0d) read/write

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not Used (Write)												pld[3..0]			
Not Used (Read)								User-In[7..0]							

pld[3..0]

During a write, these four one-bit wide registers hold the data to be programmed into the (up to) four channels of FPGA's.

User-In[7..0]

During a read, bits [7:0] are the discrete USER_IN bits.

Eight discrete general-purpose inputs. These LVTTL (3.3V) inputs are current limited with 330Ω series resistors. They are accessible on the connector J9 as follows:

- User In0 J9 pin-19
- User In1 J9 pin-18
- User In2 J9 pin-17
- User In3 J9 pin-16
- User In4 J9 pin-15
- User In5 J9 pin-14
- User In6 J9 pin-13
- User In7 J9 pin-12

Status/ID Ch1 (byte A16 offset 0x0e-0x0f) read/write

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
not used								la1[7..0]							

la1[7..0]

This is a 16-bit value returned during the IACK VME sequence in response to LIRQ1 from the Local Bus. If multiple interrupts from the local bus are active simultaneously, this one has highest priority.

Status/ID Ch2 (byte A16 offset 0x10-0x11) read/write

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
not used								la1[7..0]							

la2[7..0]

This is a 16-bit value returned during the IACK VME sequence in response to LIRQ2 from the Local Bus. If multiple interrupts from the local bus are active simultaneously, this one has the second highest priority.

Status/ID Ch3 (byte A16 offset 0x12-0x13) read/write

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
not used								la1[7..0]							

la3[7..0]

This is a 16-bit value returned during the IACK VME sequence in response to LIRQ3 from the Local Bus. If multiple interrupts from the local bus are active simultaneously, this one has the third highest priority.

Status/ID Ch4 (byte A16 offset 0x14-0x15) read/write

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
not used								la1[7..0]							

la4[7..0]

This is a 16-bit value returned during the IACK VME sequence in response to LIRQ4 from the Local Bus. If multiple interrupts from the local bus are active simultaneously, this one has the lowest priority.

Host I/F Version (byte A16 0x16-0x17) read

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Version[15..0]															

Version[15..0]

The four nibbles of the 16-bit register are used for the four digits of the version number. Thus, version 00.07 would result in a register with contents 0x0007, while version 1.52 would result in 0x0152.

VME-1553 Jumpers

Header JB1 on the VME-1553 board is used to enable VXI plug-and-play operation and to set the address of the boards A16 region for VME/VXI systems.

PNP

When installed, this jumper disables the VXI plug-and-play feature. With the jumper removed, the VXI plug-and-play feature is enabled. This jumper should remain installed on all VME systems.

VXI_A[7..0]

If the VXI plug-and-play feature is disabled, these jumpers determine where the A16 portion of the board is placed by the following formula:

A16 base address = (VXI_A[7..0] * 0x40) + 0xC000

CHAPTER 4

IP Module I/O Register and Interrupt Vector

I/O Register

The I/O Register is used only for IP-1553 boards. A single sixteen-bit I/O register is accessed for all 64 I/O addresses.

I/O Word Offset	I/O Byte Offset	Register
0x0 – 0x40	0x0 – 0x80	i/o_register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Config Bits		0	1	Page Bits/PLD Status							0	0	0	reset

reset

Writing a 1 to this bit resets the board. All the board logic except the IP interface state machine is cleared. The RAM and WCS aren't cleared but should be re-initialized after using this function as the data might be corrupted. This function can be invoked only upon complete board initialization or if the board fails to respond due to a catastrophic failure such as loss of power. This bit is not readable by the host.

Page Bits/PLD Status

When the Config Bits equal 00 or 01 binary, these seven bits determine the 2K-byte page of memory to be accessed. Bit 10 is the MSB; bit 4 is the LSB. Up to 128 pages can be represented, however, only 64 pages are currently in use. These bits are fully accessible to the host.

When the Config Bits do not equal 00 or 01 binary, these bits are undefined.

Configuration Bits

These bits determine the level of configuration currently in progress. The host programs them during the board configuration. The board powers on in "Configure WCS" mode. Board configuration requires the WCS to be programmed first, then the registers and RAM must be initialized, and then the run bits may be set to begin BC, RT, or BM operation.

This process is referred to as "system setup" in other documentation. See the section on the 1553_control_reg for a description on the run bits. Bit 13 is also called "wcs_setup_complete" and is an active "0".

Value of Bits 14..13	Function
00	PLD and WCS Configuration Complete
01	Configure WCS
10	Configure PLD
11	Reserved

Interrupt Vector

The 8-bit interrupt vector is for IP-1553 and IP-D1553 boards only. The vector is optionally used by software. The interrupt vector may be read at any address defined as interrupt vector space. Refer to the "BusTools1553-API Software User Manual and Hardware Installation Guide" or the appropriate manual for the carrier board to determine the interrupt vector map space.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Undefined bits								Interrupt Vector[7..0]							

CHAPTER 5

Writeable Control Store

Writeable Control Store (WCS)

Each bus has a 64K x 48-bit Writeable Control Store (WCS). The WCS contains the microcode that runs the 1553/1773 hardware. The WCS is downloaded to the board from the host bus prior to running the 1553 bus.

The WCS is supplied as six 8192-deep, byte-wide Intel Intellec Hex files, which the API downloads onto the board during setup. The upper 56K x 48 bits of WCS aren't programmed. The API uses three 16-bit writes or two double word writes (allowed on 32-bit model boards) to program each 48-bit WCS word. Each WCS word is stored on an 8-byte boundary, least significant byte first. The seventh and eighth bytes of every eight bytes are not used. The software may write, read back and verify data at each of the other six bytes whenever the wcs_setup_complete bit is not set. The WCS may be read back to ensure data integrity. For PCI-1553, cPCI-1553, PMC-1553, ISA-1553 model boards, see the CSC Register description for use of the WCS Set-up Complete bit. For IP-1553 model boards, see the IO Register description for use of the WCS Set-up Complete bit.

The WCS for the PCCARD-1553, VME-1553/VXI-1553, QPMC-1553 and IP-D1553 is part of the on-board FPGA configuration PROM data that is loaded when the board is powered.

CHAPTER 6

Hardware Register

Several 16-bit hardware registers need to be set up before running Bus Controller, Bus Monitor, or Remote Terminal operations. They are mapped into word offsets 0x0 through 0x1F (byte offsets 0x00 through 0x3F). The base addresses for the different model boards are as follows:

Condor Base Model	Page	Base Byte Address
PCI-1553 (Bus 1), or cPCI-1553 (Bus 1) or PMC-1553 (Bus 1)	N/A	0400000
PCI-1553 (Bus 2), cPCI-1553 (Bus 2)	N/A	1400000
VME-1553, QPMC-1553, IP-D1553	N/A	0800
ISA-1553 (Bus 1) PCCARD-1553 (Bus 1)	0	1000
ISA-1553 (Bus 2)	0	3000
IP-1553	0	0000

Hardware address decoding may allow the register space to be repeated. Only access the registers using the base address plus the offset. All discrete bits are active "1" unless otherwise specified. Reserved or undefined bits should be programmed with logic zeros.

Hardware Register Summary

Memory Address Word Offset	Memory Address Byte Offset	Write Only	Register
0x0	0x00-0x01	R/W	1553_control_reg
0x1	0x02-0x03	W only	Time tag counter control reg
0x2	0x04-0x05	W only	Clear_tag_timer***
0x3	0x06-0x07	R/W	Playback control/status reg*
0x4	0x08-0x09	W only	Response_reg
0x5	0x0a-0x0b	N/A	Reserved for late response time-out
0x6	0x0c-0x0d	W only	Bc_enable_external_sync
0x7	0x0e-0x0f	W only	Clear playback status*
0x8	0x10-0x11	W only	Variable_volt **
0x9	0x12-0x13	R only	LPU Revision Number
0xA	0x14-0x15	W only	Interrupt_vector
0xB	0x16-0x17	W only	write_interrupt_bit
0xC	0x18-0x19	R/W	Read tag time counter; Write clears count Mux.
0xD	0x1a-0x1b	N/A	Reserved
0xE	0x1c-0x1d	W only	minor_ frame_timer
0xF	0x1e-0x1f	N/A	Reserved
0x10-0x12	0x20-0x25	W	Tag time counter load reg
0x13-0x14	0x26-0x29	W	Tag time counter increment reg
0x15-0x1f	0x26-0x3f	N/A	Reserved

* Not available on IP-1553 model boards.

** Available as an option on some models.

*** Only available on IP-1553 model boards.

1553_control_reg (bytes 0x00-0x01)

The host programs this register during system setup. All bits are active "1". Reserved bits should be programmed with logic "0".

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cpl	ien	exs	tb	sa31	rt31	int	iw	bex	1553 a	bcb	exto	bm-run	rt-run	bc-run	smw

Field	Definition
smw	single mode warning
bcrun	bc run
rtrun	rt run
bmrun	bm run
exto	external ttl output
bcb	bc busy
1553a	MIL-STD-1553A mode
bex	bc enable external sync (read only)
iw	internal wrap
int	interrupt
rt31	rt 31 broadcast
sa31	sub-address 31 mode code
tb	test bus enable (VME/VXI 1553 only)
exs	rt enable external sync
ien	cpu interrupt enable
cpl	1553 bus coupling

single_mode_warning

Boards that support only a single 1553 mode (BC, RT or BM) turn off all run bits if the software attempts to set two or more of the three bits at a time. The single_mode_warning bit is also be set to indicate that this has happened.

bc_run, rt_run, bm_run

Setting these three bits cause 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 (IP-1553-M, PCI-1553-M, etc.) support all three of these modes simultaneously. The single-function boards (IP-1553-S, PCI-1553-S, etc.) will only allow one of these bits to be set at a given time.

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

external_ttl_output

The external TTL output is the logical OR of this bit and of a signal generated by the microcode. Setting this bit forces the external TTL output to a high level. Clearing this bit allows the microcode to control the output.

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 (formerly last message) block and on Noop and Conditional message blocks if the end of minor frame bit is set.

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 bus controller registers and flags 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: With firmware revision 2.55 and beyond, the software is no longer able to write to the bc_busy bit.

1553a

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. The legal/illegal command word information in the rt_control_buffer must be modified accordingly.
- Sub-address field of 11111 is not a mode command. This is not controlled by this bit, but rather with the sa31 bit, which is also located in the 1553_control_ reg. The legal/illegal command word information in the rt_control_buffer must be modified accordingly.
- There are no mode codes with an associated data word. The MSB of the mode code field is ignored, so that a mode code of 10000 is the same as 00000, dynamic bus control. The legal/illegal command word information in the rt_control_buffer must be modified accordingly.
- The MIL-STD-1553A Specification defines only the Dynamic Bus Control mode code, however, all MIL-STD-553B mode codes will be supported if they are legal. Generally, these mode code values will be considered illegal by default, as programmed in the legal/illegal command word information in the rt_control_buffer.
- Only the terminal address, message error and terminal flag bits are defined in the 1553 status word, but this isn't affected by the 1553a bit.

time_tag_enable_external_sync

This bit set allows the active external TTL input to clear the tag timer. This affects the BM and all RT's since they use the same hardware timer.

bc_enable_external_sync

This bit may be read by the application to determine the state of bit 7 of word offset address 0x06 (byte offset 0x0c).

A separate location has been created for the write because a read-modify-write application instruction to change other bits in this register could cause the bit to be written over if the hardware were to update the bit before the instruction has completed.

internal_wrap (dis_ext_bus)

A complete bus simulation is possible without affecting the MIL-STD-1553 bus itself. Setting the internal_wrap bus 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. The bc_run and rt_run bits should be turned off before turning on or off the dis_ext_bus bit to guarantee correct operation.

cpu_interrupt

The PCI bus uses shared interrupt lines. Therefore, PCI, cPCI, PMC, and IP Modules on PCI carriers need to provide a means other than the interrupt vector for determining that it was the requesting source. This is provided by the CPU interrupt and the CPU interrupt enable bits.

The CPU interrupt bit is read from the bc_control register and is a J-K flip-flop, which can be set by microcode or by the host writing a "1" to bit 9 of the "write interrupt bit" register. It can be cleared by the host writing a "0" to bit 9 of the "write interrupt bit" register or by the host performing an interrupt acknowledge cycle. In addition, this bit is cleared on when a board is reset and when the WCS is initialized. For IP-1553 model boards, this bit is output from the board as the active low intreq0 signal. For PCI-1553, cPCI-1553 and PMC-1553 model boards; this bit is output from the board as the INTA signal. For ISA-1553 / PCCARD-1553 model boards, this bit is passed on as the enabled IRQ.

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 then 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.

Note: The software clears the interrupt normally in about 200-300 microseconds best case.

If another message, which has interrupts set, comes in the software should get that interrupt, and a "time-out" interrupt will occur 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.

rt31_broadcast, sa31_mode_code

The rt31_broadcast and sa31_mode_code bits should be set for MIL-STD-1553B, unless overridden by the user application. If rt31_broadcast is not set, then the 5-bit RT address field of a command word will be interpreted as RT 31 rather than as a broadcast command. If it is set, then the disa and disb bits of RT 31 in the rt_enables word of the RT address buffer must both be set. If sa31_mode_code is not set, then 0 and 31 are valid sub-addresses and mode codes are not processed.

In order to comply with MIL-STD-1553A, both of these bits should be turned off (logic zero).

test_bus_enable

Applicable to the VME/VXI-1553 products only. When set this bit disconnects the external 1553 bus and routes the 1553 signals to an internally terminated direct coupled 1553 test bus. This feature allows confidence testing through the isolation transformers and transceivers without transmitting off the board.

rt_enable_external_sync

The rt_enable_external_sync 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.

cpu_interrupt_enable

This register bit is logically AND'ed with the interrupt bit to produce the interrupt request signal (PCI INTA or IP interrupt 0). The interrupt enable bit can't be programmed or altered before the wcs_setup_complete bit is set. It could be possible to get an interrupt that you thought was disabled if you reinitialize the board (setting and later clearing the wcs_setup_complete bit) without performing a board reset. A board reset should always be performed before initializing the WCS, so this shouldn't be a problem. To be sure, clear the wcs_setup_complete bit, program the interrupt enable bit, then set the run bits, in that order.

1553_bus_coupling

Writing a logic one to this bit makes the bus direct coupled (no coupling transformer required). A logic zero makes the bus transformer coupled. It affects both channels A and B of the dual redundant bus. The board powers up in transformer coupled mode.

Note: Writing to this bit have no effect on IP-1553, PMC-1553, and cPCI-1553 boards, as their coupling is selected using board mounted jumpers.

time tag counter control reg (bytes 0x02-0x03) - write only

This register contains control bits for the time tag counter (TTC). The Time Tag Counter Control Register controls loading of the value in the Time Tag Counter Load Register (TTCL) into the Time Tag Counter. Three modes of operation are supported: a software mode and two hardware modes. The hardware modes allow either loading of the TTC through software intervention or auto-incrementing the TTC without software intervention. The Time Tag Counter Control Register cannot be read back by the host. In addition, some model boards support IRIG loaded time tag. See Chapter 11, "IRIG", for more details.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												auto	irig	swl	hwl

Field	Definition
hwl	hardware load time tag counter
swl	software load time tag counter
irig	load time tag counter with IRIG decoder
auto	automatically increment time tag counter

hardware load time tag counter

Setting this bit forces the time tag counter (TTC) to be loaded with the contents of the time tag counter load register (TTCL) with each low to high pulse on the TTL input pin. This allows for external synchronization of the time tag. The TTCL register must be loaded prior to setting this bit. In this mode, when the TTC is loaded an interrupt to the host will be generated. This allows the host software to load the value in the TTCL for the next external sync pulse. It is recommended to use the auto-increment mode rather than hardware load tag counter mode.

software load time tag counter

Setting this bit forces the time tag counter (TTC) to be loaded with the contents of the time tag counter load register (TTCL). The TTCL register must be loaded prior to setting this bit. This bit affects only the corresponding MIL-STD-1553 bus. This mode may be used with an appropriate software routine to increment the TTC upon each interrupt from an external synchronizing source.

irig

Setting this bit forces the time tag counter (TTC) to be loaded with the contents of the IRIG decoder. Otherwise, it is loaded with the contents of the time tag counter load register (TTCL) with each low to high pulse on the TTL input pin.

Auto-Increment time tag counter

Setting this bit forces the time tag counter (TTC) to be incremented by the contents of the time tag counter load register (TTCL) with each low to high pulse on the TTL input pin, without software intervention. This allows for external synchronization of the time tag. The TTCL register must be loaded prior to setting this bit. If interrupts are desired when the load occurs, then also set the hardware load tag counter bit to generate the interrupts.

clear_tag (bytes 0x04-0x05) - write only

Note: This register is only used for IP-1553 board models.

Writing a "1" to bit 0 clears the tag timer. Because any write to this register clears the tag, do not perform any other writes to this register.

Description of the Tag Timer

The 32-bit tag timer is used for time tagging 1553 messages. The timer consists of a 32-bit counter, which is clocked at 1 MHz, which means the LSB represents one microsecond, and the counter overflows at 2 to the 32nd microseconds. The counter outputs are loaded into a register at the mid-sync time of the first command word of each message. The register is then loaded into the rt_message_buffer and bm_message_buffer at the completion of each message. The host may not directly access the tag timer. However, writing to this register resets the timer. The tag timer is also cleared by a hardware reset or software reset. It may also be cleared by a pulse on the external input if enabled by the enable_ext_sync bit of the 1553_control_reg. The external input must go active high for at least one microsecond and may stay high indefinitely. The timer starts counting only after the external sync signal is de-asserted. Finally, the counter may be cleared by reception of a synchronize mode command provided it is enabled in the orphaned_hardware_bits register.

Caution: All RT's use the same physical timer as the Bus Monitor. Therefore, the tag timer for the Bus Monitor as well as all RT's clear simultaneously.

Playback Control/status Register (bytes 0x06-0x07)

This register provides control and status for the Playback feature, which is available on most of Condor Engineering's MIL-STD-1553 boards (It is not applicable to IP-1553 model boards). Playback allows prerecorded missions to be played back onto the 1553 bus for additional analysis.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										halt	buf-emp	err	run	int-en	start

Field	Definition
start	software playback start
inten	software playback interrupt enable
run	hardware playback run
err	hardware playback error

bufemp	hardware playback buffer empty
halt	software mid-playback stop

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 register file 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 any write to the Clear Playback Status hardware register by the host.

hardware playback buffer empty

This bit is set by the microcode when the pb_tail_pointer register file value is equal to pb_head_pointer register file value. This bit is cleared by any write to the Clear Playback Status hardware register by the host.

software mid-playback stop

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

response_reg (byte offset 0x08-0x09), reserved (byte offset 0x0a-0x0b) - write only

The response register contains two, six-bit values for the RT response time-out and late response, and is programmable by the host. The response time-out may be programmed for 4 (001000) to 31.5 (111111) microseconds, in 500 nanosecond increments. The default is 14 microseconds for 1553B. The late response may be programmed for 4 (001000) to 31 (111110) microseconds, in 500 nanosecond increments. The default is 12 microseconds for 1553B. These registers are not readable and should not be changed when the 1553 bus is active. Both the response time out and late response initialize to 31 microseconds upon board reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		Response time out						Reserved		Late response					
Reserved															

Each 6-bit increment equates to a time value in the range of 4 to 32 microseconds, with 500-nanosecond resolution.

The second register is not used but is reserved for a 16-bit response time-out value. The first register (0x08-0x09) would then contain a 16-bit late response value.

bc_enable_external_sync (byte offset 0x0c-0x0d) - write only

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

Setting bit 7 synchronizes the BC major frame to an external event. If the bit is set, an external sync pulse sets the bc_run bit. The bit may be read at bit 7 of address 0x00, the 1553_control_register.

Caution: Bus Controller initialization must occur prior to setting this bit.

Clear Playback Status (byte offset 0x0e - 0x0f)

This register is not applicable to IP-1553 model boards. It is used to clear status and error bits in the playback control/status register. Writing to this location clears the error and buffer empty bits (bit 3 and 4) bits in the playback control/status register.

variable_volt (byte offset 0x10 - 0x11)

This register is not applicable to PMC-1553 and IP-1553 model boards. It determines the output voltage swing of the MIL-STD-1553B interface signals. It should be programmed prior to setting bc_run, bm_run, or rt_run bits in the 1553_control_register. *It may only be written to as a 16-bit word.* Byte writes yield invalid register contents. The register is write-only. For QPMC-1553 only the data field is valid. The other bits are don't cares.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ref	res	ldac	pdb	pda	ab	cr[1..0]		data							

Field	Definition
ref	Selects between internal (0) and external (1) reference
res	Reserved
ldac	Load DAC bit for synchronous update of DAC outputs
pdb	Power-down DAC B
pda	Power-down DAC A
ab	address bit to either DAC A (0) or DAC B (1)
cr[1..0]	Control bits for various data loading functions
data	Data; bit 7 is the MSB

The physical device contains two DACs, A and B. Only DAC A is wired up to the circuit board. The ext bit should be logic "0" since an internal voltage reference is used.

Control Bits

lda	ab	cr1	cr0	Function Implemented
0	X	0	0	Both DAC registers loaded from shift register
0	0	0	1	Update DAC A input register from shift register
0	1	0	1	Update DAC B input register from shift register
0	0	1	0	Update DAC A DAC register from input register

lda	ab	cr1	cr0	Function Implemented
0	1	1	0	Update DAC B DAC register from input register
0	0	1	1	Update DAC A DAC register from shift register
0	1	1	1	Update DAC A DAC register from shift register
1	0	X	X	Load DAC A input register from shift register and update both DAC A and DAC B DAC registers
1	1	X	X	Load DAC B input register from shift register and update both DAC A and DAC B DAC registers

The output voltage swing on the transmitters is defined by the following equation:

Vout = 4 x (Vref x Data/256)

Vref = 2.5 Volts; Data is 8-bit user programmable from 0 to 255 (0 to ff hex).

The default value for the register is "13ff hex" which provides the maximum voltage swing of 10 volts at the output of the transmitter.

LPU Revision Register (byte offset 0x12 - 0x13)

The LPU Revision Register is used for all 1553 model boards, except IP-1553 boards. It is split into the major and minor LPU revision numbers. The upper byte is the major revision number. The lower byte is the minor revision number. For most models, the LPU is a software-programmable logic device and is contained in the API.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Major revision number								Minor revision number							

interrupt_vector (byte offset 0x14)

The 8-bit interrupt vector may be written to at the low byte of word address 0x0a. It is read during an interrupt acknowledge cycle on IP-1553 model boards. PCI- based IP carriers do not normally use the interrupt vector but do use the interrupt bit and interrupt enable bit in the 1553 control register instead. The interrupt vector is cleared with a board reset.

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

write_interrupt_bit (byte offset 0x16-0x17)

Writing a "1" to bit 9 sets the interrupt, intreq0n; writing a "0" to bit 9 clears the interrupt. This is how the software clears the interrupt in systems not using interrupt vectors. This bit is readable at bit 9 of the 1553_control_register.

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

Time Tag Counter (bytes 0x18-0x19)

The 45-bit Time Tag Counter (TTC) may be directly read by the host. It uses a free running one-microsecond clock, so the value read back will not generally be the same value as was loaded from the TTCL and TTCI registers. The API must write (any data value) to the register to reset it to read word 0. Subsequent reads to the register return Word 0, followed by Word 1, followed by Word 2. Since the timer is free-running and may change between the read of each word, the API should contain an algorithm to detect when Words 1 and 2 are stable before reading Word 0. Words 1 and 2 are stable for a period of 65.536 milliseconds.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0 (TTC [15..0])															
Word 1 (TTC [31..16])															
0	0	0	Word 2 (TTC [44..32])												

minor_frame_timer (byte offset 0x1c-0x1d)

This write-only 16-bit hardware register determines the time of each minor frame. The least significant bit represents 25 microseconds. Values from 40 (28 hex) through 65,535 (FFFF hex) may be programmed into the register. This provides for a minor frame time of from 1 millisecond to approximately 1.6 seconds. A board reset loads 40,000 (1.0 second) into the register. All minor frames expressed as a frequency not divisible by 25 microseconds have an approximated frequency with 25-microsecond resolution. The minor frame time should be programmed prior to setting the bc_run bit. The first minor frame begins immediately after bc_run is set and has an accuracy of +/-10 microseconds with a stability of 0.01%.

Note: The first minor frame to be executed is 25 microseconds less than what is programmed in this register. From then on all timing is accurate.

Bus Controller Timing

If the hardware detects insufficient time to process a 1553 message in a message list for the current minor frame, then the message (and all subsequent messages in the minor frame) is not processed, and the minor frame overflow bit in the orphans register is set. If the message is in a high priority aperiodic message list, the minor frame overflow bit is not set and the message is transmitted at the start of the next minor frame. For low priority aperiodic messages, the minor frame overflow bit is not be set, and the message is transmitted when the hardware detects that there are no more regular messages to transmit and there is sufficient time for the aperiodic message.

Warning: The minimum frame time tested is one millisecond (1k Hz). Proper operation is not guaranteed with a shorter frame time.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16 bit value * 25 microseconds representing minor frame time															

Time Tag Counter Load Register (bytes 0x20-0x25)

The time tag counter load register (TTCL) is a 45-bit buffer that may be written by software to allow software programming of the 45-bit time tag counter. The software first loads this register with the desired value, then, through the use of the Time Tag Control register, commands the hardware to load the value of the buffer into the time tag counter. The hardware insures that this load is an atomic operation to insure the correct initialization of the Time Tag Counter.

The hardware supports three modes of loading the Time Tag Counter Load Register into the Time Tag Counter: a software mode and a hardware mode and an automatic mode. These modes are described in the time tag counter control register (TTCC) description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCL [15..0]															
TTCL [31..16]															
undefined							TTCL [44..32]								

Time Tag Counter Increment Register (bytes 0x26-0x29)

The time tag counter increment register (TTCI) is a 32-bit register that is used to increment the 45-bit time tag counter in auto-increment mode. The

software first loads this register with the desired value, then, through the use of the Time Tag Control register (TTCC), commands the hardware to increment the time tag counter by this value whenever an external input pulse occurs.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCI [15..0]															
TTCI [31..16]															

Reserved Registers

Most of the hardware register address space does not contain physical registers. To ensure future upgrade compatibility, do not write to these locations.

CHAPTER 7

Register File Usage

The Register File consists of 128, 26-bit words (112 words for IP-1553 model boards) that are used extensively by the microcode when running Bus Controller, Bus Monitor, or Remote Terminal operations. They are mapped into word offsets 0x00 through 0x7F (byte offsets 0x00 through 0xFF). For IP-1553 model boards only word offsets 0x10 through 0x7F (byte offsets 0x20 through 0xFF) are available. To maintain software or firmware compatibility between products, the file registers at word offsets 0x00 through 0x0F are not used on any of the products. The base addresses for the different model boards are as follows:

Condor Base Model	Page	Base Byte Address
PCI-1553 (Bus 1), cPCI-1553 (Bus 1), or PMC-1553 (Bus 1)	N/A	0500000
PCI-1553, (Bus 2), cPCI-1553 (Bus 2)	N/A	1500000
VME-1552	N/A	1000
ISA-1553/PCCARD-1553 (Bus 1)	0	1400
ISA-1553 (Bus 2)	0	3400
IP-1553	0	0020

For paged model boards, the register file is mapped into page 0. All discrete bits are active "1" unless otherwise specified.

Exercise care when writing to internal registers when the BC, RT, or BM, are running. The microcode being executed on the board doesn't expect any register bit to change unless it performs the change. Writing to the board at the wrong time could cause the BC, RT, or BM to fail, thus requiring it to be setup again.

For PCI-1553, cPCI-1553, and PMC-1553 model boards, the 128-word file registers for bus 1 are located at byte addresses 500000 through 5000ff. Accessing addresses 500100 through 5fffff reads and writes to the same 128-word file registers since a[19..8] are not decoded in the hardware. Software should not access addresses 500100 through 5fffff, as they are reserved for additional file registers that do not currently exist in hardware. Similarly, the 128-word file registers for bus 2 are located at byte addresses 1500000 through 15000ff.

For VME-1553 model boards, the 128-word file registers for channel 1 are located at byte addresses 1000 through 10ff. Accessing addresses 1100 through 1fff reads and writes to the same 128-word file registers since a[19..8] is not decoded in the hardware. Software should not access addresses 1100 through 1fff, as they are reserved for additional file registers that do not currently exist in hardware. The 128-word file registers for channels 2, 3, and 4 are located at byte addresses 201000 through 2010ff, 401000 through 4010ff, and 601000 through 6010ff.

For ISA-1553 and PCCARD-1553 model boards, the 128-word file registers are located at byte addresses 1400 through 14ff. Accessing addresses 1500 through 17ff reads and writes to the same 128 file registers since a[9..8] is not decoded in the hardware. Software should not access addresses 1500 through 17ff, as they are reserved for additional file registers that do not currently exist in hardware. Similarly, for ISA-1553 boards the 128-word file registers for bus 2 are located at byte addresses 3400 through 34ff.

Register File Summary (0x10 - 0xFF)

The following table summarizes the complete list of file registers. Sample initial values are provided as a reference to initialize the file registers if you bypass Condor Engineering's API. Actual usage is defined in the following paragraphs, but you can reference this table as a starting point when building your initialization call. Words 00 through 0F aren't used.

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
0 - D	0x00-0x1A	0	N/A	Reserved; not available on IP-1553
E - F	0x1C-0x1F	0	N/A	Temporary pointers used by microcode only.
10	0x20-0x21	0	pbstrptr	Playback start pointer
11	0x22-0x23	0	pbendptr	Playback end pointer
12	0x24-0x25	0	pbtalptr	Playback tail pointer

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
13	0x26-0x27	0	pbhedptr	Playback head pointer
14	0x28-0x29	0	pbinttsh	Playback threshold word count
15	0x2A-0x2B	0	pbintsts	Playback interrupt status
16	0x2C-0x2D	0	pbthrcnt	Playback interrupt threshold count
17	0x2E-0x2F	0	pbcwdptr	Playback current word pointer
18	0x30-0x31	FEFF	clrbit8	Constant feff hex
19	0x32-0x33	FDFF	clrbit9	Constant fdff hex
1A	0x34-0x35	FBFF	clrbit10	Constant fbff hex
1B	0x36-0x37	F7FF	clrbit11	Constant f7ff hex
1C	0x38-0x39	EFFF	clrbit12	Constant efff hex
1D	0x3A-0x3B	DFFF	clrbit13	Constant dfff hex
1E	0x3C-0x3D	BFFF	clrbit14	Constant bfff hex
1F	0x3E-0x3F	7FFF	clrbit15	Constant 7fff hex
20	0x40-0x41	0	cmd1save	Command Word for RT address detect (BC,BM)
21	0x42-0x43	0	cmd2save	Command word for receive command of RT-RT
22	0x44-0x45	0	rtmsgpt1	RT message pointer for transmit command of RT-RT
23	0x46-0x47	0	eomflag	End-of-message flag
24	0x48-0x49	0	sta1save	
25	0x4A-0x4B	0		Reserved
26	0x4C-0x4D	0	bcstcmds	Broadcast command save
27	0x4E-0x4F	0	tempreg	Temporary microcode storage register

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
28	0x50-0x51	0	rtlastsw	Pointer to base address of Last Status Word Buffer
29	0x52-0x53	0	rtswptr	Temporary pointer to RT message buffer to store status word
2A	0x54-0x55	0	hpmsgpts	High priority aperiodic message pointer
2B	0x56-0x57	0	lpmsgpts	Low priority aperiodic message pointer
2C	0x58-0x59	0	hpmsgptt	High priority aperiodic message pointer Temp
2D	0x5A-0x5B	0	lpmsgptt	Low priority aperiodic message pointer Temp
2E	0x5C-0x5D	0	hpaflag	High priority aperiodic list flag (ffff or 0)
2F	0x5E-0x5F	0	lpaflag	Low priority aperiodic list flag (ffff or 0)
30	0x60-0x61	0	hpabump	High priority list bump normal flag
31	0x62-0x63	0	bcdatptt	BC data pointer for transmit commands
32	0x64-0x65	0	erptr	
33	0x66-0x67	0	eventreq	
34	0x68-0x69	0	disa0	RT 15..0 bus A disables
35	0x6a-0x6b	0	disa1	RT 31..16 bus A disables
36	0x6c-0x6d	0	disb0	RT 15..0 bus B disables
37	0x6e-0x6f	0	disb1	RT 31..16 bus B disables
38	0x70-0x71	0	clrword	constant 0000 hex
39	0x720x73	FFFF	setword	Constant ffff hex
3A	0x74-0x75	0	retryptr	Retry buffer pointer used by microcode only
3B	0x76-0x77	0	trig	Internal Bus Monitor triggering.
3C	0x78-0x79		rtmonitr	Flag used for RT Monitor.

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
3D	0x7A-0x7B	001F	Mask001f	Constant 001f hex
3E	0x7C-0x7D	003F	Mask003f	Constant 003f hex
3F	0x7E-0x7F	0	rtmonitt	Flag used for RT Monitor.
40	0x80-0x81	0	bcmsgptr	BC message block current word pointer
41	0x82-0x83	0	bcmsgpts	BC message block initial word pointer
42	0x84-0x85	0	bcdatptr	BC data pointer
43	0x86-0x87	0	bcinten1	BC interrupt enables
44	0x88-0x89	0	bcinten2	BC interrupt enables
45	0x8A-0x8B	0	bcretryi	BC retry information
46	0x8C-0x8D	1000	mfreg	Minor frame register
47	0x8E-0x8F	0	mfcntr	Minor frame counter
48	0x90-0x91	0	rtctlptr	RT control buffer pointer
49	0x92-0x93	0	rtmsgptr	RT message current word pointer
4A	0x94-0x95	0	rtmsgpts	RT message initial word pointer
4B	0x96-0x97	0	rtmsgpt2	RT message current word pointer for transmitting RT of an RT-to RT message
4C	0x98-0x99	0	bmmsgpts	BM message initial word pointer
4D	0x9A-0x9B	0	bmmsgptr	BM message current word pointer
4E	0x9C-0x9D	0	owordcnt	Output word count used by bc and by playback
4F	0x9E-0x9F	0	owcnt	Output word count used by rt
50	0xA0-0xA1	0	rteipts1	RT error injection pointer
51	0xA2-0xA3	0	rteiptst	RT error injection pointer for transmit command of an RT-to RT message

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
52	0xA4-0xA5	0		Reserved
53	0xA6-0xA7	0	iqueptr	Interrupt queue pointer
54	0xA8-0xA9	0	rtbbase	RT address buffer base address
55	0xAA-0XAB	0	rtfbase	RT filter buffer base address
56	0xAC-0XAD	0	bmfbase	BM filter buffer base address
57	0XAE-0xAF	0	eventptr	BM trigger event pointer
58	0xB0-0xB1	0	retrypts	Retry buffer pointer setup by the API
59	0xB2-0xB3	0	msgsta1	Message status bits
5A	0xB4-0xB5	0	msgsta2	Message status bits
5B	0xB6-0xB7	00A0	orphans	Orphan hardware flags
5C	0xB8-0xB9	0	umctbl0	Undefined mode code constant table
5D	0xBA-0xBB	FFF2	umctbl1	Undefined mode code constant table
5E	0xBC-0xBD	FFFF	umctbl2	Undefined mode code constant table
5F	0xBE-0xBF	FFCD	umctbl3	Undefined mode code constant table
60	0xc0-0xc1	1	setbit0	Constant 0001 hex
61	0Xc2-0xc3	2	setbit1	Constant 0002 hex
62	0xc4-0xc5	4	setbit2	Constant 0004 hex
63	0xc6-0xc7	8	setbit3	Constant 0008 hex
64	0xc8-0xc9	10	setbit4	Constant 0010 hex
65	0xca-0xcb	20	setbit5	Constant 0020 hex
66	0xcc-0xcd	40	setbit6	Constant 0040 hex
67	0xce-0xcf	80	setbit7	Constant 0080 hex

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
68	0xd0-0xd1	100	setbit8	Constant 0100 hex
69	0Xd2-0xd3	200	setbit9	Constant 0200 hex
6A	0xd4-0xd5	400	setbit10	Constant 0400 hex
6B	0xd6-0xd7	800	setbit11	Constant 0800 hex
6C	0xd8-0xd9	1000	setbit12	Constant 1000 hex
6D	0xda-0xdb	2000	setbit13	Constant 2000 hex
6E	0xdc-0xdd	4000	setbit14	Constant 4000 hex
6F	0xde-0xdf	8000	setbit15	Constant 8000 hex
70	0xe0-0xe1	FFFE	clrbit0	Constant fffe hex
71	0xe2-0xe3	FFFD	clrbit1	Constant fffd hex
72	0xe4-0xe5	FFFB	clrbit2	Constant fffb hex
73	0xe6-0xe7	FFF7	clrbit3	Constant fff7 hex
74	0xe8-0xe9	FFEF	clrbit4	Constant ffef hex
75	0xea-0xeb	FFDF	clrbit5	Constant ffdf hex
76	0xec-0xed	FFBF	clrbit6	Constant ffbf hex
77	0xee-0xef	FF7F	clrbit7	Constant ff7f hex
78	0xf0-0xf1	400	bc-flags	BC Flag register
79	0xf2-0xf3	0	endflags	Flag register
7A	0xf4-0xf5	0	bccntrls	BC control word storage
7B	0xf6-0xf7	0	bceipts/pbeipts	BC/Playback error injection pointer
7C	0xf8-0xf9	0	wordseq	Word sequencer bits
7D	0xfa-0xfb	0	dword	1553 word input from decoder
7E	0xfc-0xfd	0	tempsta1	Temporary microcode storage register

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
7F	0xfe-0xff	0	temp	Temporary microcode storage register

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_reg for how to determine if there is bus activity.

high priority (0x54 - 0x55) and low priority (0x56 - 0x57) aperiodic message pointers

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, these pointers are 19-bit values with the three least significant bits truncated.

See the section, “Aperiodic Message Lists” in Chapter 8, “RAM Usage” for details.

high priority (0x58 - 0x59) and low priority (0x5A - 0x5B) aperiodic temporary message pointers

These two pointers are used by the firmware to store the regular message pointer (bcmsgpts) while the aperiodic list(s) are being processed.

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, these pointers are 19-bit values with the three least significant bits truncated. See Chapter 8, “RAM Usage” for details.

high priority (0x5C - 0x5D) and low priority (0x5E - 0x5F) aperiodic message list flags

The high and low priority aperiodic message list flags are used by the microcode to begin the processing of an aperiodic message list. These two registers *must* be cleared by software prior to running the Bus Controller.

high priority list bump normal flag (0x60 - 0x61)

The firmware uses this flag to detect a that a high priority list was processed prior to the first message in a frame and thus the normal frame still needs to be transmitted onto the 1553 bus.

bc_msg_pointer (0x80 - 0x81), bc_msg_pointer_save (0x82 - 0x83) and bc_msg_pointer_temp (0xA4 - 0xA5)

The bc_msg_pointer_save (BCMSGPTS) is used by the hardware to point to the next BC message block to be processed. It is initialized by the host to point to the first word of the first message in a major frame. It is updated by the bc_next_msg_pointer just after the command word is output onto the 1553 bus.

Note: This register must be programmed to point to a valid BC Message Block prior to turning on the bc_run bit or the board may not operate correctly.

The bc_msg_pointer (BCMSGPTR) is used by the hardware to point to each word in the BC message block as it is being processed. The microcode loads the bc_msg_pointer_save register into this register just before transmitting the command word onto the 1553 bus. It subsequently updates this pointer when processing a message. The host *must not* write to this register during message processing.

The bc_msg_pointer_temp (BCMSGPTT) is used by the microcode to temporarily store the bc_msg_pointer for use in the end_of_message routine.

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, these pointers are 19-bit values with the three least significant bits truncated. See Chapter 8, “RAM Usage” for details.

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

bc_data_pointer (0x84 - 0x85), bc_data_data_pointer_temp (0x62 – 0x63)

The 16-bit incremented bc_data_pointer keeps track of the data word to be processed in multiple word messages. The hardware loads the data_pointer_a or data_pointer_b from the bc_message_block into this register and then increments through the data.

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, this pointer is a 19-bit value with the three least significant bits truncated. See Chapter 8, “RAM Usage” for details.

The bc_data_pointer_temp register is loaded with the contents of bc_data_pointer during the processing of a message in order to free up bc_data_pointer for the next message. This allows you to overlap the processing of consecutive messages to meet minimum intermessage gap time requirements.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Points to the current data word of the current BC message.															

bc_interrupt_enables (0x86 - 0x89)

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 these two registers to enable the interrupt conditions of the BC. The bits are identical to the similarly named bits in the bc_message_status, rt_interrupt_enables, rt_ message_status, bm_interrupt_enables, and bm_message_status words. (See the section, “rt_interrupt_enables” in Chapter 8, “RAM Usage” for a complete description of these bits.)

bc_interrupt_enable1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nig	bce	wb	res	ch	ira	lr	er	ncd	pe	2bus	mbe	is	lw	iw	hw

Field	Definition
hw	high word
iw	invalid word
lw	low word
is	inverted sync error
mbe	mid-bit error
2bus	two-bus error
pe	parity error
ncd	non-contiguous data
er	early response
lr	late response
ira	incorrect rt address
ch	channel

res	reserved
wb	response on wrong bus
bce	bit count error
nig	no/short intermessage gap

bc_interrupt_enable2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
res	bmo	te	tb	me	nres	rtry	rtb1	rtb0	res	mc	st	rrt	rtf	bcm	em

Field	Definition
em	end of message
bcm	broadcast message
rtf	RT-to-RT message format
rrt	reset rt
st	self test
mc	mode code
res	reserved
rtb0	rt busy bit 0
rtb1	rt busy bit 1
rtry	retry occurred
nres	no response
me	message error bit
tb	API use: trigger begin
te	API use: trigger end
bmo	API use: bus monitor overflow

bc_retry_info (0x8A - 0x8B)

The host programs this register during setup. The information is used by the hardware to retry a message if the retry_enable bit is set in the bc_message_block. Only one retry is attempted per message. All of the bits below represent the conditions, which cause a retry.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					me	instr	srq	Reserved				busy	ssf	Rsrv	tf

Field	Definition
tf	terminal flag bit set
ssf	subsystem flag set
busy	busy bit set
srq	service request bit set

instr	instrumentation bit set
me	message error bit set

out_word_count (0x9C - 0x9D)

The out_word_count register is used by the BC to keep track of the number of data words remaining to be output for the current message.

Playback also uses this register (see Playback description).

bc_flags (0xF0 - 0xF1)

Bits 0 through 5 are used internally by microcode. They must be cleared by software during BC initialization. The nrsp bit is programmed by the host software and is not altered by the microcode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					nrsp	res	res	res	ro	wem	roc	neom	rts1	bcoi	woic

Field	Definition
woic	wait on invalid command
bcoi	bc output incomplete
rts1	retry on status word 1
neom	do not process end of message for bc
roc	message status retry bit
wem	wait end of message
ro	rt-to-rt output
res	reserved
nrsp	retry on no response

wait_on_invalid_command

This bit is used by the microcode to allow inputting of an invalid command word that has established a valid sync plus two data bits before outputting the next message in the bus list. It is cleared at the start of a major frame and when a valid command is input from the bus. It is tested when an invalid command is detected, and if not set, then it is set and the bc_wait register is cleared, allowing the microcode to execute the next message in the bus list. If set, it prevents clearing of the bc wait register, preventing the bus list from getting out of sync.

bc_output_incomplete

This flag is set by the microcode after it sends the command word to the encoder if there are more words to be output by the BC. The microcode sends the remaining data words (on transmit commands) or the transmit command word (on RT-to-RT commands) until the last word has been sent. The flag is cleared after the last word is sent.

retry_on_status_word1

This bit is for microcode sequencing when there is a retry caused by a failure with the transmit status word of an RT-to-RT command. A second retry isn't injected if the receiving RT's status word also has a failure.

do not process eom for BC

This bit is for microcode sequencing when so that the end of message routine (eom) is not processed until all retries are complete.

message status retry

This bit is for microcode sequencing when a retry message has occurred.

wait end of message

This flag is used to hold off the BC No-op and Conditional Message instructions from being processed before the previous message has processed the "end of message" routine.

rt-rt_output

This bit is set to prevent the microcode from clearing the "wait on invalid command" flag multiple times on an invalid rt-to-rt command.

retry on no response

If retries are enabled, setting this bit mandates a retry if the RT doesn't respond to the BC. This bit must be programmed by the host.

bc_control_save (0xF4 - 0xF5)

Used by microcode to store the control word from the BC message block for use by the end-of-message routine.

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

bc_error_inject_pointer_save (0xF6 - 0xF7)

Used by microcode to store the error injection pointer for the BC as the words of the message are sent to the encoder. This register is also used in Playback mode.

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

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_reg is set) and there is bus activity.

Rt_disables (0x68 - 0x6F)

These four registers must be programmed by the API to determine which RT's are enabled and which are disabled. There are 64 bits: 32 RT disable bits for bus A; 32 RT disable bits for bus B. The first two words represent Bus A; the last two words represent Bus B. The LSB of the first word represents RT 0, Bus A; the MSB of the second word represents RT 31 of Bus A; the LSB of the third word represents RT 0 of Bus B, etc.. A logic "0" in the bit position represents an enable; a logic "1" means disable.

broadcast command save (0x4C - 0x4D)

This register is used to store all broadcast commands as they are input from the 1553 bus. It is used by the Transmit Last Command mode code for all Remote Terminals simulated by the board. The firmware determines whether to send the contents of this register as the data word rather than

the command word stored in the rt_address_buffer for each RT, depending on which command was most recently received.

rt_last_status_word_pointer (0x50 - 0x51)

This pointer is programmed by the driver to point to the base address of the rt_last_status_word_buffer. The 32-word buffer is on an even 32-word boundary. It is used to save the status word for later use by the transmit status word and transmit last status mode commands. This buffer is conceptually part of the rt_address_buffer and is best appended to the end of it.

rt_status_word_pointer (0x52 - 0x53)

This register is a temporary pointer to RT message buffer to store status word. It is used by microcode only.

rt_message_pointer_save (0x94 - 0x95)

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 rt_message_pointer is saved by the microcode at the start of a message to be used at the end of the message to access the interrupt enable and message status words. The microcode updates this register at the end of processing a message. 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.

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, this pointer is a 19-bit value with the three least significant bits truncated. See Chapter 8, “RAM Usage” for details.

rt_message_pointer (0x92 - 0x93)

This register is loaded with a pointer to the rt_message_buffer word being processed.

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, this pointer is a 19-bit value with the three least significant bits truncated. See Chapter 8, “RAM Usage” for details.

rt_message_pointer1 (0x44 - 0x45)

This register is used to process only RT-to-RT messages. This register is loaded with a pointer to the rt_message_buffer word being processed for the receive command of RT-to-RT messages.

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, this pointer is a 19-bit value with the three least significant bits truncated. See Chapter 8, “RAM Usage” for details.

rt_message_pointer2 (0x96 - 0x97)

This register is used to process only RT-to-RT messages. This register is loaded with rt_message_pointer_save for the receive command of an RT-to-RT message. RT-to-RT commands require a second rt_message_pointer since two buffers are processed even though only one contains a data block.

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, this pointer is a 19-bit value with the three least significant bits truncated. See Chapter 8, “RAM Usage” for details.

rt_control_pointer (0x90 - 0x91)

This register is loaded with the contents of the rt_filter and keeps track of the rt_control buffer word being processed.

owcnt (0x9E - 0x9F)

The owcnt register is used by the RT to keep track of the number of data words remaining to be output for the current message.

rt_error_inject_pointer_save1 (0xA0 – 0xA1) and rt_error_inject_pointer_save2 (0xA2 - 0xA3)

These registers are used by the microcode to hold error inject information pointers.

rt_address_buffer_base_address (0xA8 - 0xA9)

This pointer is programmed by the driver to point to the base address of the rt_address_buffer. The rt_address_buffer_base_address points to an even 128-word boundary.

rt_filter_base_address (0xAA - 0xAB)

This pointer is programmed by the driver to point to the base address of the rt_filter_buffer. The rt_filter_base_address points to an even 2K-word boundary (0x800).

rt_enables (0x68 - 0x6F)

The API driver initializes these four registers to enable the RT's. The first two registers enable bus A (0x68 – 0x6B) and the next two registers enable bus B (ox6C – 0x6F). A logic one enables the RT; a logic zero disables the RT.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15 A	RT14 A	RT13 A	RT12 A	RT11 A	RT10 A	RT9 A	RT8 A	RT7 A	RT6 A	RT5 A	RT4 A	RT3 A	RT2 A	RT1 A	RT0 A
RT31 A	RT30 A	RT29 A	RT28 A	RT27 A	RT26 A	RT25 A	RT24 A	RT23 A	RT22 A	RT21 A	RT20 A	RT19 A	RT18 A	RT17 A	RT16 A
RT15 B	RT14 B	RT13 B	RT12 B	RT11 B	RT10 B	RT9 B	RT8 B	RT7 B	RT6 B	RT5 B	RT4 B	RT3 B	RT2 B	RT1 B	RT0 B
RT31 B	RT30 B	RT29 B	RT28 B	RT27 B	RT26 B	RT25 B	RT24 B	RT23 B	RT22 B	RT21 B	RT20 B	RT19 B	RT18 B	RT17 B	RT16 B

rtmonitr (0x78 - 0x79), rtmonitt (0x7E – 0x7F)

These registers are flags used by the firmware. They should be cleared at board initialization, but otherwise are not accessed by software. When an RT is running in RT-Monitor mode, rtmonitr is non-zero; normally it is zero. A second flag is used for the transmitting RT in an RT-to-RT transfer, as the board could be monitoring both RT's in RT-Monitor mode.

Bus Monitor 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 BM is running and there is bus activity.

bm trigger event type (0x48 - 0x49)

This register is used by the hardware to store the trigger event type.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
store trigger event type.															

bm_message_pointer (0x9A - 9B), bm_message_pointer_save (0x98 - 0x99)

The bm_message_pointer_save register is a pointer to the first location of a message in the bm_message_buffer. It is initialized by the host. Buffering may be many messages deep since this pointer is updated by the hardware at the end of every message. The pointer doesn't change if the buffer is only one message deep.

The bm_message_pointer register is a pointer to the current location of a word in the bm_message_buffer. At the start of a message, this register has the same value as the bm_message_pointer_save. As a message is being processed, this pointer may point to any word in the current message. The API uses this fact to determine which message the hardware is currently processing. The bm_run bit may be turned off when the bm_message_pointer_save points to the message after the message that the bm_message_pointer is pointing to, since it is updated at the end of each message.

For all boards except IP-1553, the bm_message_pointer is a 19-bit value with the three least significant bits truncated. See Chapter 7, "RAM Usage" for details.

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

bm_filter_base_address (0xAC - 0xAD)

This pointer is programmed by the driver to point to the base address of the bm_filter. The bm_filter_base_address points to an even 2K-word boundary (0x800).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Points to the base address of the BM filter buffer															

endflags (0xF2 - 0xF3)

These flags are used internally by the microcode but must be cleared by software prior to running the bus.

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

Field	Definition
res	reserved
nof	nth occurrence flag

nof

This flag is set when the nth occurrence of a command has been received. It is used at the end of message routine to determine if the Bus Monitor interrupt should be set for this message. All commands are saved to the Data Buffer regardless of this flag, but the host is only interrupted once every nth time.

bm_trigger_event_pointer (0xAE - 0xAF)

This pointer is programmed by the driver to point to the base address of the bm_trigger_event buffer.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Points to the base address of the BM trigger event buffer															

Playback Registers

playback start pointer (0x20 - 0x21)

The host loads this register with the base address of the Playback Data Buffer. The address given by the register is shifted left by three bits to obtain the actual word address (19 bit) of the beginning of the buffer.

playback end pointer (0x22 - 0x23)

The host loads this register with the end address of the Playback Data Buffer. The address given by the register is shifted left by three to obtain the actual word address (19 bit) of the end of the buffer.

playback tail pointer (0x24 - 0x25)

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 (0x26 - 0x27)

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 (0x28 - 0x29)

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 (0x2A - 0x2B)

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 (0x2C - 0x2D)

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

- Playback interrupt status (pbintsts) 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 (pbinttsh) register.

playback current word pointer (0x2E - 0x2F)

This is a 19-bit address pointer used to point directly at the playback message buffer in RAM. At the beginning of playback, this value is initialized by the ucode as the playback start pointer (pbstrptr) multiplied by 8. The host should not write to this register.

out_word_count (0x9C - 0x9D)

For playback, this register keeps track of the output word count for each message. The word count includes command, status, data, and intermessage timing words.

This register is also used for BC mode. Playback mode is mutually exclusive of BC mode and the two functions may not run simultaneously.

pb_error_inject_pointer_save (0xF6 - 0xF7)

This register is used by the microcode to store the error injection pointer for playback as the words of the message are sent to the encoder. Playback uses the error inject buffer to invert the sync for data words as it sends them as if they were command words in order to control the timing. This register is also used in BC mode.

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

Multi-Purpose Registers

This section defines multi-purpose registers.

command_word1 (0x40 - 0x41) and command_word2 (0x42 - 0x43)

These two registers are used by the microcode to save the command word(s) upon command reception on the 1553 bus.

The command_word1 and command_word2 registers are used to save the RT address for outputting the status word (RT) or for incorrect RT detection for the simulated BC and BM. The second command is used only for RT-to-RT messages. It is for the receiving RT.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1553 command word															

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 bus is running and there is bus activity.

interrupt_queue_pointer (0xA6 - 0xA7)

This register is described in the section “Interrupt Queue” in Chapter 8, “RAM Usage”. The register must be set to a valid interrupt queue location prior to setting the bc_run, bm_run or rt_run bits if interrupts are enabled.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Points to the first word of the current queue member															

message_status1 and message_status2 (0xB2 - 0xB5)

These two registers are used by the firmware to store temporary message status for the rt_message_status, bc_message_status and bm_message_status words. See Chapter 8, “RAM Usage”, rt_interrupt_enables for a complete description of these bits.

orphaned_hardware_bits (0xB6 - 0xB7)

Miscellaneous control and status bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hpfo	bcoierr	bbserr	res	enrerr	mfo	res	res	res	fet	res	rtts	mdb	mda	smcd	res

Field	Definition
res	reserved
smcd	sync mode code with data word broadcast
mda	bm disable channel a
mdb	bm disable channel b
rtts	rt load time tag on sync
fet	fire ext. trigger if sync mode code
mfo	minor frame overflow
enrerr	encoder not ready on mfo
bbserr	bc busy flag set on mfo
bcoierr	bc output incomplete flag set on mfo
hpfo	high priority frame overflow

sync_mode_code_with_data_word_broadcast

This flag is used internally by the microcode in the RT mode to designate the sync w/data word mode code broadcast command was received.

bm_disable_channel_a, and bm_disable_channel_b

The software may disable a channel to be monitored. The bm_run bit must also be set to enable the BM. Normally, both channels are enabled (logic zeros).

rt_load_time_tag_on_sync

This bit is valid only for the synchronize without data word and 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 hardware timer.

fire_ext_trigg_if_sync_mode_code

This bit is set by software to cause the board to fire the external trigger output TTL signal when the RT receives a sync mode command.

minor_frame_overflow

If the Bus Controller hardware determines that there is insufficient time to transmit and process all the messages in a frame, it sets the minor frame overflow bit. This bit must be reset by the software and doesn't give any indication of which messages overflowed and when they overflowed; the Bus Monitor must be examined to determine that.

The system design must ensure that frames do not overflow, taking into account extra words associated with RT to RT messages, retries (CEI 1553 products can perform one retry per message), high word error injection and long programmable responses. The hardware determines if there is sufficient time to process a message by using the following algorithm, which provides a safety margin for which a bus loading timing budget must also allow:

terms defined:

broadcast = (RT address = 31) AND (rt31_bcst bit of 1553_control reg = 1)

mode code = (subaddress = 0) OR (subaddress = 1 AND sa31_mc bit of 1553_control register = 1)

gap time = intermessage gap time that *follows* this message and is programmed with the message.

```
Begin
If  broadcast then
     sufficient-time = 300 + (# of data words * 25) usecs + gap time
else if mode code then
     if no data word then
          sufficient-time = 125 usecs + gap time
     else
          sufficient-time = 150 usecs + gap time
     endif
else
     sufficient-time = 125 + (# of data words * 25) usecs + gap time
endif
End
```

Notes:

1. For every 16 us programmed for the gap time in the message block, 25 us will be added to sufficient-time.
2. The response time is assumed to be within twelve microseconds.
3. High word errors are not included to the allowable time comparison.
4. There is some variation due to firmware execution and therefore, another 50 microseconds should be allowed to *guarantee* that the

hardware won't report there is insufficient time and not send the message.

5. The hardware does not limit the minimum frame time below the limits of this algorithm; however, the API should limit the minimum frame time to a more practical one millisecond.

minor_frame_overflow associated errors

Bits 11 through 14 indicate hardware signals or microcode flags that were active during a minor frame overflow and are intended to be used in a diagnostic or troubleshooting environment only.

high priority overflow

The firmware was attempting to process a high priority message when the minor frame time ended. This is not an error and is for informational purposes only.

undefined_mode_code_table[3..0] (0xB8 - 0xBF)

The driver programs these four constants during setup. The hardware uses them to determine undefined mode commands (illogical commands) for all RT's including broadcast. This table represents Table I-1.3 of the DDC MIL-STD-1553 Designer's Guide. The T/R (MSB) and Mode Code (5 LSB) columns make a six-bit field which point to the 64 bits in this table. The logic value of the bit represents the "YES" (0) or "NO" (1) entry in the INVALID column of the DDC table. The LSB of the first word represents the first entry, the 2nd LSB the second entry, and the MSB of the fourth word represents the 64th entry. The constants are as follows:

undefined_mode_code_table0	0000	Receive mode code, no data
undefined_mode_code_table1	FFF2	Receive mode code, data
undefined_mode_code_table2	FFFF	Transmit mode code, no data
undefined_mode_code_table3	FFCD	Transmit mode code, data

Use the following table (from MIL-STD-1553) to find a definition of the undefined mode commands (illogical commands) for 1553B. Broadcast transmit commands are always treated as invalid.

T/R	Mode Codes	Data Word Associated	Broadcast Allowed	Reserved	Invalid
Receive	0 - 16	UND	UND	UND	YES
Receive	17	YES	YES	NO	NO
Receive	18 - 19	UND	UND	UND	YES
Receive	20 - 21	YES	YES	NO	NO

T/R	Mode Codes	Data Word Associated	Broadcast Allowed	Reserved	Invalid
Receive	22 - 31	YES	TBD	YES	NO
Transmit	0	NO	NO	NO	NO
Transmit	1	NO	YES	NO	NO
Transmit	2	NO	NO	NO	NO
Transmit	3-8	NO	YES	NO	NO
Transmit	9 - 15	NO	TBD	YES	NO
Transmit	16	YES	NO	NO	NO
Transmit	17	UND	TBD	UND	YES
Transmit	18 - 19	YES	NO	NO	NO
Transmit	20 - 21	UND	TBD	UND	YES
Transmit	22 - 31	YES	TBD	YES	NO

TBD = “To be Determined” as indicated in MIL-STD-1553B
UND = “Undefined, not specifically assigned in 1553B”

For MIL-STD-1553A, these four registers are programmed with 0xffff, which means all mode commands are defined and processed if legal.

Temporary registers (0x4E - 0x4F, 0xEC – 0xED and 0xEE - EF)

These registers are used internally by the microcode as general purpose registers. The software should not alter them except for diagnostic purposes.

eom_flag (0x46 - 0x47)

This flag is used internally by the microcode to indicate that the end_of_message function needs to be processed. It must be cleared by software prior to running the bus. The flag is set (non-zero) when the last word of a message is input from the 1553 bus or a no-response time-out or low word error occurs. Instead of processing the end_of_message routine immediately, which informs the host that the message is complete, the microcode sets this flag which allows processing of the next BC message first, allowing for shorter inter-message gap times.

word sequencer bits (0xF8 - 0xF9)

This register is loaded by the firmware for each 1553 word input from the decoder. It contains the message format (receive, transmit, broadcast, etc.) and other word information. It is used by the firmware to determine the action to take upon receipt of each word.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
message format					other word info					Current data word count					

decoder word (0xFA - 0xFB)

This register is loaded by the firmware for each 1553 word input from the decoder. It contains the 1553 word itself (command, status, data).

temporary message status (0xFC - 0xFD)

This register is loaded by the firmware for each 1553 word input from the decoder. It contains the message status bits that are used for the bc_message_status, rt_ message_status and bm_message_status words.

The message status bits for the 1553 status and data words are logically or'ed with the message status bits for the 1553 command word before they are stored in the BC, BM and RT message buffers. For the Bus Monitor, the individual message status words are also stored separately for each 1553 word.

constants

These registers contain constants required by the hardware. The host must load these constants prior to running the board. See the following table for details.

Register Name	Word Address	Byte Address	Hex Data
CLRBIT8	18	30-31	FEFF
CLRBIT9	19	32-33	FDFF
CLRBIT10	1A	34-35	FBFF
CLRBIT11	1B	36-37	F7FF
CLRBIT12	1C	38-39	EFFF
CLRBIT13	1D	3A-3B	DFFF
CLRBIT14	1E	3C-3D	BFFF
CLRBIT15	1F	3E-3F	7FFF
CLRWORD	38	70-71	0
SETWORD	39	72-73	FFFF
MASK001F	3D	7A-7B	001F
MASK003F	3E	7C-7D	003F

Register Name	Word Address	Byte Address	Hex Data
SETBIT0	60	C0-C1	1
SETBIT1	61	C2-C3	2
SETBIT2	62	C4-C5	4
SETBIT3	63	C6-C7	8
SETBIT4	64	C8-C9	10
SETBIT5	65	CA-CB	20
SETBIT6	66	CC-CD	40
SETBIT7	67	CE-CF	80
SETBIT8	68	D0-D1	100
SETBIT9	69	D2-D3	200
SETBIT10	6A	D4-D5	400
SETBIT11	6B	D6-D7	800
SETBIT12	6C	D8-D9	1000
SETBIT13	6D	DA-DB	2000
SETBIT14	6E	DC-DD	4000
SETBIT15	6F	DE-DF	8000
CLRBIT0	70	E0-E1	FFFE
CLRBIT1	71	E2-E3	FFFD
CLRBIT2	72	E4-E5	FFFB
CLRBIT3	73	E6-E7	FFF7
CLRBIT4	74	E8-E9	FFEF
CLRBIT5	75	EA-EB	FFDF
CLRBIT6	76	EC-ED	FFBF
CLRBIT7	77	EE-EF	FF7F

CHAPTER 8

RAM Usage

There is a 512K x 16-bit physical SRAM for each 1553 bus (64K x 16-bit for IP-1553 model boards). The application may use this RAM in any manner desired within the constraints outlined in the detailed description that follows.

For the PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, there is a second 512K-word address space following the RAM address space, which is assigned to future RAM expansion. The RAM expansion space should not be written to as it may write over the programmed RAM. This is true because the hardware may not decode the address line that determines which 512K-word space is being accessed.

For ISA-1553 and PCCARD-1553 boards, the 1M byte RAM is accessed in 2K-byte address space using the page register to select the upper nine address lines.

For two bus MIL-STD-1553B PCI-1553, cPCI-1553, PMC-1553, ISA-1553, and PCCARD-1553 model boards, there is separate RAM address space and RAM expansion address space for each bus.

The following table shows the minimum and maximum RAM usage for BC, RT, and BM modes of operation. The API programs the actual RAM address allocation. For IP-1553 boards, the on-board registers are mapped into the first 256-byte address space, which makes the RAM mapped there inaccessible.

RAM Usage	Number of Bytes *	Number of Bytes **
RT Address Buffer	256	256
RT Last Status Word	64	64
RT Filter Buffer	4K	4K
RT Control Buffer for non-Broadcast	384 to 12K	384 to 12K
RT Control Buffer for Broadcast	4K	4K
RT Message Buffer (data storage)	0 to 118K	0 to 1,000K

RAM Usage	Number of Bytes *	Number of Bytes **
BC Message Block / BC Data Buffer	0 to 124K	0 to 1,000K
BM Filter Buffer	4K	4K
BM Control Buffer	16 to 16K	16 to 16K
BM Message Buffer (data storage)	0 to 102K	0 to 1,000K
BM Trigger Buffer	128	128
Error Injection Buffer	0 to 2K	0 to TBD
Interrupt Queue	0 to 6K	0 to TBD

* PCI-1553, CPCI-1553, PMC-1553, ISA-1553, PCCARD-1553
** IP-1553

19-Bit RAM Pointers

The following description applies to all board models except. IP-1553.

To accommodate 512K words of memory using 16-bit pointers, the driver provides the hardware with only the 16 most significant bits of the pointers. The hardware automatically shifts the pointers up by three bits, filling the three least significant bits with zeros to form 19-bit pointers. This limits these pointers to accessing only eight-word blocks of memory.

The following buffers may be located above the lower 64K and thus require 19-bit pointers: the RT Message Buffer, the BC Message Block, the BC Data Buffer, and the BM Message Buffer. Each message entry in the four buffers must begin on an 8-word boundary. Buffer lengths are now even multiples of eight, where undefined words are appended to the end of the message. Buffer lengths are as follows:

- BC Message Block 16 consecutive words per message
- BC Data Buffer A 40 consecutive words per message
- BC Data Buffer B 40 consecutive words per message
- RT Message Block 48 consecutive words per message
- BM Message Buffer 88 consecutive words per message

These buffers may be located anywhere in memory that isn't allocated to other functions, including the first, 64K address space. Each BC message requires 96 words, unless BC Data Buffer A is the same as BC Data Buffer

B, in which case each BC message requires 56 words. In addition, space should be reserved for BC Message Blocks for NO-OP, BC conditional, and end of frame Control Flow Blocks.

The remaining buffers are limited to the lower 64K-word address space. Therefore, the driver provides pointers to these buffers with only the 16 least significant bits. The hardware automatically appends leading zeros to these pointers. This is the same as the PC-1553 and IP-1553 product lines.

- RT Address Buffer, including Last Status Word (160 words)
- RT Filter Buffer (2K words)
- RT Control Buffer (6K words maximum)
- BM Filter Buffer (2K words)
- BM Control Buffer (8K words maximum)
- BM Trigger Buffer (64 words)
- Error Injection Buffer
- Interrupt Queue

The combined size of the Error Injection Buffer and the Interrupt Queue is limited to only 64K words, less the amount used by the other six buffers. Practically, these two buffers are no more than a few thousand words deep, leaving room for messages in the first 64K addresses.

The following is a complete list of pointers that the host stores in RAM. These pointers need to be shifted right by three bits by the host. The hardware shifts them left by three bits to provide 19-bit pointers to RAM, but stores only the 16 most significant bits back into these registers.

- rt_control_buffer.rt_message_pointer
- rt_message_buffer.rt_next_message_pointer
- bc_message_block.data_pointer_a
- bc_message_block.data_pointer_b
- bc_message_block.bc_next_message_pointer
- bc_message_block.bc_previous_message_pointer
- bc_message_block.control_flow.branch_bc_msg_pointer
- bc_message_block.control_flow.bc_next_message_pointer
- bc_message_block.control_flow.bc_previous_message_pointer
- bm_message_buffer.bm_message_pointer
- interrupt_queue.message_pointer (hardware writes this/host reads it)

In addition, the following file registers' contents need to be shifted right by three bits by the host.

- file_register.bcmsgpts (41)
- file_register.rtmsgpts (4a) (hardware writes this)

- file_register.rtmsgpt2 (4b) (hardware writes this)
- file_register.bmmsgpts(4c) (hardware writes this)
- file_register. hpmsgpts (2a)
- file_register. lpmsgpts (2b)

Other File registers used internally by the microcode as RAM pointers are 19-bits wide; however, the host has access only to the 16 least significant bits.

The bc_message_block.control_flow.address_to_compare pointer requires two words in order to point down to the single word level within the complete 512K word address space:

- bc_message_block.control_flow.address_to_compare_a19_4
- bc_message_block.control_flow.address_to_compare_a3_1

RT Memory Usage

IP-1553 hardware may be set-up to simulate any or all RT's on a 1553 bus. The memory buffers described here and the registers described in a later section 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 four buffers, which may be located anywhere within memory, with a few restrictions. The four buffer types may be placed in any order, but are described in the following order:

- rt_address_buffer
- rt_filter_buffer
- rt_control_buffer
- rt_message_buffer

rt_address_buffer

The rt_address_buffer contains 128 words beginning on a 256-byte boundary. There are four words of information for each of 32 RT addresses. When a command word is received, the first five bits (the RT address) are multiplied by four, and the result is added to the contents of the rt_address_buffer_base_address register to produce a *word* pointer to a location in the buffer. Each RT is described by the following four data words which are listed in sequential order, beginning with RT address zero and ending with RT address thirty-one.

- rt_enables
- rt_status_word
- rt_last_command_word
- rt_bit_word

rt_enables

This word contains a bit to inhibit the terminal flag.

Note: This word formerly contained the RT enable/disable bits and was named "rt_enables". The RT enable bits are now programmed into four file registers.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rtoff	Unused							rsrv	Unused			dbcf	itf	edbc	rtmon

Field	Definition
rtmon	RT Monitor mode
edbc	enable dynamic bus control
itf	inhibit terminal flag
dbcf	dynamic bus control flag
rtoff	turn rt off with dynamic bus control
rsrv	reserved for API use for RT Monitor mode

All unused bits are zero.

rt_monitor_mode

This bit is set by the API to place the RT in monitor mode. In monitor mode, the RT is monitored but not simulated. This allows for quick display updates in real-time with minimal overhead. The RT Map Monitor is useful during simulation if you:

- do not want the overhead of an interrupt. The data is in a fixed location and may be accessed with a single memory call.
- do not want to use Condor's API.
- do not want the overhead of parsing out the messages.

There is a bit for each RT that determines if the board simulates or monitors the RT, but not both. This means that for quick display updates for specific (mapped) values of a particular RT, the same board will not simultaneously be simulating this RT. However, other RT's may be simulated and for multi-mode boards, the Bus Controller may be simulated. The Bus Monitor may be running simultaneously on this same channel. A separate channel must be used to both simulate and monitor the RT simultaneously.

enable_dynamic_bus_control

This data word 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 it's status word. The mode code must also be legalized in the RT Control Buffer for this to occur. It is up to the API to transfer control to the RT.

inhibit_terminal_flag

When this bit is set, the terminal flag bit in the RT's status word is not set on the 1553 status word output onto the bus, regardless of the bit value in the rt_status_word (see next paragraph). This bit is cleared by the host during setup and toggled by the hardware upon reception of mode codes 00110 and 00111. Reception of the "Override Inhibit Mode Command" doesn't affect the status word of the mode command itself, but allows the terminal flag to be set in the status word of subsequent messages.

dynamic_bus_control_flag

This bit is used internally by the microcode to indicate that an enabled legal dynamic mode command has been received. This bit is tested to set the acceptance bit in the status word. Once set, it is the responsibility of the API to clear the acceptance bit in the RT address buffer. Even if the acceptance bit is set in the RT address buffer, the bit is clear in all status words returned by anything other than a dynamic bus control mode code. The API must clear this bit during initialization.

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 hardware may also set some of the status word bits in accordance with MIL-STD-1553B. The software must initialize this word before running the RT by setting the RT address and clearing all other bits.

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

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

RT address bits

This five-bit field represents RT's at address 0 through 31. If the rt_31_broadcast bit in the hardware register 1553_control_reg is set then there will not be an RT 31. This word isn't used in this case.

message error

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” or a “transmit last command” mode code. 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 provide that the enable_dynamic_bc bit of the orphaned_hardware_bits is set.

This function is not presently available.

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. 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. 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.

rt_filter_buffer

The rt_filter_buffer consists of 2,048 words (4K bytes), one word for each of 32 RT addresses, transmit/receive bit, and 32 sub-addresses. When a command word is received, the first eleven bits (RT address, transmit/receive bit, subaddress) are used to replace the lower eleven bits of the rt_filter_base_address register. The resulting address is used to retrieve a one-word entry from the rt-filter_buffer. The word retrieved from the rt_filter_buffer is used as a pointer to the associated rt_control_buffer. The rt_filter_buffer must be located on a 2048-word address boundary. The rt_filter_buffer base address is programmed into a file register by the application.

The following figures show how the rt_filter_buffer is addressed from the information obtained in 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
rt_filter_buffer base offset 0x000 (word offset)

Example 2

0	0	0	0	1	1	0	1	1	1	1

RT Address 1
t/r transmit
Subaddress 15 (0x0F)
rt_filter_buffer base offset 0x06F (word offset)

Example 3

0	1	1	0	1	0	1	0	0	0	0

RT Address 13 (0x0D)
t/r receive
Subaddress 16 (0x10)
rt_filter_buffer base offset 0x350 (word offset)

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. As the message is being processed, the microcode updates the next message pointer, extracted from the message buffer, to provide a circular linked-list of messages. If the pointer in the message buffer points to itself, then transmit data is static, and receive data should be read by the application before it becomes overwritten. Linking many message buffers allows the application to read or update the buffers less frequently so no message is overwritten before it is read. It also allows a canned function to be loaded and read by the BC during operation.

The rt_control_buffer also contains legal and illegal command word information, which determines how each RT responds to the command for each transmit/receive bit, sub-address, word count combination. The RT, transmit/receive bit, sub-address is selected by the rt_filter_buffer, which

points to the appropriate rt_control_buffer location. The rt_control_buffer contains 32-bits, one for each legal word count/mode code, which are packed into two words for each RT, transmit/receive bit, and sub-address combination. This allows the board to detect legal commands down to the word count/mode code level.

For point-to-point (non-broadcast) commands, three words are used for each unique RT, transmit/receive bit, and sub-address combination. If every combination had its own legalization information and/or message buffer pointer, there would be 2,048 times three words, or a 6K-word rt_control_buffer. Most combinations aren't used in any one system. Therefore, a single three-word entry can be used for all disabled RTs and illegal transmit/receive bit and sub-address combinations. An exception is RT 31 when used for broadcast commands, as required in MIL-STD-1553B (see below). A typical rt_control_buffer may use only dozens of words rather than thousands. The order of each three-word entry is as follows:

Buffer Word#	Function
0	legal word count/mode code (15..1,32)
1	legal word count/mode code (31..16)
2	rt message pointer

For broadcast commands, the rt_control_buffer structure is a little different. Each transmit/receive bit, sub-address combination requires 32 bits (2 words) for each RT address (except address 31) and an rt message pointer for all RT's. This totals 63 words per transmit/receive bit, sub-address combination. A reserved word rounds it out to an even 64 words per combination. The buffer structure for each combination is as follows:

Buffer Word#	Enabled RT	Function
0	rt0...30	rt message pointer
1	rt0	legal word count/mode code (15..1,32)
2	rt0	legal word count/mode code (31..16)
3	rt1	legal word count/mode code (15..1,32)
4	rt1	legal word count/mode code (31..16)
....		
61	rt30	legal word count (15..1,32)
62	rt30	legal word count (31..16)
63	N/A	reserved

There are 64 transmit/receive bit, sub-address combinations, which means that the broadcast command portion of the rt_control_buffer can be up to 4K words (64 x 64 words) deep. This buffer might be much smaller since not all combinations are implemented in real world systems. For instance,

the transmit/receive bit is always 0 (receive) for broadcast commands, except for the second command word in a RT-to-RT transfer and for some mode codes. Therefore, only rt_control_buffer entries for transmit commands for sub-addresses 1 through 30 (non-mode codes) are needed if the system is to support RT-to-RT broadcast messages. A single 64-word buffer with all legal word count bits turned off needs to be used for all undefined combinations.

It might seem confusing that the rt_control_buffer is based on unique transmit/receive bit, sub-address combinations, and not on the RT address itself. But when a broadcast command is received, the microcode needs to determine if the broadcast, transmit/receive, sub-address and word count is legal for each RT that it is simulating. The RT address is not that of the broadcast command (31), but that of the RTs being simulated by the board. Therefore, the microcode sequences through each enabled RT to determine if it is legal, and whether or not to set the broadcast message received bit in each of the RT's status words. The microcode requires that the message pointer and all 62 legal word count words be programmed, whether the RT is in the system or not. This is due to the fact that the micro-routine sequences through all of the RTs when it receives a broadcast command. It's the rt_control_buffer that tells it which RTs should take action. The legal word count bits for RTs not simulated by the board must always be logic zero.

The word count field of the 1553 command word determines the mode code for mode commands. In this case, the legal word count words are used to determine legal mode codes.

For non-broadcast commands, this buffer contains three words of information for up to 2K RT address, transmit/receive bit, and subaddress combinations. Only buffers are created for combinations that are enabled. A separate three-word buffer is used for disabled RTs. The total buffer depth may be from 192 words (3 x 64) for one RT enabled, or as large as 6K words (3 x 64 x 32) for all 32 RTs enabled. The three words must be in the following sequential order:

- rt_legal_word_count1 (or rt_legal_mode_code1)
- rt_legal_word_count2 (or rt_legal_mode_code2)
- rt_message_pointer

rt_legal_word_count1 and rt_legal_word_count2

Thirty-two 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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
word_count 15 - word_count 1, word_count 32															
word_count31 - word_count 16															

		hexadecimal equates	
word_count	32	0x00000001	(Long 32 bit equate)
word_count	1	0x00000002	Defines legal word count
word_count	2	0x00000004	
	...	...	
word_count	29	0x20000000	
word_count	30	0x40000000	
word_count	31	0x80000000	

mode code mapping

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							rrt	oitf	itf	oxsd	xsd	stst	tsw	sync	dbc
Reserved										osxs	sxsd	tbw	tlc	swo	tvw

Field	**Definition**
	(rt_enable_mc1 definitions with data words)
dbc	dynamic_bus_control
sync	synchronize
tsw	transmit_status_word
stst	initiate_selftest
xsd	transmitter_shutdown
oxsd	override_xmit_shutdown
itf	inhibit_terminal_flag_bit
oitf	override_inhibit_tf_bit
rrt	reset_remote_terminal
reserved	reserved bits
	(rt_enable_mc2 definitions w/o data words)
tvw	transmit_vector_word
swo	synchronize
tlc	transmit_last_command
tbw	transmit_bit_word
sxsd	selected_xmitter_shutdown

osxs	override_sel_xmt_shutdown
reserved	reserved bits

dynamic_bus_control

The dynamic_bus_control mode command instructs the RT to take bus control from the BC. To accept control, the enable_dynamic_bc bit in the orphans register (0xb6) must be set. The BC mode must be setup on the board prior to accepting bus control. Single mode boards cannot accept control.

selected_xmitter_shutdown and override_sel_xmt_shutdown

Selected_xmitter_shutdown and the override_ sel_xmt_shutdown don't alter the state of the transmitters as the board only simulates a dual-redundant RT. The BC must use transmitter_shutdown and override_transmitter_shutdown mode commands to turn on or off the transmitters for each channel.

rt_message_pointer

This 16-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 hardware every time a new message is processed. The pointer doesn't change if the buffer is only one message deep.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT message pointer															

rt_message_buffer

This 44- or 48-word buffer is used to store 1553 messages and message status. The command, status, and data words are stored in the buffer in the order that they are received on the bus. See the MIL-STD_1553B specification for the allowable Information Transfer Formats. 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. Different data structures are required for BC-RT, RT-BC, and broadcast message formats. The RT status word for receive commands is stored in a variable location, depending on the number of data words.

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:

Word	BC-RT (receive)	RT-BC (transmit)	Broadcast
0	rt_next_message_pointer	rt_next_message_pointer	rt_next_message_pointer
1	rt_error_inject_pointer	rt_error_inject_pointer	rt_error_inject_pointer
2	rt_interrupt_enable1	rt_interrupt_enable1	rt_interrupt_enable1
3	rt_interrupt_enable2	rt_interrupt_enable2	rt_interrupt_enable2
4	rt_message_status1	rt_message_status1	rt_message_status1
5	rt_message_status2	rt_message_status2	rt_message_status2
6	rt_time_tag0	rt_time_tag0	rt_time_tag0
7	rt_time_tag1	rt_time_tag1	rt_time_tag1
8	rt_time_tag2	rt_time_tag2	rt_time_tag2
9	rt_command_word	rt_command_word	rt_command_word
10	reserved	rt_status_word**	reserved
10+1 to 10+n	rt_data_words (0 to 33)	rt_data_words (0 to 33)	rt_data_words (0 to 33)
10+n+1	rt_status_word**	Reserved*	reserved*
10+n+2 to rt_msg_buf_alloc*	reserved	reserved	reserved

Where "n" is the number of data words received for the message. Only one "high word" will be stored per message, hence the maximum number of data words is 33, which is an error condition.

* The number of reserved words in the buffer is determined by the API. Refer to the "BusTools/1553-API User's Manual and Hardware Installation Guide" to determine how many words are allocated for each message.

** For RT-to-RT transfers the status word stored is it's own status word.

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 and transmitting RT enabled	Transmitting RT message buffer

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

rt_next_message_pointer

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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT's next message pointer															

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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT Error Injection Pointer															

rt_interrupt_enables

The host is interrupted only at the end of a message on message errors or on specific 1553 protocol status if the appropriate enable bits are set. The first of these two words contain enable bits to interrupt on specific error status of each message. The second word contains enable bits to interrupt on specific 1553 protocol status. All unused bit positions must be programmed with logic zeros by the software.

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 these two registers to enable the interrupt conditions of the BC. The bits are identical to the similarly named bits in the bc_message_status, rt_interrupt_enables, rt_ message_status, bm_interrupt_enables, and bm_message_status words (See the section, “rt_interrupt_enables” in Chapter 8, “RAM Usage”, for a complete description of these bits) .

rt_interrupt_enable1 (and message_status1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nig	bce	wb	res	ch	ira	lr	er	ncd	pe	2bus	mbe	is	lw	iw	hw

Field	Definition
hw	high word
iw	invalid word
lw	low word
is	inverted sync error
mbe	mid-bit error
2bus	two-bus error
pe	parity error
ncd	non-contiguous data
er	early response
lr	late response
ira	incorrect rt address
ch	channel
res	reserved
wb	response on wrong bus
bce	bit count error
nig	no/short intermessage gap

rt_interrupt_enable2 (and message_status2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
res	bmo	te	tb	me	nres	rtry	rtb1	rtb0	res	mc	st	rrt	rtf	bcm	em

Field	Definition
em	end of message
bcm	broadcast message
rtf	RT-to-RT message format
rrt	reset rt
st	self test
mc	mode code
res	reserved
rtb0	rt busy bit 0
rtb1	rt busy bit 1
rtry	retry occurred
nres	no response
me	message error bit

Field	Definition
tb	API use: trigger begin
te	API use: trigger end
bmo	API use: bus monitor overflow

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 or the low_word bit is set.

invalid_word

The invalid word bit is the logical "or" of the inverted sync, mid-bit error, parity error, non-contiguous data, and bit count error bits. The invalid word bit may also be set with 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, or if there is a low bit count.

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 is not set when there is a bit count error.

non_contiguous_data

This bit is set when there is a gap of ½ us to 2 us of bus dead time, between words.

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 at the Bus Controller. Boards configured as Bus Monitor time-out only at the input to the BM and do not take into account cable delays. These errors have no meaning for an RT, but are present in the rt_message_status as well.

late_response and no_response

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 Bus Controller to the mid-sync time of the status word at the Bus Controller. Boards configured as Bus Monitor time-out only at the input to the BM and do not take into account cable delays. These errors have no meaning for an RT, but are present in the rt_message_status as well. Normally, a response time between 12 and 14 microseconds is a late response. Anything after that is a no_response error, but these times are programmable from 4 to 60 microseconds in the hardware response_reg.

incorrect_rt_address

This function is disabled in the current firmware revision. This bit is set when the address bits returned in the status word do not match the address bits that were transmitted with the command word.

bus

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.

response_on_wrong_bus

This bit is set when the response is returned on a different bus than the command word was transmitted on.

bit_count_error

This bit is set when a sync bit is followed by less than 16 valid data bits. If there are greater than 16 bits, the extra bits are ignored. However, the parity bit is not valid.

no_intermessage_gap/early_command

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. Invalid command words are ignored.

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 and the rt_31_broadcast bit in the 1553_control_register is set.

rt_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. 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.

self_test

This bit is set upon reception of a BIT mode command.

mode_code

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

rtb0 and rtb1

These two bits are “RT busy” bits and are used by the API to determine when it can write to the RT message buffer. When a command word is input on the 1553 bus, the microcode sets rtb0 and rtb1. When the message is complete, the microcode clearx rtb0 and set rtb1. The software can poll this condition and make any updates after the message is complete.

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 and cannot be altered by the hardware or microcode.

rt_message_status

These two words contain discrete bits that are set by the hardware and used to generate interrupts. The host may read these words to determine the interrupt source or to obtain message status. Several of these bits may be set on the same message. The hardware does not attempt to prioritize errors, except that a bit count error turns off the parity error bit. The word/bit structure of these two words are identical to the rt_interrupt_enables words and are described in the preceding paragraph.

The BC and BM get loaded with the same message status and so some of these bits may not have meaning for the RT.

rt_time_tag

These three words (rt_time_tag0 followed by rt_time_tag1) represent the 48 bits of resolution (32 bits for IP-1553 model boards) of the tag timer at the reception of the mid-zero crossing of the parity bit of command word1. The counter has one microsecond resolution, giving a maximum time of approximately nine years (one hour and twelve minutes for IP-1553 boards).

For IP-1553 and pre-firmware revision 2.55 PCI-1553 / PMC-1553 / cPCI-1553 / ISA-1553 / PCCARD-1553 model boards, the timer can be reset by software by writing to hardware register (0x04), by an external TTL input (if enabled: see 1553_control_reg description), or by the reception of a sync mode command (if enabled: see description of orphaned_hardware_bits).

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

rt_command_word, rt_status_word, and rt_data_words

These are the command, status, and data words input from 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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT Address					me	ins	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

rt_data_words

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

BC Memory Usage

Bus Controller messages are programmed into 16-word message blocks linked with one or two associated data buffers. Message blocks are located on any 32-byte boundary in RAM. Data buffers are located on any 80-byte

boundary in RAM. Each message block contains two pointers to the associated data buffers and 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.

Once the BC message blocks are set up and the BC Message Pointer Register (bcmsgpts) is set up, the BC Run bit may be set, which begins bus transmission. The hardware updates the bcmsgpts register at the start of each message and shadow it in another register (bcmsgptr) as it increments through the words of the message.

Separate message blocks, which do not transmit messages, are required for conditional branching in the message block sequence and to stop the BC in a single-pass major frame.

The 1553 control (bc_channel, bc_run, bc_busy, rt31_broadcast and sa31_mode_code bits), minor frame and response registers should also be set up before setting the bc_run bit. File registers for the interrupt enable conditions, retry information, flags, interrupt queue pointer, orphaned hardware bits, undefined mode code table, and endflags must also be set up before running the BC. Refer to the separate description for each of these registers.

Note: Data is stored in the BC data buffer for RT-to-RT messages beginning with firmware revision 2.20. This is not necessary, but prevents confusion over the fact that the BC data buffer is sometimes empty for real-time message buffer viewing. Now the buffer is never empty.

Aperiodic Message Lists

Revision 2.20 adds the capability to insert an aperiodic list of messages in a regular frame sequence. The list must be set up first. Then the software 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.

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.

Aperiodic message lists may contain linked BC message blocks that could be any combination of: messages, NOOPs, conditional sequencing control,

or a “stop BC” command, just like a regular BC message list. An aperiodic message may not be permitted to start on even 64K-word boundaries (0000 hex). The API ensures this condition.

High Priority

The “hpmsgpts” hardware register 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. 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 hpmsgpts 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 insufficient 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 a message or two for the one 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. There is hardware logic that 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 “lpmsgpts” hardware register is loaded by the API with the address of the first message of a list of low priority aperiodic messages. During the time after processing the message containing the “End-of-Frame” bit, the firmware polls this register. If the value of the register is non-zero, and the frame timer indicates that enough time remains in the minor frame, the firmware transmits the next message in the list of messages addressed by the lpmsgpts register.

At the conclusion of this message processing, if the next message pointer is non-zero, the process is repeated. 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 second 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_block

The bc_message_block is a 16-word buffer containing a 1553 message or message flow control for a bc message list. It *must* be programmed with valid values if the bc_message_pointer_save register (bcmsgpts) points to the block when the bc_run bit is set. If not, the board may behave inconsistently and should be reinitialized. The following description applies to aperiodic message lists and regular message lists. The mapping of the bc message block depends on the four least significant bits in the bc_control word (word 0) as defined in the following table:

Word	Message	Conditional	Stop BC	Noop
0	bc_control	bc_control	bc_control	bc_control
1	command_word1	test_word_address1	reserved	ignored
2	bc_error_inject_pointer	test_word_address2	reserved	ignored
3	intermessage_gap_time	data_pattern	reserved	ignored
4	data_pointer_a	bit_mask	reserved	ignored
5	data_pointer_b	cond_count_value	reserved	ignored
6	command_word2	cond_counter	reserved	ignored
7	status_word1	reserved	reserved	ignored
8	status_word2	reserved	reserved	ignored
9	bc_message_status1	reserved	reserved	ignored
10	bc_message_status2	branch_msg_pointer	reserved	ignored
11	next_msg_pointer	next_msg_pointer	next_msg_pointer	next_msg_pointer
12	reserved	reserved	reserved	reserved
13	reserved	reserved	reserved	reserved
14	reserved	reserved	reserved	reserved
15	reserved	reserved	reserved	reserved

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 branch_msg_pointer into the bc message pointer register (bcmsgpts) if the condition is true. If it is false, it loads the next_msg_pointer into the bc message pointer register (bcmsgpts). 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. 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 0010 hex.

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_next_msg_pointer is loaded into the bc_msg_pointer (bcmsgpts) register by the hardware in case the BC is turned on again without initializing this register. Stop_bc is not valid in the first message block of a minor frame.

noop

Executing a NOOP causes the microcode to load the next message pointer at word 11 into the firmware message pointer register (bcmsgpts) without processing a message. Words 1 through 10 are ignored, which allows the bc_message_block to be converted from a NOOP to or from a message, stop BC, or conditional by flipping a bit in the bc_control word (see below). If the NOOP is at the start and/or end of a minor frame, then the start minor frame and/or end minor frame bits in the bc control word should be programmed, respectively.

Note: A NOOP may be converted to a message at any time. A message may only be converted to a NOOP when the bc_busy bit in the 1553_control_register is not set.

bc_control

The bc_control word contains information passed between the API and the microcode on a message-per-message basis. It has been modified for firmware Revision 2.20. Software written for Revision 1.40 and earlier must be translated in order to function correctly with Revision 2.20 or later firmware.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rapi	res	res	res	retry	rtry	altb	bus	emf	dta	rtrt	mint	mnfr	ctl1	ctl0	noop

Field	Definition
noop	NOOP message block
ctl[1..0]	message control bits
mnfr	start minor frame

Field	Definition
mint	interrupt on this message
rtrt	RT-to-RT message format (moved in rev 2.00)
dta	use data buffer A
emf	end minor frame
bus	select bus: 0 = bus A, 1 = bus B
altb	alternate bus flag
rtry	retry enable
retry	retry flag
rapi	reserved for use by the API (see BusTools API documentation)

noop_message_block

This bit determines if the BC message, Stop BC, or Conditional is to be ignored. This allows you to turn on or off a message by flipping a single bit. This bit is active low (NOOP = 0).

message_control bits

These two bits determine the function of the bc_message_block. They are determined as follows:

bits 2-1:

00	NOOP
01	BC Message
10	Stop BC
11	Conditional

A value of 00 is not valid unless the noop_message_block bit is active (clear). If the microcode detects a value of 00 when the noop_message_block bit is not active, it shuts off the bc_run bit and sets the invalid bc control word (ibcerr) flag in the orphans register (0x5b). A value of 10, when the start_minor_frame bit is active, isn't valid and shuts off the bc_run bit and set the invalid bc control word (ibcerr) flag in the orphans register (0x5b).

start_minor_frame

This bit *must* be set at the first message block of each minor frame in a normal message list. Otherwise, it *must* be cleared. It must never be set for any aperiodic messages. The microcode tests this bit at the start of processing each message block. It is used to synchronize to the minor frame timer, set the bc_busy bit and detect minor frame overflow errors. The message_contol bits shall not define a Stop BC message block when the start_minor_frame bit is set.

interrupt_on_this_message

This bit is a qualifier for the bc_interrupt_enables register. It is normally a logic one (default), which enables the interrupt for this message, if the selected bits are set in the bc_interrupt_enables register. This may be used to select certain messages in the minor frame to generate an interrupt and others to not generate an interrupt. This bit is ignored for Conditional, Stop BC, and NOOP message blocks.

rt_rt_message_format

This bit is set by the software if the information transfer format for this message is a remote terminal to remote terminal (RT to RT) transfer. The microcode uses it to output a receive command followed by a transmit command onto the 1553 bus. This bit is ignored for Conditional, Stop BC, and NOOP message blocks.

use_data_buffer_a

When set, the data buffer used is pointed to by the data_pointer_a in the bc_message_block. When clear, data_pointer_b points to the data buffer used. The bit is initialized by the driver, normally set for data buffer “a” and subsequently toggled by the driver as it updates alternate buffers with more current data. This 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 message block has been processed. This bit must be set for the last message block of each minor frame in a regular message list and must be cleared otherwise. This is true even if the last message entry is a NOOP. It must never be set for any aperiodic messages. For Stop BC the microcode automatically clears the bc_busy bit, but the bit might be turned on anyway so that the message block can be turned into a NOOP by flipping a single bit (see noop_message_block above).

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.

alternate_bus_flag

This bit is an internal flag used by the microcode to determine if a retry is on the same or alternate bus.

retry_enable

This bit is set if a retry is to occur on either a no-response error or on selected status word bits. The no-response error is selected in the flags register, and the status word bits are selected in the bc_retry_info register. The purpose of this bit is to allow retries on selected messages only. This bit is ignored for Conditional, Stop BC, and NOOP message blocks. Retry status is presented in the bc_message_status2 word.

The number and type of retries is programmed in the bc_retry_buf. 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.

retry_flag

This bit is an internal flag used by the microcode to determine if a retry has occurred.

command_word1

This 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.

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

bc_error_inject_pointer

See the description for rt_error_inject_pointer.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pointer to error injection buffer															

intermessage_gap_time

The intermessage gap time is programmable from two to 65,535 microseconds, with the exceptions shown in following table.

Message Format	Minimum Inter-message Gap Time (following the message)	Minimum Inter-message Gap Time (Tested)
All non-Broadcast messages	4.0 us [less than this is out of spec]	2.0 us (no gap)
Broadcast Commands followed by 2 or more data words	4.0 us [less than this is out of spec]	2.0 us (no gap)
Broadcast messages followed by a single data word (including mode codes)	15.0 us	Implementing BM triggers (worst case): 14.0 us (non-mode codes) 15.0 us (mode codes)W/o implementing BM triggers: 12.0 us (non-mode codes) 13.0 us (mode codes)
Broadcast mode codes w/o a data word (except those listed below)	40.0 us	W/o implementing BM triggers: 33.5 to 35.0 us
Broadcast Inhibit Terminal Flag and Override Inhibit Terminal Flag mode codes	45.0 us	W/o implementing BM triggers: 42.5 us

This 16-bit value has a one-microsecond resolution. 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. The API programs a default of eight microseconds. For broadcast commands with zero or one data words programmed with less than the amount shown in the above table, the hardware outputs the next message as soon as it is able.

NOOP and conditional message blocks inserted between messages, although unlikely, may increase the minimum intermessage gap time. In this case, increase the time for deterministic intermessage gap times.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Time Between Messages															

data_pointer_a and data_pointer_b

These pointers point to two data word buffers. The first, buffer A, exists for all messages except mode commands without data words. The second, buffer B, exists if you elect to use alternating buffers. For receive messages, from one to 33 data words are stored in sequential order by the host, to be output on the bus. The 33rd word is provided for high word count error injection. For transmit messages, from one to 32 words are input from the 1553 bus and stored here.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pointer to data buffer A or B															

command_word2

This 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 hardware if it is not an RT-to-RT transfer.

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				

status_word1

This is 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.

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

Field	Definition
reserved	reserved bits
sr	service request
in	instrumentation
me	message error
rt_address	RT address bits

status_word2

This is the 16-bit status word associated with the receive command of an RT-to-RT transfer, input from the 1553 bus. It is not valid if it is not an RT-to-RT transfer.

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

bc_message_status

These two words contain discrete bits that are set by the hardware and used to generate interrupts. The host may read these words to determine the interrupt source or to obtain message status. Several of these bits may be set on the same message. The hardware doesn't attempt to prioritize errors. The word/bit structure of these two words is identical to the rt_interrupt_enables words and therefore they are described in the rt_interrupt_enables paragraph.

The RT and BM get loaded with the same message status so some of these bits may not have meaning for the BC.

bc_message_status1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nig	bce	wb	res	Bus	ira	lr	er	ncd	pe	2bus	mbe	is	lw	iw	hw

Field	Definition
hw	high word
iw	invalid word
lw	low word
is	inverted sync error
mbe	mid-bit error
2bus	two-bus error
pe	parity error
ncd	non-contiguous data

Field	Definition
er	early response
lr	late response
ira	incorrect rt address
bus	bus
res	reserved
wb	response on wrong bus
bce	bit count error
nig	no/short intermessage gap

bc_ message_status2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
res	bmo	te	tb	me	nres	rtry	rtb1	rtb0	res	mc	st	rrt	rtf	bcm	em

Field	Definition
em	end of message
bcm	broadcast message
rtf	RT-to-RT message format
rrt	reset rt
st	self test
mc	mode code
res	reserved
rtb0	rt busy bit 0
rtb1	rt busy bit 1
rtry	retry occurred
nres	no response
me	message error bit
tb	API use: trigger begin
te	API use: trigger end
bmo	API use: bus monitor overflow

bc_next_msg_pointer

This word contains the address of the next bc_message_block to be executed.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pointer to next BC message															

test_word_address (2 words)

These two words contain a single pointer to the word to be tested for conditional message flow. For IP-1553 boards, only the first word is used. It contains the 16-bit word address.

Note: For PCI-1553, cPCI-1553, and PMC-1553 boards the first word (word 1) contains the 16 most significant bits of the 19-bit address. The second word (word 2) contains the three least significant bits of the address in its three least significant bit positions.

test_word_address0 (word 1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
b18	b17	b16	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3

test_word_address1 (word 2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	b2	b1	b0

The test_word_address may point to any word in RAM, but is generally a command word, status word, data word, or message status word. To take some burden off the application, there may be limits placed by the API as to what words may be tested. Refer to “BusTools/1553-API User’s Manual and Hardware Installation Guide” for further information.

data_pattern

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 will contain logic ones, and if they are not masked out, cause the test to fail.

bit_mask

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.

cond_count_value and cond_counter

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 16-bit value (max. 65,536 counts), 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_retry_buf

The API can program up to eight retries and specify using the same or alternate bus for each retry. The BC_RETRY_BUF is a 9-word buffer located anywhere in memory containing the following information:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
(word 1) 0														ctrl	
(word 2) 0														ctrl	
(word 3) 0														ctrl	
(word 4) 0														ctrl	
(word 5) 0														ctrl	
(word 6) 0														ctrl	
(word 7) 0														ctrl	
(word 8) 0														ctrl	
(word 9) 0															

Field	Definition
ctrl	00 - last retry 01 - retry on same bus 10 - retry alternate bus

BC_RETRY_BUF(0) is for the first retry up BC_RETRY_BUF(7) is for the eight retry. You must program the last retry with the “last retry” code of “00”. This is BC_RETRY_BUF(0) if you want no retries and is BC_RETRY_BUF(8) if you program 8 retries.

Note: There may be a intermessage gap time limitation for the use of retries. The minimum intermessage gap time allowed when using retries is 5 microseconds.

BM Memory Usage

There are two phases of 1553 message acquisition:

- Storing messages in on-board RAM.
- Informing the host of the acquisition by writing message information to the interrupt queue.

When you run the Bus Monitor (BM), it stores all unfiltered 1553 bus messages in RAM. 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-word bm_filter, which in turn contains a pointer to the bm_control_buffer. The bm_control_buffer determines if this particular message should be captured 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 message buffer pointer (register) is updated with the address of the next message, allowing for a programmable circular buffer.

Host acquisition begins as soon as the BM is run. If a trigger is specified, it waits until all trigger events have occurred. In either case, if the interrupt is enabled, the interrupt queue is updated, and an interrupt is generated to the host upon message completion. The “interrupt” mentioned here may be polled and need not generate a physical interrupt. Capturing continues until a stop trigger occurs (not presently available) or a halt command turns off the bm_run bit. The number of messages stored prior to the trigger is determined by the driver software; how large the circular buffer is, and how fast the software can grab pre-trigger messages before post-trigger messages write over them.

BM memory usage consists of four buffers, which may be located anywhere within memory with a few restrictions on base address boundaries. The four buffer types may be placed in any order, but are described in the following order:

- bm_filter_buffer
- bm_control_buffer
- bm_message_buffer
- bm_trigger_buffer

bm_filter_buffer

The bm_filter_buffer is similar to the rt_filter_buffer. It is used for pre-filtering down to the sub-address level. If such filtering is not required, then multiple pointers point to the same entry in the bm_control_buffer. This buffer is restricted to have a base address at a 2K word (4K-byte) in on-board memory.

The bm_filter_buffer contains 2,048 words, 2 words (transmit and receive) for each of 32 subaddresses of 32 RT addresses. Each word contains a pointer to an associated bm_control_buffer. 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.

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 (word 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 (word 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 base offset 0x350 (word 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 base offset 0x7FF

bm_control_buffer

This buffer contains four words of information for each unique pointer in the bm_filter. A separate four-word 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 eight words (for no filtering) to as large as 8K words for filtering of all possible 32 RT's, transmit/receive bits, and subaddress combinations filtered separately. Generally, this buffer may be quite small (no larger than 256 words).

One solution is to create a four-word buffer for each of the 2K combinations, which requires 8k words, but has the advantage of simplicity. The contents of each buffer would then be repeated many times if little or no filtering is used. Nth occurrence would be valid only for unique address/subaddress/t_r combinations if this solution is used.

The first word of each entry is pointed to from the bm_filter_buffer and may be located anywhere within memory. However, the five words must comply with the following sequence.

- bm_enable_word_count1 or bm_enable_mc1
- bm_enable_word_count2 or bm_enable_mc2
- bm_nth_occurrence (2 words)

bm_enable_word_count1 and bm_enable_word_count2

These two words are used to filter down to the word level. They have the same format as the RT illegal word counts, except that setting a bit *enables* the corresponding word count. For RT-to-RT messages, if either command word passes through the filter, then the message will be enabled.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
word_count14 - word_count32															
word_count31 - word_count15															

		hexadecimal equates	
word_count	32	0x00000001	(Long 32 bit equate)
word_count	1	0x00000002	Defines legal word count
word_count	2	0x00000004	
	...	...	
word_count	29	0x20000000	
word_count	30	0x40000000	
word_count	31	0x80000000	

bm_enable_mc1 and bm_enable_mc2

Some of these mode codes are described in the "rt_legal_word_count1 and rt_legal_word_count2" section of this document. Refer to the MIL-STD-1553B Specification for a full description of mode codes.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							rrt	oitf	itf	oxsd	xsd	stst	tsw	sync	dbc
Reserved										osxs	sxs	tbw	tlc	sw	tvw

Field	Definition
	(bm_enable_mc1 definitions with data words)
dbc	dynamic_bus_control
sync	synchronize
tsw	transmit_status_word
stst	initiate_selftest
xsd	transmitter_shutdown
oxsd	override_xmit_shutdown
itf	inhibit_terminal_flag_bit
oitf	override_inhibit_tf_bit
rrt	reset_remote_terminal
reserved	reserved bits
	(bm_enable_mc2 definitions w/o data words)
tvw	transmit_vector_word
swo	synchronize
tlc	transmit_last_command
tbw	transmit_bit_word
sxs	selected_xmitter_shutdown
osxs	override_sel_xmt_shutdown
reserved	reserved bits

bm_nth_occurrence

Note: New designs should avoid using this feature as it is intended for future firmware revisions to count a single nth occurrence for all unfiltered messages.

This function provides the capability to capture every message (when the BM is running, the channel is enabled, and the message isn't filtered out) and interrupt the host CPU only on every Nth occurrence, if the interrupt is enabled. The host is not slowed down with servicing interrupts, and can read N messages at a time on a single interrupt, without losing data. The messages are in a link list, where the first word of each message points to the first word of the next message, which may be useful in tracking the message sequence. The buffer size of N buffers must be somewhat smaller than the total on-board memory allotted for the messages so that data does not get overrun.

This function consists of two words: the first is the message count, N, and the second is the internal count kept by the microcode. The software must initialize both counters to the same value (default is 0001 hex). The microcode decrements the second word each time the message is received and re-loads the first word into the second when the count reaches zero.

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

bm_message_buffer

The bm_message_pointer addresses this 84-word buffer, which is used to store 1553 messages and status. The command, status, response time, and data words are stored in the buffer in the order that they were received on the bus, interleaved with a separate message status word for each 1553 word. Only one extra word is saved on high_word errors in order to limit the size of this buffer. Different data structures are required for BC-RT, RT-BC, RT-RT message formats and different number of data words. Broadcast messages have the same format as BC-RT except that the response time and status word are not valid. Similarly, Broadcast messages have the same format as RT-RT except that bm_response_time2 and bm_status_word2 are not valid.

This buffer contains the receive and transmit messages that are associated with the 1553 Bus. Each message is configured into an 84-word buffer. Each buffer contains the following data elements in the given order:

Word	BC-RT (receive)	RT-BC (transmit)	RT-RT
0	bm_next_message_pointer	bm_next_message_pointer	bm_next_message_pointer
1	bm_interrupt_enable1	bm_interrupt_enable1	bm_interrupt_enable1
2	bm_interrupt_enable2	bm_interrupt_enable2	bm_interrupt_enable2
3	bm_message_status1	bm_message_status1	bm_message_status1
4	bm_message_status2	bm_message_status2	bm_message_status2
5	bm_time_tag0	bm_time_tag0	bm_time_tag0
6	bm_time_tag1	bm_time_tag1	bm_time_tag1
7	bm_time_tag2	bm_time_tag2	bm_time_tag2
8	bm_command_word	bm_command_word	bm_command_word
9	bm_message_status_cw1	bm_message_status_cw1	bm_message_status_cw2
10	see note below	bm_response_time1	bm_command_word2
11	see note below	bm_status_word1	bm_message_status_cw1
12	see note below	bm_message_status_sw1	bm_response_time1
13	see note below	see note below	bm_status_word1
14	see note below	see note below	bm_message_status_sw1
15-84	see note below	see note below	see note below

The remaining buffer structures depend on the number of data words input on the 1553 bus.

Words 10 through 84 for BC-RT are in the following sequential order:

- bm_data_words (2 words per data word)
- bm_response_time1 (1 word)
- bm_status_word1 (2 words)

Words 13 through 84 for RT-BC are in the following sequential order:

- bm_data_words (2 words per data word)

Words 15 through 84 for RT-RT are in the following sequential order:

- bm_data_words (2 words per data word)
- bm_response_time2 (1 word)
- bm_status_word2 (2 words)

bm_next_message_pointer

This pointer is loaded into the bm_message_pointer_save register. This provides for multiple message processing before the host needs to be interrupted.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bus Monitor's next message pointer															

bm_interrupt_enables and bm_message_status

The host is interrupted only at the end of a message on message errors or on specific 1553 protocol status if the appropriate enable bits are set. The first of these two words contain enable bits to interrupt on specific error status of each message. The second word contains enable bits to interrupt on specific 1553 protocol status. All unused bit positions must be programmed with logic zeros by the software.

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 these two registers to enable the interrupt conditions of the BM. The bits are identical to the similarly named bits in the rt_interrupt_enables, rt_ message_status, bc_interrupt_enables and bc_message_status words. See Chapter 8, “RAM Usage”, rt_interrupt_enables for a complete description of these bits.

The 32-message status bits are set by the hardware or microcode to inform the host of message status. They may be used to generate interrupts if the corresponding bit is set in the bm_interrupt_enables word. The first of the

two bm_message_status words are saved in the bm_message_buffer for each command, status, and data word in the message. The message status words presented here are the logical OR of each 1553 word's message status word, plus additional message status bits, which pertain to the whole message.

bm_interrupt_enable1 (and bm_message_status1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nig	bce	wb	res	ch	ira	lr	er	ncd	pe	2bus	mbe	is	lw	iw	hw

Field	Definition
hw	high word
iw	invalid word
lw	low word
is	inverted sync error
mbe	mid-bit error
2bus	two-bus error
pe	parity error
ncd	non-contiguous data
er	early response
lr	late response
ira	incorrect rt address
ch	channel
res	reserved
wb	response on wrong bus
bce	bit count error
nig	no/short intermessage gap

bm_interrupt_enable2 (and bm_message_status2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
res	bmo	te	tb	me	nres	rtry	rtb1	rtb0	res	mc	st	rrt	rtf	bcm	em

Field	Definition
em	end of message
bcm	broadcast message
rtf	RT-to-RT message format
rrt	reset rt
st	self test
mc	mode code
res	reserved

Field	Definition
rtb0	rt busy bit 0
rtb1	rt busy bit 1
rtry	retry occurred
nres	no response
me	message error bit
tb	API use: trigger begin
te	API use: trigger end
bmo	API use: bus monitor overflow

bm_time_tag

Same time tag words as described for the rt_time_tag words.

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

bm_response_time1 and bm_response_time2

The 1553 status response times are represented as two 6-bit binary numbers with the LSB in bit 0, having a 500-nanosecond resolution. Response time 1 is valid for all 1553 messages for which there are responses; response time 2 is valid only for the receiving RT of an RT-to-RT transfer.

bm_repsonse_time1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused										Response Time 1					

Precedes the status word for RT-to-RT and broadcast RT-to-RT transfers.

bm_repsonse_time2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused										Response Time 2					

Response Time 2 is valid only preceding the receiving RT's status word for RT-to-RT transfers.

bm_command_words, bm_status_words, and bm_data_words

These are the command, data, and status words of the 1553 message. Two command words and two status words are required for RT-to-RT transfers. The command, status, and data words are stored in the buffer in the order that they are received on the bus. high_word errors are reported in the bm_message_status and the first extra data word is saved in this buffer.

bm_command_word1

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				
This word's status (First word of msg. status, as it applies.)															

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_command_word2

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				
This word's status															

bm_status_word1

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
This word's status															

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

selected_xmitter_shutdown and override_sel_xmt_shutdown

Selected_xmitter_shutdown and the override_ sel_xmt_shutdown do not alter the state of the transmitters as the board only simulates a dual-redundant RT. The BC must use transmitter_shutdown and override_transmitter_shutdown mode commands to turn on or off the transmitters for each channel.

bm_data_words

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Words															
This word's status															

bm_status_word2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT Address					me	ins	sr	Reserved			br	by	sf	db	tf
This word's status															

Status Word 2 is only valid with RT-to-RT transfers.

bm_trigger_buffer

The bus monitor trigger buffer consists of six words: a header word and five words a single event. This buffer must be programmed during board setup, with no driver interaction required during run time except to clear the trigger if re-triggering is desired. The current firmware version implements only a single event to start triggering and does not support stop triggering.

- bm_trigger_header
- bm_trigger_event_buffer (5 words)

bm_trigger_header

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							tem	xot	Res	terr	tint	ext	henb	senb	inte

Field	Definition
inte	bm_interrupt_host
senb	enable start trigger
henb	enable stop trigger
ext	enable external trigger

Field	Definition
tint	trigger (internal use)
terr	trigger on errors
Res	reserved for trigger on errors only
xot	external output on trigger
tem	Trigger on every message
reserved	reserved bits

bm_interrupt_host

When the bm_run bit of the 1553_control_register is set the bus monitor will capture 1553 messages and store them in the circular buffer that has been setup in RAM. The interrupt queue isn't updated for 1553 bus messages, however, until bm_interrupt_host is set. This allows you to store messages before a trigger point as well as after a trigger point.

If trigger events are not implemented, the bm_interrupt_host bit is set prior to executing the run command, which is the default. If implementing trigger event(s), the bm_interrupt_host bit is set by the hardware when the enable_start_trigger bit is set and all of the events occur in the given order. The host can determine the message that the trigger occurred on by a bit set in the interrupt queue.

The bm_interrupt_host bit may also be armed by an external TTL input if the enable_external_trigger bit is set. It is set when the next message is input on the 1553 bus.

At present, "capturing" messages to the interrupt queue is stopped only when the bm_interrupt_host bit is cleared by software.

enable_start_trigger

If the Bus Monitor is to be triggered by a 1553 bus event, then the software simultaneously sets the enable_start_trigger bit and clears the bm_interrupt_host bit. Once the trigger event(s) occur, the hardware turns this bit off and turn on the bm_interrupt_host bit.

Multiple or repetitive starting and stopping of queuing Bus Monitor interrupts requires host interaction.

trigger

This bit is used internally by the hardware to detect if the first event occurred in a multiple event trigger. It must be cleared by the driver before bm_run is set, but should never be altered while bm_run is set.

enable_external_trigger

This bit provides the capability to use a trigger source hardwired externally to the board. The bit may be set or cleared at any time. The external trigger is disabled if either the start_trigger is set or when the enable stop_trigger is set.

trigger_on_errors

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

external_output_on_trigger

Setting this bit causes a 50 microsecond active high pulse on the EXT OUT port when the 1553 bus trigger or external input trigger occurs.

bm_trigger_event_buffer

The five words are in the following sequence:

- bm_event_type
- bm_trigger_event_bit_mask
- bm_trigger_event_value
- bm_event_count
- bm_trigger_control

bm_event_type

This word includes three bits for the type of event to trigger on and five bits to determine which data word to trigger on if data type is selected (bit 7 is set). The word has the following format.

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

Field	Definition
com	command
sta	status

bm_trigger_event_bit_mask

This word is logical ANDed with the bm_trigger_event_value to test only a single bit or field (multiple bits) of the bm_trigger_event_value. If no mask is specified, this word contains FFFF hex 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															

bm_trigger_event_value

This is the value of the command, status, or data word to be used as the trigger event. This value is ANDed with the associated bm_trigger_event_mask. The result is compared with each word input on the 1553 bus of similar type to determine if a single count of this event has occurred.

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

bm_event_count

The bm_event_occurred bit is set after the event occurs the number of times specified in this word. The 16-bit binary count is programmed prior to setting the bm_run bit and is decremented by the hardware each time the event is detected. The count must be re-loaded by the driver if it is to be used for subsequent trigger(s).

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

bm_trigger_control

For the current firmware revision, bm_trigger_control sets the end bit and clear all other bits. This word contains the logical operator, arms bit, end bit, and bm_event_occurred flag.

The driver enters the logical operator on the event of the second operand. The first operand and the first operand following an arm always requires the 'or' logical operator (the first operand is ORed with 0). The bit is set for 'and' and clear for 'or'. The 'arms' bit is set on the event preceding what it arms. For each word input on the 1553 bus, the hardware processes the events until the 'end' bit is encountered or until the 'arms' bit is encountered and the arming events have not yet occurred. The logic hierarchy is determined by the order in the list.

The bm_event_occurred flag is used internally by the hardware as a flag to indicate that the specified event has occurred. The driver must clear this flag prior to setting bm_run if triggering is to be used.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused								end	0	0	evo	0	0	0	0

Field	Definition
evo	event occurred
end	identifies the last event to be processed

Playback

The Playback function available on most of Condor Engineering's MIL-STD-1553 boards (not applicable to IP-1553 model boards) reads data from the Playback Data Buffer. The data in the Playback Data Buffer is preceded by a message code that describes the data type and how much data follows.

Message Code Format

The Playback Data Buffer uses the following message code format.

Message Code Format		
Bit(s)	**Function**	**Description**
15	Channel	Valid only during xmit message codes 0 = 1553 channel A used 1 = 1553 channel B used
14	Xmit message code	xmit – When set, this bit indicates a transmit message code. If the bit count error bit is cleared the hardware will expect the number of data words indicated in bits 5 through 0 to follow the message code. If the bit error count bit is set the hardware will expect two words to follow the message code as explained in the bit count error bit description
13	Gap message code	gap – the following two words contain the time to be inserted between transmitted words. The first word following the gap time message code

Message Code Format		
Bit(s)	**Function**	**Description**
		contains the lower 16 bits while the second word is the 16 higher bits of the 32 gap time. The timer has a resolution of 500 ns. The host must calculate the gap time from the word count, time tag, and response time information found in the bus monitor message buffer.
12	End of playback	When set this bit informs the hardware the end of the playback data has been reached. It will also clear the run bit (bit 2) in the playback status register.
11-9	Not used	–
8	Parity error	Valid only during 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 set this bit indicates the sync pattern preceding each word is a high followed by a low (nominally command and status sync). If this bit is clear then the sync pattern preceding each word is a low followed high (nominally data sync).
6	Bit count error	▪ Valid only during xmit message codes. ▪ When the bit count error is set, it indicates that two words will follow the message code and bits [5..0] of the xmit, message code indicate the number of significant bits contained in the two words. The first bits sent is the MSB of the first word while the last bit sent is the LSB of the second word. ▪ When the bit error count bit is cleared bits [5..0] indicate the number of words that follow the message code.
5-0	Word count/ Bit count	▪ When the bit count error bit (bit 6) is clear this field indicates number of words following the message code to be transmitted on the 1553 bus for xmit message codes. Valid word counts are 1 through 63. ▪ When the bit count error bit (bit 6) is set this field indicates the number of significant bits contained in the two words following the 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. ▪ This bit field is ignored for gap message codes.

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**		*intermessage gap*		
XMIT MESSAGE CODE (command)	DATA (command)	XMIT MESSAGE CODE (n data words)	DATA (word 1)	…	DATA (word n)	GAP MESSAGE CODE (response time)	DATA (gap LSW)	DATA (gap MSW)	XMIT MESSAGE CODE (status)	DATA (status)	GAP MESSAGE CODE (gap time)	DATA (gap LSW)	DATA (gap MSW)

RT-BC Transfer

TRANSMIT COMMAND		*response time*			**STATUS WORD**		**DATA WORD 1**		…	**DATA WORD n**	*intermessage gap*		
XMIT MESSAGE CODE (command)	DATA (command)	GAP MESSAGE 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)	GAP MESSAGE 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)	GAP MESSAGE 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**		*intermessage gap*		
GAP MESSAGE CODE (response time)	DATA (gap LSW)	DATA (gap MSW)	XMIT MESSAGE CODE (status)	DATA (status)	GAP MESSAGE CODE (gap time)	DATA (gap LSW)	DATA (gap MSW)

Error Injection Buffer

The words in this buffer determine the error(s) 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. The buffer consists of a number of specific error_inject_buffers of 34 words each. The number of error_inject_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 format for all words in the buffer follows.

error_injection_word

There are 34 error injection words per message.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
err_enc			ira	pres	wce	0	pe	ie	bce	err_inf					

Field	Definition
err_inf	information associated with the error to be injected.
bce	bit count error (BC, RT)
ie	inverted sync error (BC, RT)
pe	parity error (BC, RT)
wce	word count error (BC, RT)
pres	programmable response (RT)
ira	respond_to_wrong_address (RT)
err_enc	encoded error 000 = none 001 = Mid-sync Zero-Crossing error (BC, RT) 010 = Mid-bit Zero-Crossing error (BC, RT) 011 = reserved 100 = reserved 101 = reserved 110 = Bi-phase error (BC, RT) 111 = reserved

error_info

These five bits qualify some of the injected errors.

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 bit_count register representing the bit count in binary. The counter accepts values from 1 to 32. The RT validation test plan specifies only use of values 14, 15, 17, 18, or 19. The bit_count_error is programmed on a word-per-word basis.

inverted_sync

This bit set inverts the sync bit of the associated word.

parity_error

This bit inverts the parity bit of the associated word.

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.

programmable_response

This RT selection allows the application to enter a programmable response time of up to 31½ microseconds, 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 8-microseconds *are not guaranteed* to be attainable by the hardware.

respond_to_wrong_address

This bit set causes the RT to respond with the status word using the address programmed in the five LSBs of the error_info bits. Otherwise, it will respond with the correct address.

zero_crossing_errors

These error injection codes delay the zero crossing point by 300 nanoseconds from where it is expected. For mid-bit zero-crossing errors, the four lsb's of the error_info bits determine the selected bit 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 transmitted.

bi_phase_error

This error injection inhibits a zero crossing for the selected bit. The four lsb's of the error_info bits determine the selected bit: 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.

A bi-phase bit error is when there is no zero-crossing for the entire bit time. It is not predictable how a 1553 decoder will interpret a word when this error is injected on 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. Therefore, the API does not support injection in either of the first two bits, but the user should be aware of the possible behavior should this case occur at the decoder.

Interrupt Queue

An interrupt queue keeps a history of interrupt events. It is written to by the hardware 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 interrupts so that it need not respond to them immediately.

The host determines the length of the queue by writing next_interrupt-queue_pointer values during board initialization. It may be made any length, provided there is sufficient RAM, but must be three words minimum. Each interrupt requires three words and the last interrupt buffer should point back to the first. The interrupt_queue_pointer (a file register) points to the first word of the current member of the queue and is initialized by the host. The hardware clears the interrupt_acknowledged bit, sets the interrupter_mode bit, writes the message block or response buffer pointer, and updates the interrupt_queue_pointer with the next_interrupt_queue_pointer for each interrupt event. It then triggers the hardware interrupt line to the host. When servicing the interrupt, the host detects the source of the event (BC, BM, RT or tag timer), uses the message_pointer to gather further information using error_detect status, clears the (BC, BM, or RT) message_status word, and sets the interrupt_acknowledge bit. The host services each event in the circular queue until it finds an event with the interrupt_acknowledged bit set. The three words are:

- interrupter_mode
- message_pointer
- next_interrupt_queue_pointer

The application must poll the interrupt acknowledge bit at a given interval, and read in all messages from where the last poll left off, up to the current interrupt_queue_pointer location. The polling interval must be less than the time it takes to overflow the buffer.

The two interrupt enable words for BC, BM, and RT messages are ANDed with the corresponding two message status words for the firmware to set an interrupt in the interrupt queue. In addition, each BC message has an on/off bit for specific message selection.

For Bus Monitor applications only, one may alternatively use the Nth occurrence feature to interrupt on 'one out of every N' messages, and read N messages once the Nth message sets the interrupt.

interrupter_mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										tp	bc	bm	rt	tmr	iack

Field	Definition
iack	interrupt acknowledge (written to by host CPU)
tmr	tag timer load
rt	remote terminal interrupt
bm	bus monitor interrupt
bc	bus controller interrupt
tp	BM trigger point

tag timer load

If this bit is set, the tag timer has been loaded with the contents of the tag time counter load (TTCL) register.

bm_trigger

This bit is set in addition to the "bm" bit when the message monitored is the one that triggered capturing of data, or triggered stop capturing data. It is not set if you select "unconditional trigger".

message_pointer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Points to the message (either BC, BM, or RT) that caused the interrupt.															

next_interrupt_pointer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Points to the next interrupt in the queue.															

CHAPTER 9

PCI Configuration Space

PCI Configuration Space

This chapter applies to PCI-1553, cPCI-1553, and PMC-1553 model boards only.

The PCI configuration space is fully described in the PCI Local Bus Specification, Revision 2.1. It is available from the PCI Special Interest Group, P.O. Box 14070, Portland, OR 97214. (800) 433-5177 (U.S.). (503) 797-4207 (international).

Specifics for Condor Engineering PCI-1553, cPCI-1553 and PMC-1553 model boards are described here.

Vendor ID

Condor Engineering's Vendor ID is 13C6 hex.

Device ID

Condor Engineering's Device ID is 1553 hex for all MIL-STD-1553 boards.

Command Register

- Bit 0, I/O Access Enable, is not relevant and should always be clear.
- Bit 1, Memory Access Enable, must be set in order to access the board's memory space.
- Bit 2, Master Enable, is not relevant and should always be clear.
- Bit 3, Special Cycle Recognition, is not relevant and should always be clear.
- Bit 4, Memory Write and Invalidate Enable, is not relevant and should always be clear.
- Bit 5, VGA Palette Snoop Enable, is not relevant and should always be clear.
- Bit 6, Parity Error Response, cannot be set.
- Bit 7, Wait Cycle Enable, cannot be set.
- Bit 8, System Error Enable, cannot be set.
- Bit 9, Fast Back-to-Back Enable, cannot be set.
- Bits 15-10 are reserved and cannot be set.

Status Register

- Bit 5, 66Mhz-capable is always clear.
- Bit 6, UDF Supported is TBD.
- Bits 10-9, Device Select Timing, is 10 binary, or *slow*.

Revision ID

Revision ID is used in conjunction with the Board ID bits (CSC Register, byte 1). Production units are designated beginning with 01 hex and incrementing forward. The Revision ID identifies the PCB and PCI Interface chip firmware only. Microcode and 1553 Local Processing Unit (LPU) firmware is downloaded by the software.

Class Code

For PCI-1553, cPCI-1553, and PMC-1553 models, the Class Code is ff0000 hex.

Subsystem ID

For PCI-1553, cPCI-1553, and PMC-1553 models, the Subsystem ID is 0000 hex.

Subsystem Vendor ID

For all PCI-1553, cPCI-1553, and PMC-1553 models, the Subsystem Vendor ID is 13C6 hex.

Base Address Registers

For all PCI-1553, cPCI-1553, and PMC-1553 models, only one contiguous 32M-byte memory address space is required and no I/O space is required. Therefore, only the first Base Address Register in PCI configuration space will be used. The adapter board resides in 32-bit address space.

Interrupt Pin

This register contains a value of 1 as the board connects to the INTA# pin. Refer to Chapter 5, Hardware Register Usage, 1553_control_reg (cpu_interrupt and cpu_interrupt_enable) for interrupt usage.

Interrupt Line Register

This value is system dependent.

CHAPTER 10

IP-1553 and IP-D1553 ID PROM

ID PROM

This chapter applies to IP-1553 and IP-D1553 model boards only.

Every IP contains an ID PROM to assist in software configuration. Condor IP modules have a 32x8 bit PROM, which is shown in the following table (all values are in hex).

Address	Function	Value
0x01	ASCII “I”	0x49
0x03	ASCII “P”	0x50
0x05	ASCII “A”	0x41
0x07	ASCII “C”	0x43
0x09	Manufacturer ID	0x79
0x0B	Model number	*
0x0D	Revision	**
0x0F	Reserved	0x00
0x11	Driver ID, low byte	**
0x13	Driver ID, high byte	**
0x15	Number of bytes used	0x13
0x17	CRC	**
0x19-0	Unused	0x00
0x1B	Unused	0x00
0X1D	Unused	0x00
0X1F	Hardware ECO Level	**
0x21	PLD Major	**
0x23	PLD Minor	**
0x25	PCB Revision	**
0x27-3F	Unused	0x00

* Value is 05 for IP-1553-M, 08 for IP-1553-S, 0C for IP-D1553-1M, 0D for IP-D1553-1S, 0E for IP-D1553-2M and 0F for IP-D1553-2S.

** See README.DOC included in the distribution software for this release for values as shipped.

For IP-D1553 boards, this ID PROM is shadowed in memory space in order to ease diagnostic and troubleshooting.

CHAPTER 11

IRIG

IRIG Memory Space

This chapter applies QPMC-1553 model boards only.

Memory Address Byte Offset	Write Only	Register
0x000802 – 0x000803	W only	time tag control, ch1
0x200802 – 0x200803		time tag control, ch2
0x400802 – 0x400803		time tag control, ch3
0x600802 – 0x600803		time tag control, ch4
0x100 - 0x101	W only	irig dac
0x102 - 0x103	R/W only	irig control
0x104 - 0x105	W only	irig toy[15..0]
0x106 - 0x107	W only	irig toy[31..16]
0x108 - 0x109	R/W	discrete in/out [15..0]
0x10A - 0x10B	R/W	discrete in/out [18..6]
0x10C - 0x10D		

Time Tag Counter Control Register

See Chapter 6, “Hardware Register”, for the definition of this register.

IRIG DAC

The IRIG dac sets the threshold level for detecting either AM or DC level shifted IRIG data. The threshold set according to the formula:

$$V_{threshold} = 3.3V * n/255$$

where *n* is the decimal equivalent of bits 7..0 written to the IRIG dac register. The default value written to this register should be 2 Volts or 22 hex.

IRIG Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused												val	int	out	cal

cal Read only. Set when the conditioned IRIG input exceeds the threshold set with the IRIG dac. This bit is cleared when the read is completed.

int Read/Write. When set, the IRIG source is internal. Otherwise, it comes from the external IRIG-encoder.

out Read/Write. When set, the internal IRIG encoder outputs valid IRIG data based on the TOY data. When cleared, the output is driven low.

val Read only. Set if IRIG time code input is valid. Otherwise, clear.

IRIG Time of Year (TOY)

29:28	27:24	23:20	19:18	17:14	13:11	10:7	6:4	3:0
100s Of Days	Tens Of Days	Units Of Days	Tens Of Hours	Units Of Hours	Tens Of Minutes	Units Of Minutes	Tens Of Seconds	Units Of Seconds

The lower word *must be written first* and is latched in a hardware buffer. When the upper word is written, the entire TOY is presented to the IRIG encoder.

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP