Universal Core Architecture Reference Manual

MIL-STD-1553 Products

Supporting Products:

- AMC-1553
- cPCI-1553
- IP-D1553
- ISA-1553
- PCCARD-1553
- PCCARD-D1553
- RPCCARD-D1553
- PCI-1553
- PMC-1553
- Q104-1553
- Q104-1553-P
- QCP-1553
- QPCI-1553
- QPCX-1553
- QPM-1553
- QPMC-1553
- RXMC2-1553
- QVME-1553
- QVME2-1553
- R15-AMC
- R15-EC
- RPCIe-1553
- RXMC-1553
- VME-1553
- R15-LPCIe

abaco SYSTEMS

Copyrights

MIL-STD-1553 Universal Core Architecture Reference Manual (1500-003)

Firmware Revision: 5.22

Document Revision: 1.01

Document Date: 16 February 2018

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

support@abaco.com (email)

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

Additional Resources

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

www.abaco.com

Contents

CHAPTER 1

Introduction

Introduction

This document describes Abaco Systems' powerful Universal Core Architecture (UCA) 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 Abaco Systems' MIL-STD-1553. 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, see the "BusTools1553-API Software User Manual". For a hardware description, see the MIL-STD-1553 Hardware Installation Manual.

The following information can be found in this manual:

- Chapter 1 describes general interface requirements and resource mapping for Abaco Systems MIL-STD-1553 boards using the Universal Core Architecture. These are:

AMC-1553	QPCX-1553
cPCI-1553	QPM-1553
IP-D1553	QPMC-1553
ISA-1553	QVME-1553
PCCARD-1553	QVME2-155
PCCARD-D1553	R15-AMC
RPCCARD-D1553	R15-EC
PCI-1553	RPCIe-1553
PMC-1553	RXMC-1553
Q104-1553	RXMC2-1553
Q104-1553-P	R15-LPCIe

QCP-1553	VME-1553
QPCI-1553	VXI-1553

- Chapter 2 describes the host interface of all boards with the exception of the native VME based boards.
- Chapter 3 describes A16 Configuration Register for all native VME based boards.
- Chapters 4 through 9 apply to all model boards.
- Chapter 10 describes optional I/O functionality for IRIG, Avionics discretes, hardwired RT addressing and trigger sources
- Chapter 11 discusses special-order options which affect the memory map.

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.

Interface Descriptions

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

The PCI-1553 is offered 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 offered 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 offered 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 offered 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 offered 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 PCCARD-D1553 and RPCCARD-D1553 are documented in the next section.

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.

For a complete description of PCI Configuration Space, see chapter 8, “PCI Configuration Space”.

The QPCI-1553 uses a 128-byte PCI configuration space for the PLX (BAR0), a 128 byte PLX I/O space which is required but not actually used (BAR1) and an 8MB memory region (BAR2).

The QPCX-1553 allocates a 512-byte PCI configuration space for the PLX (BAR0), a 256 byte PLX I/O space which is required but not actually used (BAR1) and an 8MB memory region (BAR2).

The R15-AMC, RPCIe-1553, R15-LPCIe, AMC-1553, QPM-1553, R15-EC, RXMC-1553, RXMC2-1553, and QPMC-1553 use a 256-byte PCI configuration space and an 8MB memory region.

The QCP-1553 uses a 128-byte PCI configuration space, a 512 byte memory region located at Base Address Register 0, a 256 byte I/O

space located at BAR1, and an 8MB memory region located at BAR2.

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 *a[24]* determines the 1553 bus number; *a[23]* 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: *a[13]* 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 *a[10]* 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

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)	CH 1 RAM (512K words)
0300000-03FFFFF	N/A	N/A	CH 1 RAM (reserved for expansion)
0400000-04FFFFF	1000-13FF	1000-13FF	CH 1 LPU Hardware Registers
0500000-05FFFFF	1400-17FF	1400-17FF	CH 1 Register File
0600000-06FFFFF	1800-1FFF (PAGED)	N/A	CH 1 WCS (1st 64K words used; rest repeats)
0700000-07FFFFF	N/A	N/A	CH 1 Undefined
1200000-12FFFFF	2800-2FFF (PAGED)	N/A	CH 2 RAM (512K words)
1300000-13FFFFF	N/A	N/A	CH 2 RAM (reserved for expansion)
1400000-14FFFFF	3000-33FF	N/A	CH 2 LPU Hardware Registers
1500000-15FFFFF	3400-37FF	N/A	CH 2 Register File
1600000-16FFFFF	3800-3FFF (PAGED)	N/A	CH 2 WCS (1st 64K words used; rest repeats)
1700000-17FFFFF	N/A	N/A	CH 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 are valid ISA / PCCARD transactions. The page register must be set-up prior to accessing RAM and WCS.

VME/VXI/QVME/QVME2-1553

The VME/VXI/QVME/QVME2-1553 are sold in six configurations, all of which are documented in this manual.

- VME/VXI/QVME/QVME2 -1553-M Single-Channel multi-function 1553 module
- VME/VXI/QVME/QVME2 -1553-MM Dual-Channel Multi-function 1553 module
- VME/VXI/QVME/QVME2 -1553-MMMM Quad-Channel Multi-function 1553 module
- VME/VXI/QVME/QVME2 -1553-S Single-Channel single-function 1553 module
- VME/VXI/QVME/QVME2 -1553-SS Dual-Channel single-function 1553 module
- VME/VXI/QVME/QVME2 -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/QVME/QVME2-1553 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
000000-0000FF	Not Used
000100-001FFF	Misc. (Irig, Discretes, and Diff I/O)
000800-000FFF	CH 1 LPU Hardware Registers
001000-001FFF	CH 1 Register File
100000-1FFFFF	CH 1 RAM
200800-200FFF	CH 2 LPU Hardware Registers
201000-201FFF	CH 2 Register File
300000-3FFFFF	CH 2 RAM
400800-400FFF	CH 3 LPU Hardware Registers
401000-401FFF	CH 3 Register File

VME/VXI-1553 Byte Address Offset	Function
500000-5FFFFF	CH 3 RAM
600800-600FFF	CH 4 LPU Hardware Registers
601000-601FFF	CH 4 Register File
700000-7FFFFF	CH 4 RAM

Q104-1553, Q104-1553-P, QPCI-1553, QPCX-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, QCP-1553, PCCARD-D1553, RPCCARD-D1553, IP-D1553, RXMC-1553, RXMC2-1553, R15-LPCIe, RPCIe-1553 and R15-EC

The Q104-1553 is offered in twelve configurations, all of which are documented in this manual:

- Q104-1553-1M Single-Channel multi-function 1553 module, fixed voltage, ISA interface
- Q104-1553-2M Dual-Channel multi-function 1553 module, fixed voltage, ISA interface
- Q104-1553-4M Quad-Channel multi-function 1553 module, fixed voltage, ISA interface
- Q104-1553-1S Single-Channel single-function 1553 module, fixed voltage, ISA interface
- Q104-1553-2S Dual-Channel single-function 1553 module, fixed voltage, ISA interface
- Q104-1553-4S Quad-Channel single-function 1553 module, fixed voltage, ISA interface
- Q104-1553-1M-P Single-Channel multi-function 1553 module, fixed voltage, Universal PCI interface
- Q104-1553-2M-P Dual-Channel multi-function 1553 module, fixed voltage, Universal PCI interface
- Q104-1553-4M-P Quad-Channel multi-function 1553 module, fixed voltage, Universal PCI interface
- Q104-1553-1S-P Single-Channel single-function 1553 module, fixed voltage, Universal PCI interface
- Q104-1553-2S-P Dual-Channel single-function 1553 module, fixed voltage, Universal PCI interface
- Q104-1553-4S-P Quad-Channel single-function 1553 module, fixed voltage, Universal PCI interface

In addition, the above configurations may have an optional “-W” suffix for IRIG Receiver/Generator option. Other configurations are also available by special order.

The QPCI-1553 and QPCX-1553 are offered in six configurations, all of which are documented in this manual:

- QPCI-1553-1M / QPCX-1553-1M Single-Channel multi-function 1553 module, fixed voltage, Universal PCI interface
- QPCI-1553-2M / QPCX-1553-2M Dual-Channel multi-function 1553 module, variable voltage, Universal PCI interface
- QPCI-1553-4M / QPCX-1553-4M Quad-Channel multi-function 1553 module, variable voltage, Universal PCI interface
- QPCI-1553-1SA / QPCX-1553-1SA Single-Channel single-function 1553 module, fixed voltage, Universal PCI interface
- QPCI-1553-2SA / QPCX-1553-2SA Dual-Channel single-function 1553 module, fixed voltage, Universal PCI interface
- QPCI-1553-4SA / QPCX-1553-4SA Quad-Channel single-function 1553 module, fixed voltage, Universal PCI interface

In addition, the above configurations may have a “-W” suffix for IRIG Receiver/Generator option. All versions of the QPCX are available with or without variable voltage. Other configurations are also available by special order.

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

- QCP-1553-1M Single-Channel multi-function 1553 module, fixed voltage, Universal PCI interface
- QCP-1553-2M Dual-Channel multi-function 1553 module, variable, Universal voltage PCI interface
- QCP-1553-4M Quad-Channel multi-function 1553 module, variable voltage, Universal PCI interface
- QCP-1553-1SA Single-Channel single-function 1553 module, fixed voltage, Universal PCI interface
- QCP-1553-2SA Dual-Channel single-function 1553 module, fixed voltage, Universal PCI interface
- QCP-1553-4SA Quad-Channel single-function 1553 module, fixed voltage, Universal PCI interface

In addition, the above configurations may have a “-W” suffix for IRIG Receiver/Generator option. A variety of faceplate and I/O configurations are available. Other configurations are also available by special order.

The R15-AMC is offered in six configurations, all of which are documented in this manual. The AMC-1553 is only available as a 4-channel single function configuration.

- R15-AMC-1M Single-Channel multi-function 1553 module, fixed voltage, AMC interface
- R15-AMC-2M Dual-Channel multi-function 1553 module, fixed voltage, AMC interface
- R15-AMC-4M Quad-Channel multi-function 1553 module, fixed voltage, AMC interface
- R15-AMC-1S Single-Channel single-function 1553 module, fixed voltage, AMC interface
- R15-AMC-2S Dual-Channel single-function 1553 module, fixed voltage, AMC interface
- R15-AMC/AMC-1553-4S Quad-Channel single-function 1553 module, fixed voltage, AMC interface

In addition, the above configurations may have an optional “-W” suffix for IRIG Receiver/Transmitter. Other configurations are available by special order.

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

- QPM-1553-1M / QPMC-1553-1M Single-Channel multi-function 1553 module, fixed voltage, PMC interface
- QPM-1553-2M / QPMC-1553-2M Dual-Channel multi-function 1553 module, fixed voltage, PMC interface
- QPM-1553-4M / QPMC-1553-4M Quad-Channel multi-function 1553 module, fixed voltage, PMC interface
- QPM-1553 / QPMC-1553-1S Single-Channel single-function 1553 module, fixed voltage, PMC interface
- QPM-1553 / QPMC-1553-2S Dual-Channel single-function 1553 module, fixed voltage, PMC interface
- QPM-1553 / 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. A “-H” configuration for QPM-1553 offers eight general purpose differential transceivers in lieu of

several Avionics Discrete signals. Other configurations are available by special order.

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

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

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

- PCCARD-D1553-1M Single-Channel, multi-function, fixed voltage, 1553 module
- PCCARD-D1553-1S Single-Channel, single-function, fixed voltage, 1553 module
- PCCARD-D1553-2M Dual-Channel, multi-function, fixed voltage, 1553 module
- PCCARD-D1553-2S Dual-Channel, single-function, fixed voltage, 1553 module

In addition, the above configurations may have a “-W” suffix for IRIG Receiver/Generator option.

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

- RPCCARD-D1553-1M Single-Channel, multi-function, fixed voltage, 1553 module
- RPCCARD-D1553-1S Single-Channel, single-function, fixed voltage, 1553 module
- RPCCARD-D1553-2M Dual-Channel, multi-function, fixed voltage, 1553 module
- RPCCARD-D1553-2S Dual-Channel, single-function, fixed voltage, 1553 module

In addition, the above configurations may have a “-W” suffix for IRIG Receiver/Generator option.

The R15-EC is offered in four configurations, all of which are documented in this manual:

- R15-EC-1M Single-Channel, multi-function, fixed voltage, 1553 module
- R15-EC-1S Single-Channel, single-function, fixed voltage, 1553 module
- R15-EC-2M Dual-Channel, multi-function, fixed voltage, 1553 module
- R15-EC-2S Dual-Channel, single-function, fixed voltage, 1553 module

The RPCIe is offered in six configurations, all of which are documented in this manual.

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

The R15-LPCIe is offered in four basic configurations, all of which are documented in this manual.

- R15-LPCIe -1D Single-Channel dual-function 1553 module, variable voltage, PCIe interface
- R15-LPCIe -1M Single-Channel multi-function 1553 module, variable voltage, PCIe interface
- R15-LPCIe -2D Dual-Channel dual-function 1553 module, variable voltage, PCIe interface

- R15-LPCIe -2M Dual-Channel multi-function 1553 module, variable voltage, PCIe interface

  The card is offered with either a Low Profile or Full Height face plate as well as various options for removing the relays and hard wiring the transformer coupled outputs.

The RXMC-1553 is offered in four basic configurations, all of which are documented in this manual:

- RXMC-1553-1M Single-Channel, fixed voltage, multi-function 1553 module
- RXMC-1553-2M Dual-Channel, fixed voltage, multi-function 1553 module
- RXMC-1553-1S Single-Channel, fixed voltage, single-function 1553 module
- RXMC-1553-2S Dual-Channel, fixed voltage, single-function 1553 module

The RXMC2-1553 is offered in one basic configurations which is documented in this manual:

- RXMC2-1553-4M4 Quad-Channel multi-function, variable voltage front IO 1553 module

In addition, the above configurations may have a "-W" suffix for IRIG Receiver/Generator option.

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 if the software accesses the modules through a carrier board, then 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".

QPCI / QPCX / AMC/ QPM / QPMC / QCP /R15-EC/RPCIe/R15-LPCIe/ RXMC/ IP-D and Q104 (PCI Interface) Byte Address Offset	Q104 (ISA Interface) / PCCARD-D1553 /RPCCARD-D1553 Byte Address Offset	Function
000000-0007FF	0000-01FF	Host Interface Registers & IRIG, Avionics Discretes and Differential I/O

QPCI / QPCX / AMC/ QPM / QPMC / QCP /R15-EC/RPCIe/R15-LPCIe/ RXMC/ IP-D and Q104 (PCI Interface) Byte Address Offset	Q104 (ISA Interface) / PCCARD-D1553 /RPCCARD-D1553 Byte Address Offset	Function
000800-000FFF	0200-03FF	CH 1 LPU Hardware Registers
001000-001FFF	0400-07FF	CH 1 Register File
002000-002FFF	N/A	IP ID ROM shadow (IP-D1553 only)
003000-0FFFFF	N/A	Reserved
100000-1FFFFF	0800-0FFF (paged)	CH 1 RAM (1 Mbyte)
200000-2007FF	1000-11FF	Host Interface Registers & IRIG, Avionics Discretes and Differential I/O
200800-200FFF	1200-13FF	CH 2 LPU Hardware Registers
201000-201FFF	1400-17FF	CH 2 Register File
202000-2FFFFF	N/A	Reserved
300000-3FFFFF	1800-1FFF (paged)	CH 2 RAM (1 Mbyte)
400000-4007FF	2000-21FF	Host Interface Registers & IRIG, Avionics Discretes and Differential I/O
400800-400FFF	2200-23FF	CH 3 LPU Hardware Registers*
401000-401FFF	2400-27FF	CH 3 Register File*
402000-4FFFFF	N/A	Reserved
500000-5FFFFF	2800-2FFF (paged)	CH 3 RAM (1 Mbyte)*
600000-6007FF	3000-31FF	Host Interface Registers & IRIG, Avionics Discretes and Differential I/O
600800-600FFF	3200-33FF	CH 4 LPU Hardware Registers*
601000-601FFF	3400-37FF	CH 4 Register File*
602000-6FFFFF	N/A	Reserved
700000-7FFFFF	3800-3FFF (paged)	CH 4 RAM (1 Mbyte)*

* Applicable only to four-channel model boards .

The IP-D1553 occupies the lower 4MB block of IP memory space. As various carrier manufacturers may not provide completely decoded IP Address lines, the IP-D1553 responds to the upper 4MB memory space as the most-significant address line is a ‘don’t care.’ 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 describes the host interface registers. Chapter 3 describes the Configuration registers for the VME-1553, QVME-1553 and VXI-1553.

Host Interface Registers

Host Interface Registers are board-level registers that affect all MIL-STD-1553 channels. They can consist of the CSC Register, ISR Jam Port, 10K Configuration Data, Page Registers, Jumpers/Revision Register, and Interrupt Request Enable Register.

The Q104-1553, QCP-1553, QPCI-1553, QPCX-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, IP-D1553, PCCard-1553, R15-EC, RPCIe-1553, R15-LPCIe, RXMC-1553, RXMC2-1553, PCCARD-D1553, and RPCCARD-D1553 also contain registers for IRIG, discretes and differential I/O (See chapter 10, “IRIG, Discretes and Differential I/O”).

The Q104-1553-P, QCP-1553, QPCI-1553, QPCX-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, PCI-1553, cPCI-1553, R15-EC, RPCIe-1553, RXMC-1553, RXMC2-1553 and PMC-1553 boards also contain PCI configuration space that determines the model and revision number. See chapter 8, “PCI Configuration Space” for further details.

The following table shows the byte-offset location for each function located in host interface register space for each board model.

PCI-1553, PMC-1553, cPCI-1553 Byte Address a[24..0]	ISA-1553 Byte Address a[13..0]	PCCARD Byte Address a[13..0]	PCCARD-D / RPCCARD-D Byte Address a[13..0]	Q104-1553 (ISA) Byte Address a[13..0]	all other PCI and PCIe based boards Byte Address a[22..0]	Read / Write-capable	Function
0000000-0000001, 1000000-1000001	0000-0001, 2000-2001	0000-0001	0000-0001	0000-0001	000000-000001	Yes/Yes	CSC Register (Word 0)
0000002-0000003, 1000002-1000003	0002-0003, 2002-2003	N/A	0002-0003	N/A	000002-000003 (R15-AMC, AMC-1553, QPCX-1553, QPM-1553, RPCIe-1553, R15-LPCIe, R15-EC, RXMC-1553, RXMC2-1553)	Yes/Yes	CSC Register (Word 1)
0000004-0000005, 1000004-1000005	0004-0005, 2004-2005	N/A	0004-0005	N/A	N/A	No/Yes	FPGA Configuration Data
N/A	0008-0009, 2008-2009	0008-0009	0010-0011	0010-0011	N/A	Yes/Yes	Page Register Channel 1
N/A	000a-000b, 200a-200b	N/A	N/A	000a-000b	N/A	Yes/No	Jumpers/Rev. #
N/A	000c-000d, 200c-200d	N/A	N/A	000c-000d	N/A	Yes/Yes	IRQ Enable Reg.
N/A	000e-000f, 200e-200f	N/A	0012-0013	0012-0013	N/A	Yes/Yes	Page Register Channel 2
N/A	N/A	N/A	N/A	0014-0015	N/A	Yes/Yes	Page Register Channel 3
N/A	N/A	N/A	N/A	0016-0017	N/A	Yes/Yes	Page Register Channel 4
N/A	N/A	N/A	N/A	N/A	000012-000013	Yes/Yes	Board Options
N/A	N/A	0100-0120	0100-0120	0100-0120	0100-01ff	Yes/Yes	Discretes, IRIG, Differential triggers (Described in Chapter 10)
0000008-00fffff, 1000008-10fffff	0010-07ff, 2010-27ff	0002-0007, 0120-07ff	0014-07ff	0006-0009, 000e-000f, 0018-01ff	000200-0007ff, 200000-2007ff, 400000-4007ff, 600000-6007ff	Undefined	Undefined

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

The CSC Register is used to identify, configure and reset the board. There are several formats for this register: please refer to the

applicable section for your board. The VME model boards are documented in their own chapter.

R15-AMC, AMC-1553, QPM-1553, QPMC-1553, Q104-1553, Q104-1553-P, QPCI-1553, QPCX-1553, QCP-1553, R15-EC, RPCIe-1553, R15-LPCIe, RXMC-1553, RXMC2-1553 and IP-D1553 Model Boards

Word 0:

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

reset

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

The software reset bit is logically OR'ed with the card's System Reset input for the following model boards: R15-AMC, AMC-1553, QPM-1553, QPMC-1553, Q104-1553-P, QPCI-1553, QPCX-1553, QCP-1553, R15-EC (and QVME-1553, described elsewhere).

Enabling Hardwired RT Address (refer to Chapter 10 of this manual and also the UCA Hardware Installation Manual) disables the software reset for the R15-AMC, AMC-1553, QPM-1553, QPMC-1553 and Q104-1553 boards.

Board_type

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

1	QPM-1553 / QPMC-1553
2	IP-D1553
3	QPCI-1553 / QPCX-1553
4	Q104-1553 (ISA)
5	Q104-1553 (PCI)
6	PCCARD-D1553 / RPCCARD-D1553*

7	QCP-1553
8	AMC-1553
9	R15-EC
10	R15-AMC
11	RXMC-1553
12	RPCIe-1553
13	Not assigned
14	RXMC2-1553
15	R15-LPCIe

* Refer to PCCARD-D1553 / RPCCARD-D1553 Model Boards for PCCARD-D registers

All other values are invalid board types.

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
3	Three 1553 channels (special-order only)
4	Four 1553 channels

irig

This read-only bit determines if IRIG is enabled (1) or not (0).

custom

These read-only bits are used for Q104-1553-P and QPMC-1553 custom order options.

For Q104-1553-P, bit 13 is DAIS and if set, upon power-on if a command is received with sub-address 30, the RT responds with busy bit set and data word of 0x30. Bit 14 is EAIS and if set, upon power-on if a command is received with sub-address 32, the RT responds with busy bit set and data word of 0x32. DAIS and EAIS cannot be set simultaneously.

For Q104-1553-P, if both bit 13 and bit 14 are simultaneously set, then pre-programmed 1760 data is returned at power-on. Valid Hardwwire RT Addressing is required. The number of data words

in a transmit command will be transmitted by the RT at power on and the busy bit will be clear.

For QPMC-1553, if bits [14:13] are both set, the lpu version number is set to v3.92. The hardwired RT address is set per MIL-spec, however, the hardwired RT register read at 0x11C - 0x11d is the complement of the strapped RT Address (inverted from specification).

mode

This read-only bit determines if the board is single-mode (0) or multi-mode (1). Single-mode boards offer a bus monitor (BM) capability in addition to either multiple Remote Terminals (RT) or Bus Controller (BC) capability.

acr

A logic “1” in this read-only bit means that Word 1 of the CSC register, known as the Additional_Capabilities_Register (ACR) is present and is used.

Reserved

These read-only bits are logic zeros for the R15-AMC, AMC-1553, QPCX-1553 and QPM-1553 and are undefined for all other models.

Word 1, Additional Capabilities Register (ACR):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES	RESERVED					WP	RESERVED				BT	v_v	Res	options	

options

For V5.0 and later firmware, board options are loaded into flash memory by a FlashConfig utility available with the user application software.

For firmware prior to V5.0, each board listed in the board_type field of Word 0 of the CSC register may have up to four options. At present, options are defined only for the R15-AMC, AMC-1553, QPCX-1553, and QPM-1553. The following options apply:

00	undefined
01	no options
10	differential discretes (QPM-1553 only)
11	monitor only (special factory-order only)

For all other models, these read-only bits are undefined.

v_v

For the R15-AMC, AMC-1553, QPCX-1553, RPCIe-1553, R15-LPCIe, QVME2, and QPM-1553 boards, v_v = 1 indicates variable voltage 1553 transceivers (boards must be ordered with this option) and v_v = 0 indicates fixed voltage. This bit is read-only. For all other models, these read-only bits are undefined.

WP (Write Protect)

The Write Protect bit disables the ability to write to the flash (or tri-state the ISR ports for the Jamplayer). Use a FlashConfig utility (available with the user application software) to set the bit, write protecting the board. To clear the bit, the board must be sent to the factory for reprogramming. Only the RXMC-1553 and RXMC2-1553 boards have Write Protect capability.

I/O

These bits define the as-built I/O options available for the RXMC-1553:

(ACR bit[8:4])	DIS[4:1] OPN/GND	DIS[4:1] 28V/OPN	DIS[12:5]	PIO[8:1]	EIA-485
0					
1	X			X	
2		X		X	
3	X		X		
4		X	X		
5	X				X
6		X			X

For all other models, these read-only bits are undefined.

Reserved

These read-only bits are logic zeros for the R15-AMC, AMC-1553, QPCX-1553, RPCIe, QVME2 and QPM-1553 and are undefined for all other models.

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

The CSC Register is a 32-bit (double-word) register that resets the bus, determines board ID and configuration status, and configures the Altera 10K LPU (local processing unit) used by 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.

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 CH 1 WCS set-up complete
- su2 CH 2 WCS set-up complete
- sb1 CH 1 in standby power mode
- sb2 CH 2 in standby power mode

Byte 1: Board ID bits

- m1 CH 1 multiple mode (read only)
- m2 CH 2 multiple mode (read only)
- f1 CH 1 1553/1773 (read only)
- f2 CH 2 1553/1773 (read only)
- pr1 CH 1 present (read only)
- pr2 CH 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 channels. 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 channel 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 channel). There is a bit for each 1553 channel. The LPU for one channel can be reset if one bus is in an unknown state but the other channel 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 channel.

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 channel. **This bit is not applicable to the PCCARD-1553.**

multiple mode

These read-only bits inform the application if the 1553 channel 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.

These read-only bits inform the application if the 1553 channel 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.

For V5.0 and later firmware, clearing this bit allows for dual-mode capability: either BC/BM or RT/BM.

MIL-STD-1553/1773

These read-only bits inform the application if the channel 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**.

channel 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
11 VME-1553 [reserved]

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.

Board_Options (byte offset 0x12-0x13)

Some models do not use the Board_Options. All reserved bits should be logic zero.

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

Field	Definition
epr	enable System Reset

enable System Reset

By setting this bit, the application may enable the System Reset (e.g. PCI Reset) signal to clear all avionics logic on the board. The default is off (logic zero). The esr bit may be read back.. Boards supporting esr are as follows:

- AMC-1553
- Q104-1553P
- QcP-1553

- QPCI-1553
- QPCX-1553
- QPM-1553
- QPMC-1553
- QVME-1553
- R15-EC
- R15-AMC
- RPCIe-1553
- R15-LPCIe
- QVME2-1553
- RXMC-1553
- RXMC2-1553

10K_configuration_data (byte offset 0x04-0x05)

Configuration of the Flex 10K FPGA is only applicable to PCI-1553, cPCI-1553, PMC-1553, and ISA-1553 model boards.

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.

PCCARD-D1553 / RPCCARD-D1553 Model Boards

FPGA Configuration Control / Status (byte offset 0x00-0x02)

FPGA Configuration Control / Status applies only to PCCARD-D1553 and RPCCARD-D1553.

CSC Register (Word 0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ver			irig	mod	channel_num					board_type					rst 0

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:

6	PCCARD-D1553 / RPCCARD-D1553
0 or > 7	Invalid board type

Note: For other board types, refer to previous section for register details

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

mode

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

irig

This read-only bit determines if IRIG is enabled (1) or not (0).

ver

These three bits represent the version the host / CIS interface CPLD. Applicable only to PCCARD-D1553 / RPCCARD-D1553

CSC Register (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: FPGA Configuration control/status

cfm	FPGA configuration mode
cfs	FPGA configuration status bit (read only)
cfd	FPGA configuration done (read only)
cfr	FPGA configuration ready (read only)
sdat	Serial data for FPGA 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 FPGA 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), one byte at a time to the host interface chip. FPGA configuration must be completed before accessing the LPU registers and running the 1553 bus.

In PS configuration from the host, the host transfers data from the .rbf file to the target Cyclone FPGA. To initiate configuration in this scheme, the microprocessor must generate a low-to high transition on the nCONFIG pin and the target device must release nSTATUS. The host then writes the configuration data one byte at a time to the PCCARD-D1553 / RPCCARD-D1553 host interface. The host interface then shifts the byte out a bit at a time (lsb first) to the DATA0 pin of the Cyclone FPGA. The host interface provides a DCLK transition for each bit as it is shifted into the Cyclone FPGA The host interface generates a PCCARD bus wait while each byte is shifted out. After the last byte is written the CONF_DONE signal goes high.

The Cyclone FPGA starts initialization using the internal oscillator after all configuration data is transferred. The cfd bit (configuration done) goes high to show successful configuration and the start of initialization.

FPGA configuration mode: A low clears the FPGA configuration data. A low to high begins configuration. The bit should be held low a minimum of two microseconds prior to configuration.

FPGA configuration status: This read-only bit goes low immediately after power-up and is set high after cfm goes high. If an error occurs during LPU configuration, this bit goes low.

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

FPGA configuration ready: Each configuration bit written by the application clears the "FPGA configuration ready" bit. The hardware sets the FPGA 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. In the PCCARD-D1553 / RPCCARD-D1553, this bit is not used and will always be high.

Serial data for FPGA configuration: This is the read-back of bit 0 written to the "FPGA Configuration data" address. The PCCARD-D1553 / RPCCARD-D1553 does not read back the config data.

FPGA configuration data (byte offset 0x04-0x05)

Byte-wide configuration data applies only to PCCARD-D1553 / RPCCARD-D1553.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Und								dat7	dat6	dat5	dat4	dat3	dat2	dat1	dat0

dat[7..0] Parallel data for FPGA 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.

PCCARD-D1553 / RPCCARD-D1553 writes a byte, which is then shifted and clocked out serially by the host interface PLD.

page_register [4..1]

The Q104-1553, ISA-1553, PCCARD-D1553 / RPCCARD-D1553, and PCCARD-1553 boards use paged memory because the limited size of high memory space on ISA platforms.

ISA-1553 and PCCARD-D1553 / RPCCARD-D1553 boards may have two 1553 channels, and a separate page register is provided for the API to control channels independently of each other. For the same purpose, the Q104-1553 has up to four 1553 channels and thus four page registers.

Page_register[1] is located at byte offset 0x10-0x11 for Q104-1553 and PCCARD-D1553 / RPCCARD-D1553 boards, and at 0x08-0x9 for ISA-1553 and PCCARD-1553 boards.

Page_register[2] is located at byte offset 0x12-0x13 for Q104-1553 and PCCARD-D1553 / RPCCARD-D1553 boards, and at 0x0E-0xF for ISA-1553 boards.

Page_register[3] is located at byte offset 0x14-0x15 for Q104-1553 boards.

Page_register[4] is located at byte offset 0x16-0x17 for Q104-1553 boards.

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

page

These bits determine the page of memory to be accessed. Bits 8..0 are used to access up to 512 pages of 2K byte blocks, which

provides access to 1M byte address space. Bits 15..9 are not used and should be programmed with all zeros.

On the ISA-1553 board bits 8..0 map to RAM and to WCS address lines 19..11.

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.

The ISA memory field is also used on the Q104-1553 board.

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 Q104-1553, 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" of the corresponding 1553

channel, enables the interrupt onto the selected 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 all 1553 channels 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 channel. The interrupt may apply to any 1553 channel on the card; 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 that is already in use by another resource may cause 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/QVME/QVME2-1553 A16 Configuration Registers

This chapter applies to VME-1553, VXI-1553 and QVME-1553 model boards. The VME-1553 and VXI-1553 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 and QVME-1553 boards as well except where explicitly noted.

Note: QVME2-1553 is a direct replacement for QVME-1553 with no functional difference. Throughout this document, "QVME-1553" refers to both model QVME-1553 and QVME2-1553.

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 Abaco Systems' 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 A24/A32 space is required for a four-channel board (2MB/channel).

Model Code[11..0]

These bits identify how many 1553 channels and whether the board supports single or multi-function 1553 operation and has IRIG capability:

Model Code[11..0]	Number of Channels	Single or Multi-function	IRIG
0x150	1	single-function	no
0x151	2	single-function	no
0x153	4	single-function	no
0x15C	1	multi-function	no
0x15D	2	multi-function	no
0x15F	4	multi-function	no
0x350	1	single-function	yes
0x351	2	single-function	yes
0x353	4	single-function	yes
0x35C	1	multi-function	yes
0x35D	2	multi-function	yes
0x35F	4	multi-function	yes

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 its built-in self test. Since the VME 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) outputs 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

For the VME/VXI-1553 model boards 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/VXI-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.

Since the QVME and QVME2-1553 have configuration devices on board, FPGA configuration is not required. For backwards API compatibility this register has been retained on that board.

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

VME Interrupts

There are two classes of Interrupters in VME: Release on Acknowledge [ROAK] and Release on Register Access [RORA].

Interrupts initiated by the QVME-1553, QVME2-1553, VME-1553, and VXI-1553 boards are RORA class interrupters. The interrupt is cleared by the interrupt service routine rather than by the IACK cycle.

Prior to use, the application may program the Status / ID (sometimes referred to as Interrupt Vector) for a given channel, the value of which will be asserted on the VME bus during an Interrupt Acknowledge cycle. The unique value programmed to each channel may be used by the application to determine the source of the interrupt, allowing it to promptly service the interrupting channel without polling all channels to locate the owner of the interrupt.

To enable interrupts for one or several 1553 channels on the card, the enable for a given channel must be set in the Hardware Register for the applicable channel. An interrupt is generated when the programmed criteria for a given channel occurs. The interrupt may be cleared, either by clearing the interrupt enable for the interrupting channel, or by clearing the interrupt bit for that channel.

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								la2[7..0]							

Ia2[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								Ia3[7..0]							

Ia3[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								Ia4[7..0]							

Ia4[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 QVME-1553, QVME2-1553, and 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

HW_RT (QVME-1553 and QVME2-1553 only)

When installed, this jumper selects hardwired RT addressing for all channels present on the board. See the section, “Hardwired RT Address” for more information.

CHAPTER 4

Writeable Control Store

Writeable Control Store (WCS)

Each channel has up to 4k x 41-bit Writeable Control Store (WCS). The WCS contains the microcode that runs the hardware.

For the QPCI-1553, QPCX-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, Q104-1553, Q104-1553-P, QcP-1553, PCCard-1553, R15-EC, RPCIe-1553, R15-LPCIe, QVME2-1553, RXMC-1553, RXMC2-1553 and IP-D1553 cards, the WCS is part of the on-board FPGA configuration data that is loaded automatically when the board is powered on. This data can only be updated in the field by the appropriate JAM Player supplied by Abaco Systems.

For VME-1553 and PCCard-D1553 / RPCCARD-D1553 cards, the WCS is contained in the FPGA image that is downloaded to the board by the API during setup.

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553 model boards, the WCS is downloaded to the board from the host 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 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.

CHAPTER 5

Hardware Registers

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 0x000 through 0x7FF (byte offsets 0x000 through 0xFFF). Access the registers using the base address plus the offset. The base addresses for the different model boards are as follows:

Base Model	Page	Base Word Address	Base Byte Address
PCI-1553 (CH 1) cPCI-1553 (CH 1) PMC-1553 (CH 1)	N/A		0400000
PCI-1553 (CH 2) cPCI-1553 (CH 2)	N/A		1400000
VME/VXI/QVME/QVME2-1553 Q104-1553-P (PCI Interface option) QPCI-1553 / QPCX-1553 R15-AMC, AMC-1553 QPM-1553 / QPMC-1553 QCP-1553 IP-D1553 R15-EC RPCIe-1553 R15-LPCIe RXMC-1553 RXMC2-1553 R15-LPCIe	N/A	000400 (CH 1), 100400 (CH 2), 200400 (CH 3), 300400 (CH 4)	000800 (CH 1), 200800 (CH 2), 400800 (CH 3), 600800 (CH 4)
Q104-1553 (ISA Interface) PCCARD-D1553 RPCCARD-D1553	N/A	0100 (CH 1), 0900 (CH 2), 1100 (CH 3), 1900 (CH 4)	0200 (CH 1), 1200 (CH 2), 2200 (CH 3), 3200 (CH 4)
ISA-1553 (CH 1) PCCARD-1553 (CH 1)	0		1000
ISA-1553 (CH 2)	0		3000

In the descriptions that follow, 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 (Word 0)
0x1	0x02-0x03	W only	Time tag counter control reg
0x2	0x04-0x05	N/A	Reserved
0x3	0x06-0x07	R/W	Playback control/status reg
0x4	0x08-0x09	R/W	Response_reg
0x5	0x0a-0x0b	N/A	Reserved
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	W only	Time tag read-back register load
0xD	0x1a-0x1b	W only	BIT status
0xE	0x1c-0x1d	R/W	16-bit Minor Frame Register
0xF	0x1e-0x1f	R/W	1553_control_reg (Word 1)
0x10-0x12	0x20-0x25	W only	Tag time counter load reg
0x13-0x14	0x26-0x29	W only	Tag time counter increment reg
0x15-0x17	0x2a-0x2f	R only	Time tag read-back register
0x18	0x30-0x31	R only	WCS Version Number
0x19	0x32-0x33	R only	WCS heartbeat register
0x1a	0x34-0x35	R only	LPU Build Number
0x1b	0x36-0x37	R/W	RT address for RT Val.*
0x1c-0x1d	0x38 -0x3b	R/W	32-bit Minor Frame Register*
0x1e-0x7ff	0x3c -0xfff	N/A	Reserved

* Available as an option on certain models.

1553_control_reg (Words 0x0 and 0xf; bytes 0x00-0x01 and 0x1e-0x1f)

The host programs this register during system setup. All bits are active “1”. Reserved bits should be programmed with logic “0”. The register contains two words: Word 0 and Word 1.

Word 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	1553a	bcb	exto	bm-run	rt-run	bc-run	imw

Word 1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
res	BM_Only	sRT	32_Bit_Time	FST	AMFO	SC_Fail	uil	mic	fbct	bms	eti	BufSwapNow	EnBufSwap	fac_use	

Field	Definition
imw	invalid 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 (BC Trigger)
iw	internal wrap
int	CPU interrupt
rt31	rt 31 broadcast
sa31	sub-address 31 mode code
tb	test bus
exs	rt enable external sync (RT Trigger)
ien	cpu interrupt enable
cpl	1553 bus coupling
fac_use	reserved for factory use
BufSwapNow	Reserved for Buffer Swapping

EnBufSwap	Reserved for Enable Buffer Swapping
eti	enable trigger interrupt
bms	BC message scheduling
fbct	fixed BC message timing
mic	monitor invalid commands
uil	undefined is illegal
SC_Fail	Security Chip Fail (read only)*
AMFO	Allow message to overflow the frame
FST	frame start timing mode*
32_Bit_Time	32-bit timer mode
sRT	RT Validation mode (read only)*
BM_Only	Bus Monitor only mode (read only)*
Res	Reserved

* Available on V5.0 firmware and later.

invalid_mode_warning

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

- The BC_Run and the RT_Run bits are turned on simultaneously on a Dual mode board.
- A monitor_only board was purchased and either the BC_Run or the RT_Run bit was turned on.
- The user programs the BM_Only channel configuration option bit to be active and either the BC_Run or the RT_Run bit is turned on.
- More than one of the mutually exclusive BusTester, BM_Only or SingleRT mode bits are set at the same time. This indicates that a flash configuration problem has occurred.

When the invalid mode warning bit goes active, it disables Mil-Std-1553 operation. The operation is enabled by writing a logic "1" to the warning bit after the offending BC_Run or RT_Run bit is turned off. To turn off the warning and enable 1553 operation in the case of multiple mutually exclusive modes set, the configuration flash must be reprogrammed with no more than one mode bit.

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

bc_run, rt_run, bm_run

Setting these three bits causes board operation as a Bus controller, Remote terminal(s), and Bus Monitor, respectively. These bits may be set once the on-board RAM and associated file registers have been initialized. Multi-function boards (PCI-1553-M, etc.) support all three of these modes simultaneously. The single-function boards (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

Setting this bit forces the external TTL output to the active voltage level until this bit is cleared. Clearing this bit allows the output to be pulsed to the active level for 50 microseconds when the RT receives a sync mode command or when the Bus Monitor trigger point occurs, if either of these conditions is setup. See the section describing the *fire_ext_trigg_if_sync_mode_code* bit of the *orphaned_hardware_bits* word in chapter 6, “Register File Usage”, or the section describing *external_output_on_trigger* bit of the *bm_trigger_header* word in chapter 7, “RAM Usage”, for how to setup these capabilities.

For PCI-1553, cPCI-1553, PMC-1553, ISA-1553, VME-1553, QVME-1553, QVME2-1553, PCCARD-1553 and IP-D1553 model boards, the active voltage level is a TTL “high” voltage.

For Q104-1553, Q104-1553-P, QPCI-1553, QPCX-1553, QCP-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, RPCIe-1553 boards, the output must be routed through one of two single-ended “discrete” outputs, or through the differential RS-485/422 transmitter. The PCCARD-D1553 / RPCCARD-D1553 and R15-EC boards may route their triggers to either of the discrete outputs or to the TTL trigger out signal. (See chapter 10, “IRIG, Discretes and Differential I/O” for details on programming these output ports). The RXMC-1553 and RXMC2-1553 boards use the “External Trigger Out” registers discussed in chapter 10 to route the trigger out signals. The R15-LPCIe has dedicated trigger outputs for each 1553 channel.

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: Care should be exercised if using bc_busy while enabling low priority aperiodic messages. The bc_busy bit can be clear while processing the aperiodic messages.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 are supported if they are legal. Generally, these mode code values are 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.

bc_enable_external_sync (BC Trigger)

This bit may be read by the application to determine the state of bit 7 of word offset address 0x06 (byte offset 0x0c). Refer to the paragraph describing bc_enable_external_sync register for further details.

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 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.

Note: The internal_wrap should not be confused with the terminated onboard bus connection, which is an option for QPCI-1553, QPCX-1553, VME-1553, QVME-1553, and VXI-1553 boards (see below).

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. In addition, this bit is cleared when a board is reset and when the WCS is initialized. For PCI-1553, cPCI-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, QPCI-1553, QPCX-1553, Q104-1553-P, QCP-1553, R15-EC, RPCIe-1553, R15-LPCIe, RXMC-1553, RXMC2-1553 and PMC-1553 model boards, this bit is output from the board as the PCI INTA signal. For IP-D1553 model boards, this bit is output from the board as the active low intreq0 signal. For specifics relating to slot-location interrupt routing for the Q104-1553-P, the "MIL-STD-1553 Hardware Installation Guide".

For ISA-1553 / Q104-1553 / PCCARD-1553 / PCCARD-D1553 / RPCCARD-D1553 model boards, this bit is passed on as the enabled IRQ. See chapter 2, "Host Interface Registers" for IRQ selection. Note that many PCMCIA Card Controllers may not clearly document their interrupt routing, if provided. Consult the manufacturer's documentation for interrupts using PCMCIA devices.

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

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.

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

test_bus_enable

Test_bus_enable is applicable to the VME/VXI/QVME/QVME2-1553 and QPCI-1553 / QPCX-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.

Note: The QPCI-1553 / QPCX-1553 includes the ability to connect a transformer-coupled Line Replaceable Unit device to the LRU Port of the card for test purposes. In this mode, the external LRU connects to the terminated onboard bus for testing of the LRU without requiring external bus coupling transformers and terminators. The QPCI-1553 / QPCX-1553 includes a bus coupling transformer integrated on the card so that external couplers are not required.

Any other channel on the card may connect to the test bus as well. However, to assure error-free operation, those channels must not conflict with any RT implemented in the LRU device under test.

rt_enable_external_sync (RT Trigger)

Setting 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. After sensing an active input, the hardware sets the rt_run bit, starting the Remote Terminal.

Holding the external input signal active, or pulsing the external input signal, has no effect on the board, unless the software clears the rt_run bit. If the board is currently running as a RT, pulsing this signal has no effect.

For Q104-1553, QPCI-1553, QPCX-1553, QCP-1553, R15-AMC, AMC-1553, QPM-1553 QPMC-1553 and RPCIe-1553 boards, the input must be routed through one of two single-wire “discrete” inputs, or through the differential RS-485/422 receiver (see chapter 10, “IRIG, Discretes and Differential I/O” for details on programming these input ports).

cpu_interrupt_enable

This register bit is logically AND’ed with the interrupt bit to produce the interrupt request signal (PCI INTA, programmed VME interrupt, programmed ISA interrupt, or IP interrupt 0). The interrupt enable bit can’t be programmed or altered before the wcs_setup_complete bit is set.

It is 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 zero to this bit makes the bus direct coupled (no coupling transformer required). A logic one makes the channel transformer coupled. This affects both bus A and bus B of the dual redundant channel. At power up, the PCI-1553, QPCI-1553, QPCX-1553, VME-1553, QVME-1553, QCP-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, RPCIe-1553, R15-LPCIe and QVME2-1553 boards are transformer coupled.

Note: The Q104-1553, Q104-1553-P, PCCARD-D1553, RPCCARD-D1553, and IP-D1553 ships transformer-coupled as the standard configuration, with direct-coupled available by special order.

reserved, reserved for factory use

These bits are for factory use only and must not be accessed by the application.

enable trigger interrupt

Setting this bit enables trigger interrupts. After sensing an active high input (minimum 400 ns pulse), the hardware sets the trigger_input interrupt queue bit and asserts a hardware interrupt. Holding the external input signal active has no effect on the board until the external input signal is pulsed again. Care must be taken prior to setting this bit to ensure that the input is not left floating and is not so noisy as to cause spurious inputs and flood the interrupt buffer.

For Q104-1553, QPCI-1553, QPCX-1553, QCP-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553 and RPCIe-1553 boards, the input must be routed through one of two single-wire “discrete” inputs, or through the differential RS-485/422 receiver (see chapter 10, “IRIG, Discretes and Differential I/O” for details on programming these input ports). The R15-LPCIe has dedicated trigger inputs for each 1553 channel.

bc _message_scheduling (bms)

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

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

Setting this option bit is global; that is, it requires that every message block has values programmed for the start and repeat rate for every message. The repeat rate is the 16-bit word at word offset 5 of each bc_message_block and the start frame is the 16-bit word at word offset 15 (0xf) of each bc_message_block. There

is a one-shot mode where the repeat rate is programmed with 0x0 then the message will only go out once (determined by the start frame entry) and never again.

BC message sequencing may be controlled by the application in real-time, based on external events, however, the start frame value should always be programmed before the repeat rate or else you risk invoking a one-shot message rather than repetitive message.

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

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

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

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

fixed BC message timing

Setting the fbct bit causes the intermessage gap time to be a “message-to-message time” which is measured from the start of the current command to the start of the next command in the bus list.

The message-to-message time is programmed with each command with a range of from about 25 to 65,535 micoseconds. The message-to-message time associated with the last command in a frame is ignored.

Retries do not reset the message timer. It is the user's responsibility to ensure that the command, all retried commands and the response meet the programmed timing. If it overruns, the firmware may not be able to transmit correct bus commands until the start of the next minor frame.

If the messages overflow the minor frame, the frame predictor logic determines if there is sufficient time for the next message in the bus list. If not, subsequent message(s) are aborted, and the new frame begins at the appropriate time. There is a minor frame overflow status bit, but it’s not linked to any frame so it must be polled prior to the end of the frame following to be useful. The predictor accuracy limits the bus B/W to approximately 95%.

monitor invalid commands (mic)

Setting the monitor invalid commands (mic) bit causes posting of an interrupt when an invalid command is received. The interrupt entry would point to a BM Buffer that only has the command word, time tag and error status word programmed into it. Since the command word is invalid, other words in the BM message buffer are undetermined. It would be up to the application to detect that the invalid_word (iw) error bit is set on a command word in order to determine that the MIL-STD-1553 status and data words in the BM buffer are not valid. An invalid command must have a valid sync and two valid data bits in order to be detected.

Feature Overview:

The decoder tells the microcode when each 1553 word is input, including invalid command words. It indicates word type (command, data, status) and invalid word if inverted sync, mid-sync, mid-bit, bi-phase or parity error. By defualt, invalid commands are treated like noise and are not recorded in the interrupt queue or BM, BC or RT message buffers; so the software has no indication that an invalid command was received, except to clear the bc busy bit, if required. The *mic* bit was created so that the application has the ability to monitor the invalid commands. If *mic* is set, the command, interrupt status words and time tag are

stored for the BC and BM. The RT does not respond and takes no action.

A command word is considered invalid under the following conditions:

- mid-bit error (Invalid Manchester),
- parity error,
- non-contiguous data (late data),
- inverted sync,
- broadcast with T/R bit set, when it is not an RT-to-RT or Mode Code message,
- broadcast Mode Code that is NOT ALLOWED by MIL-STD-1553B (there are 5 mode code values),
- undefined mode codes, if the "Undefined Is Illegal" (uil) option is NOT set.

undefined is illegal (uil)

Setting the undefined is illegal (uil) bit causes undefined mode codes to be illegal (the default for undefined mode codes is invalid). The difference is that the Remote Terminal must set the message error bit if illegal.

Feature Overview:

Some T/R, sub-address, word count combinations are not defined by the MIL-STD-1553B Spec. These commands, if they appear on the bus with the sub-address field set to mode code, are called “undefined mode codes”. They should never appear on the bus but the system should handle them consistently and appropriately should they occur. The following command-field values apply:

1) T/R=0, mode codes 0 through 15

2) T/R=0, mode codes 16, 18, 19

3) T/R=1, mode codes 17, 20, 21

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

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

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

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

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

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

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

Notes:

For CS-NR-CS, data word for the last command in Step 3 is the Command Word from Step 1; for CS-NR-ME and CS-ME-ME, data word for the last command in Step 3 is the Command Word from Step 2.

The board can respond to Undefined Mode Codes per RT Validation Specification 5.2.1.1.1 (2), (3) or (4).

Undefined Broadcast Mode Codes treated as Illegal should be CS-NR-ME/BCR; however this result meets (3) of RT Validation Specification 5.2.1.1.1 g.

Abbreviations used in the table: Rx (recieve command) ; Tx (transmit command); CS (Command-Status response); NR (no response); ME (message error bit set in the status word).

SC_Fail

This is a read-only bit and is set when there is a Security Chip failure. When the SC_Fail bit is active it will disable Mil-Std-1553 operation. This bit should never be set.

AMFO (Allow message to overflow the frame)

Setting this bit prevents the Minor Frame Overflow bit from setting when the bus list overflows the frame.

FST (Frame Start Timing mode)

This mode is available on V5.0 firmware and later. Setting this bit puts the Bus Controller in Frame Start Timing mode. In this mode, the intermessage_gap_time word in the BC_message_Block is an offset in microseconds from the start of the minor frame. The elapsed time since the start of the minor frame is compared to the programmed time and will hold off the next command until equal.

32_Bit_Time

Setting this bit extends the frame time from 16 to 32 bits resolution and the intermessage gap timer from 16 to 24-bits resolution for the Bus Controller.

Refer to the paragraphs titled "16-bit Minor Frame Register" or "32-bit Minor Frame Register" for a description of the frame timing.

SingleRT (RT_Validation mode)

This mode is available on V5.0 firmware and later. This is a read-only bit and reflects the state of the channel configuration RTVal bit (refer to the Configuration Data Organization.docx). When set, the channel is a single RT, Validated per MIL-STD-1553 Appendix A, in addition to a Bus Monitor. The RT address is specified by hardware register 0x1B.

BM_Only

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

time tag counter control reg (Word 0x01; bytes 0x02-0x03) - write only

The board contains a 45-bit time tag counter (TTC) used to time stamp the data in 1553 RT and Monitor modes. The TTC may also be used as a high-resolution general-purpose timer for the host. The TTC may also be referred to as the “Tag Timer”.

The TTC is a free-running timer that is initialized during the power-on sequence. There are several capabilities to load, synchronize and to read the TTC.

The Time Tag Counter Control Register controls loading of the TTC. Three modes of operation are supported: one 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. In addition, most model boards support an IRIG loaded time tag option (see chapter 10, “IRIG, Discretes and Differential I/O”, for more details on this option). The Time Tag Counter Control Register cannot be read back by the host.

The RXMC2 supports an external clock and reset input for control of the TTC. These inputs may be enabled by setting their respective bits in this register. The external clock edge bit may be used to specify whether the rising edge (bit set) or falling edge (bit cleared) of the external clock is used to increment the couter when enabled.

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

Field	Definition
hwl	hardware load time tag counter
swl	software load time tag counter
irig (option)	load time tag counter with IRIG decoder
auto	automatically increment time tag counter
ece	external clock enable (RXMC2)
cke	external clock edge (RXMC2)
ere	external reset enable (RXMC2)

Caution: The same physical TTC is used for all RT's and the Bus Monitor on a given MIL-STD-1553 channel. Therefore, loading, clearing with the Sync mode code or incrementing the TTC will affect all RT's and the Bus Monitor simultaneously.

hardware load time tag counter

Setting this bit forces the 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. 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. The TTCL register must be loaded prior to setting this bit. 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 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 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 TTC to be loaded with the contents of the IRIG decoder with the 1pps input signal (see chapter 10, "IRIG, Discretes and Differential I/O", for more details on this option).

Note: IRIG capability is an option which must be selected when ordering your board from Abaco Systems.

auto-increment time tag counter

Setting this bit forces the TTC to be incremented by the contents of the time tag counter increment register (TTCI) with each low to high pulse on the TTL input pin. This allows for external synchronization of the time tag without software intervention. The TTCI register must be loaded prior to setting this bit.

If interrupts are desired when the load occurs, then set the hardware load tag counter bit to generate the interrupts.

Time Tag Counter Load Register (Words 0x10-0x12; bytes 0x20-0x25)

The time tag counter load register (TTCL) is a write-only 45-bit register used for loading a value into the time tag counter (TTC). 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 TTCL 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 TTCL into the TTC: a software mode, a hardware mode, and an automatic mode. These modes are described in the time tag counter control register description.

Words 0, 1 and 2:

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 (TTC) in auto-increment mode. The software first loads this register with the desired value, then, through the use of the Time Tag Control register, commands the hardware to increment the time tag counter by this value whenever an external input pulse occurs. The LSB of the TTCI is aligned with the LSB of the time tag counter load register.

Words 0 and 1:

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

TTCI [31..16]

Time Tag Read-back Register (Words 0x15-17; bytes 0x2a-0x2f) and Time Tag Read-back Register Load (Word 0x0c; bytes 0x18-0x19)

The 45-bit Time Tag Counter (TTC) may be directly read by the host through the Time Tag Read-back Register, which allows it to be used as a high-resolution general-purpose timer.

The output of the TTC feeds two 45-bit registers. The first, is loaded at the mid-parity time of every 1553 command word received on the bus, and is stored in the RT and Bus Monitor message buffers.

The second register is loaded by the host writing (any data value) to the Time Tag Read-back Register Load address and may be directly read by the host at the three Time Tag Read-back Register word addresses.

The TTC is incremented by a board-internal, free-running, one-microsecond clock (or by an external 1PPS clock if the IRIG option has been selected). For information on loading the TTC, refer to the description of the Time Tag Counter Control Register. Generally, the 1553 application will determine the loading of the TTC and the host can use the TTC value for elapsed time by computing the delta of multiple reads, referencing the timer to an external time source (PC clock, for example), if necessary. The TTC offers a very precise clock (based on an internal crystal of at least 50 ppm stability), with an accuracy only as good as the reference used and the software and physical host bus access layers can provide.

Time Tag Read-back Register Words 0, 1 and 2:

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])												

Playback Control/status Register (Word 0x03; bytes 0x06-0x07)

This register provides control and status for the Playback feature, which is available on most of Abaco Systems' MIL-STD-1553 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									abort	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 halt
abort	software mid-playback abort

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 halt

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.

software mid-playback abort

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

response_reg (Word 0x04; byte offset 0x08-0x09), reserved (Word 0x05; 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 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
00		Response time out						00		Late response					
0x0000															

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 (Word 0x06; 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 (bex) synchronizes the BC major frame to an external event. After sensing an active input, the hardware sets the bc_run bit, starting the Bus Controller. Holding the external input signal active, or pulsing the external input signal, has no effect on the board, unless the software clears the bc_run bit. If the board is currently running as a BC, pulsing this signal has no effect. You can synchronize multiple buses to within 40 microseconds using this input. This bit may be read back at the 1553_control_reg.

For Q104-1553, QPCI-1553, QPCX-1553, QCP-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, R15-LPCIe and RPCIe-1553 boards, the input must be routed through one of two single-wire “discrete” inputs, or through the differential RS-485/422 receiver (see chapter 10, “IRIG, Discretes and Differential I/O” for details on programming these input ports).

The bex bit may be read at bit 7 of the 1553_control_register located at offset address 0x00 of the hardware register space.

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

Clear Playback Status (Word 0x07; byte offset 0x0e – 0x0f)

This register is used to clear status and error bits in the playback control/status register. Writing to this location clears the hardware playback error and hardware playback buffer empty bits in the playback control/status register.

variable_volt (Word 0x08; byte offset 0x10 – 0x11)

This register determines the output voltage swing of the MIL-STD-1553B interface signals. It should be programmed prior to setting any of the run bits in 1553_control_reg. This register is write-only. For variable volt versions of the QPM-1553, QPMC-1553, Q104-1553, Q104-1553-P, VME-1553, QVME-1553, QCP-1553, PCCARD-1553, QPCI-1553, QPCX-1553, RPCIe-1553, R15-LPCIe and QVME2-1553 only the data field is valid. The other bits are don't cares. This register is not applicable to PMC-1553, PCCARD-D1553, RPCCARD-D1553, R15-EC, and RXMC-1553 model boards.

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

Field	Definition
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
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 Number (Word 0x09; byte offset 0x12 – 0x13), LPU Build Number (Word 0x1a; byte offset 0x34 – 0x35)and WCS Revision Number (Word 0x18; byte offset 0x30 – 0x31)

The LPU revision number and the WCS revision number are hard-coded 16-bit BCD numbers. The upper three digits represent a major revision, where the upper two digits are somewhat more significant than the third (the third digit might get bumped for a new model card when existing cards had only minor changes). The lowest digit represents a minor revision, The revision numbers are represented with a period between each pair of digits where the most significant digit is not printed if zero (i.e., revision 4.20 is stored as 0420).

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

The build number is a 16-bit hexadecimal number, beginning with 0x1. The build number is used for incremental fpga compiles during development and is frozen at a particular value with the version release.

For firmware versions prior to 4.11, the LPU revision number and the WCS revision number are split into a single digit major component and a two digit minor component, with the upper 4-bits optionally used for a build number.

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

For PCI-1553, ISA-1553, PMC-1553, cPCI-1553, PCCard-D1553, RPCCARD-D1553, and VME-1553 model boards, the LPU and the WCS are in software-programmable logic devices and are contained in the API. All other boards configure their onboard logic from a configuration device which contains the LPU and WCS.

interrupt_vector (Word 0x0a; 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-D1553 model boards. PCI- based IP carriers do not normally use the interrupt vector but instead use the interrupt bit in conjunction with the interrupt enable bit in the 1553 control register. The interrupt vector is cleared with a board reset.

VME-1553, QVME-1553, and QVME2-1553 cards are configured in A16 space and use the Status/ID for each channel during an Interrupt Acknowledge cycle.

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

write_interrupt_bit (Word 0x0b; byte offset 0x16-0x17)

Writing a “1” to bit 9 sets the interrupt, [PCI INTA#, IP intreq0n, VME/VXI local interrupt]; 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								

BIT_status (Word 0x0d; bytes 0x1a-0x1b)

The API drives the BIT Pass and BIT Fail LED’s after a successful or failed pass of the BIT test. A logic “1” on the BIT Pass LED turns on the green LED; a logic “0” turns it off. A logic “1” on the BIT Fail LED turns on the red LED; a logic “0” turns it off. Default for both LED's is off, and an API Exit should also turn off the LED’s. Reserved bit values are undetermined and the application should only write zeros to them.

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

Field	**Definition**
bitp	BIT Pass
bitf	BIT Fail

16-bit minor_frame_register (Word 0x0e; byte offset 0x1c-0x1d)

This 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%.

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 (1kHz). Proper operation is not guaranteed with a shorter frame time. The firmware does not allow a minor frame time of less than 800 microseconds. The API may limit the frame to a higher frame time (lower frame rate) than the firmware limit

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															

32-bit minor_frame_register (Words 0x1c-0x1d; byte offset 0x38-0x3b)

This 32-bit hardware register determines the time of each minor frame. The least significant bit represents one microsecond. Values from 40 (0x28) through 65,535 (0xFFFFFFFF) may be programmed into the register. This provides for a minor frame time of from 40 microseconds to approximately 71 minutes. A board reset loads 40,000 (1.0 second) into the register. The minor frame time should be programmed prior to setting the bc_run bit. The first minor frame begins within two or three microseconds after bc_run is set. Refer to the description on 16-bit minor_frame_register for more on Bus Controller Timing.

Warning: The minimum frame time verified is one millisecond (1kHz). If the application allows for sufficient message, response time-out, intermessage gap times and enabled retries and a small amount of "firmware overhead" time then then there should be no problems running shorter frame times; however, minor frame overflows may occur with shorter frame times.

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

WCS Heartbeat Register (Word 0x19; bytes 0x32-0x33)

A 16-bit register for each 1553 channel is increment at an undefined rate whenever the WCS (microcode) is alive. This counter may be read and displayed such that a visual inspection sees it incrementing. Failure to increment indicates that the microcode was not successfully loaded or that the board is in ill health.

RT Address for RT Validation (Word 0x1b; bytes 0x36-0x37)This register is available on V5.0 firmware and later. A 16-bit register for each 1553 channel in sRT (RT Validation) mode will determine the RT address. This register is loaded by the driver before setting the RT_Run bit. Valid addresses are 0x0 through 0x30, programmed as a five-bit binary value in the five LSB's.

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

Reserved Registers

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

CHAPTER 6

Register File Usage

The Register File consists of 128, 19-bit words 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). The base addresses for the different model boards are as follows:

Base Model	Page	Base Byte Address
PCI-1553 (CH 1), cPCI-1553 (CH 1), or PMC-1553 (CH 1)	N/A	0500000
PCI-1553, (CH 2), cPCI-1553 (CH 2)	N/A	1500000
VME/VXI/QVME/QVME2-1553, R15-AMC, AMC-1553, QPM-1553 / QPMC-1553, QPCI-1553 / QPCX-1553,, Q104-1553-P (PCI), QCP-1553, IP-D1553, R15-EC, RPCIe-1553, R15-LPCIe, RXMC-1553 (2 channel only) RXMC2-1553	N/A	001000 (channel 1), 201000 (channel 2), 401000 (channel 3), 601000 (channel 4),
Q104-1553 (ISA) (CH 1), PCCARD-D1553 (CH1), RPCCARD-D1553 (CH1)	0	0400
Q104-1553 (ISA) (CH 2), PCCARD-1553, PCCARD-D1553 (CH2), RPCCARD-D1553 (CH2)	0	1400

Base Model	Page	Base Byte Address
ISA-1553 (CH 1), Q104-1553 (ISA) (CH 3)	0	2400
ISA-1553 (ISA) (CH 2), Q104-1553 (ISA) (CH 4)	0	3400

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 Channel 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 Channel 2 are located at byte addresses 1500000 through 15000ff.

For VME/VXI/QVME/QVME2-1553 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 channel 1and for PCCARD-1553 model boards, the 128-word file registers are located at byte addresses 1400 through 14ff. Software should not access addresses 1500 through 17ff, as they are reserved for additional file registers that do not currently exist in hardware. For ISA-1553 boards the 128-word file registers for Channel 2 are located at byte addresses 3400 through 34ff.

Register File Summary (0x0 – 0xFF)

The following table summarizes the complete list of file registers. Sample initial values are provided as a reference in initializing the file registers if you are bypassing Abaco Systems' API. Actual usage is defined in the following paragraphs, but you can reference this table as a starting point when building your initialization call.

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
0	0x00-0x01	0	defrev	WCS def file revision number
1	0x02-0x03	0424	srcrev	WCS source file revision number
2 – 6	0x04-0x0D	0	N/A	Reserved
7	0x0E-0x0F	0	iswretry	BC retry interrupt status word bits
8-9	0x10-0x13	0	savedis0, savedis1	RT disable storage for override TX shutdown
A	0x14-0x15	0	user1	User register 1
B	0x16-0x17	0	user2	User register 2
C	0x18-0x19	0	N/A	Reserved
D	0x1A-0x1B	0	N/A	Local microcode temp buffer
E-F	0x1C-0x1F	0	N/A	Reserved
10	0x20-0x21	0	pbstrptr	Playback start pointer
11	0x22-0x23	0	pbendptr	Playback end pointer
12	0x24-0x25	0	pbtalptr	Playback tail pointer
13	0x26-0x27	0	pbhedptr	Playback head pointer
14	0x28-0x29	0	pbinttsh	Playback threshold word count

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
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 current word pointer for receive command of RT-RT
23	0x46-0x47	0	AEOLflag	Aperiodic End-Of-List flag
24	0x48-0x49	0	sta1save	Store status word
25	0x4A-0x4B	0	N/A	Reserved
26	0x4C-0x4D	0	bcstcmds	Broadcast command save

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
27	0x4E-0x4F	0	tempreg	Temporary microcode storage register
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	APmsgptt	Temporary Aperiodic message pointer
2D	0x5A-0x5B	0	APflag	Aperiodic message sequence active flag
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	evregptr	BM trigger event register pointer
33	0x66-0x67	0	eventreq	BM trigger event register
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

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
39	0x72-0x73	0	BCMFOptr	BC Minor Frame Overflow Pointer
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.
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	swretry	BC retry status word bits
46	0x8C-0x8D	0	Bcmsgpt1	BC message pointer used for retries and end-of-msg
47	0x8E-0x8F	0	Bcmsgpt2	BC message pointer used for retries
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

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
4B	0x96-0x97	0	rtmsgpt2	RT message initial word pointer for receiving 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
52	0xA4-0xA5	0	N/A	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 base address pointer (setup by the API)
59	0xB2-0xB3	0	intstat1	Interrupt status bits
5A	0xB4-0xB5	0	intstat2	Interrupt status bits
5B	0xB6-0xB7	0	orphans	Orphan hardware flags

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
5C - 5F	0xB8-0xBF	0	N/A	Reserved
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
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

Address Word Offset	Address Byte Offset	Sample Initial Value	File Register Name	File Register Description
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	bcflags	BC Flag register
79	0xf2-0xf3	0	bmflags	BM 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
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 (see chapter 2, Hardware Registers") for how to determine if there is bus activity.

high priority aperiodic message pointer (Word 0x2A; Bytes 0x54 - 0x55), low priority aperiodic message pointer (Word 0x2B; Bytes 0x56 - 0x57)and temporary aperiodic message pointer (Word 0x2C; Bytes 0x58 - 0x59)

A description of the high priority aperiodic message pointer (HPmsgpts) and the low priority aperiodic message pointer (LPmsgpts) may be found in the section, “Aperiodic Message Lists” in chapter 7, “RAM Usage”.

The temporary aperiodic message pointer is used by the firmware to store the regular message pointer (bcmsgpts) while the aperiodic list is being processed.

These pointers are 19-bit values with the three least significant bits truncated.

high priority aperiodic message list flag (Word 0x2E; Bytes 0x5C - 0x5D)

The high priority aperiodic message list flag is used by the microcode to begin the processing of a high priority aperiodic message list. The register *must* be cleared by software prior to running the Bus Controller. Any non-zero value is active.

low priority aperiodic message list flag (Word 0x2F; Bytes 0x5E - 0x5F)

The low priority aperiodic message list flag is used by the microcode to begin the processing of a low priority aperiodic message list. The register *must* be cleared by software prior to running the Bus Controller. Any non-zero value is active.

Aperiodic End-Of-List flag (Word 0x23; Bytes 0x46 - 0x47)

The firmware uses this flag to detect a that the end of an aperiodic list has occurred. Any non-zero value is active.

Aperiodic Message Sequence Active flag (Word 0x2D; Bytes 0x5A - 0x5B)

The firmware uses this flag for a quick check if any aperiodic list is active. It is the logical or of the AEOLflag, the HPAflag and the LPAflag.

high priority list bump normal flag (Word 0x30; Bytes 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_save (Word 0x41; Bytes 0x82 - 0x83) , bc_msg_pointer (Word 0x40; Bytes 0x80 - 0x81), bcmsgpt1 (Word 0x46; Bytes 0x8C - 0x8D) and bcmsgpt2 (Word 0x47; Bytes 0x8E - 0x8F)

The microcode requires several pointers to the BC Message Buffer in order to manage Bus Controller traffic in real-time.

First, there needs to be a message block pointer (bc_msg_ptr_save) that keeps track of which message is being processed, and a word pointer (bc_msg_ptr) which keeps track of which word in the message is being processed. The bc_msg_pointer_save will 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.

Note: The bc_msg_ptr_save 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 microcode loads the bc_msg_ptr_save register into the bc_msg_ptr register just before transmitting the command word onto the 1553 bus. Just after the command word is output, it writes over the bc_msg_ptr_save with the next_msg_pointer, word 11 in the message block. The microcode writes over the bc_msg_ptr as it sequences through the words of a message. The host *must not* write to this register during message processing.

To meet minimum inter-message gap times, the microcode uses a prediction algorithm for the next message during current message

transmission. When a retry occurs, the current message needs to be re-transmitted, so we need two additional registers tin order to store the current and the next bc_msg_ptr_save and bc_msg_ptr. The bc_msg_ptr_save register is saved in bc_msg_pt1 before the next_msg_pointer overwrites it and the bc_msg_ptr is stored in bcmsgpt2 when the commond word is received.

All four BC message pointers are 19-bit values 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
address															

bc_data_pointer (Word 0x42; Bytes 0x84 - 0x85), bc_data_data_pointer_temp (Word 0x31; Bytes 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. If the BC Scheduling Option is selected, only data_pointer_a is valid.

This pointer is a 19-bit value with the three least significant bits truncated. See chapter 7, “RAM Usage” for details.

The bc_data_pointer_temp register is used by the microcode for temporary storage.

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 (Word 0x43; Bytes 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_interrupt_status, rt_interrupt_enables, rt_ interrupt_status, bm_interrupt_enables,

and bm_interrupt_status words. (See the section, “rt_interrupt_enables” in chapter 7, “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	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
er	early response
lr	late response
ira	incorrect rt address
bus	bus A or B
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	nres 2	iw2	no cmd	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
nocmd	no command seen by decoder

iw2 combo	invalid command 2 word or CMD1/CMD2
nres2	no-response on receive command of RT-to-RT
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_buffer_ptr (Word 0x3A; Bytes 0x74 - 0x75) and bc_retry_buffer_ptr save (Word 0x58; Bytes 0xB0 – 0xB1)

These pointers are used internally by the microcode to process retry messages. The bc_retry_buffer_ptr_save must be set-up by software to point to the first word of the bc_retry_buf if retries are to be used. The microcode uses the bc_retry_buffer_ptr to keep track of which retry is being processed.

Note: The bc_retry_buffer_ptr_save must be programmed to point to the bc_retry_buf prior to turning on the bc_run bit or the board may not operate correctly if retries are enabled.

sw_retry (Word 0x45; Bytes 0x8A - 0x8B)

This register enables the BC to retry a message if the corressponding bit in the status word is set. The sw_retry register is bitwise-anded with the 16-bit status word, except that Reserved bits are masked out by the hardware. The retry is enabled if the result is non-zero. However, the hardware retriwa a message only if the retry_enable bit in the bc_message_block is also set. The intermessage gap before a BC retry message is 8 microseconds. Take care to minimize the possibility that the RT may respond late and at the same time as the BC is transmitting the retry command.

For RT-to-RT messages both transmit and receive status words EW bitwise-anded with the sw_rety register. If either result is non-zero, the retry is enabled. For a retry to occur, the retry_enable bit in the bc_message_block must also be set.

This register may be used concurrently with the isw_retry register and with the retry on no reponse bit in the bc_flags register. These retry stimuli are logically-anded, and any one of them causes a

retry if the retry_enable bit is set in the bc_message_block. The number of retries to transmit and the bus to transmit on is determined by the bc_retry_buf.

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

isw_retry (Word 0x7; Bytes 0x0E - 0x0F)

This register enables the BC to retry a message if the corressponding bit in the interrupt status word is set. The isw_retry register is bitwise-anded with the interrupt status word (file register 0xB2-0xB3), except that Reserved bits are masked out by the hardware. The retry is enabled if the result is non-zero. However, the hardware retries a message only if the retry_enable bit in the bc_message_block is also set. The intermessage gap before a BC retry message is 8 microseconds. Take care to minimize the possibility that the RT may respond late and at the same time as the BC is transmitting the retry command.

This register may be used concurrently with the sw_retry register and with the retry on no reponse bit in the bc_flags register. These retry stimuli are logically-anded, and any one of them cause a retry if the retry_enable bit is set in the bc_message_block. The number of retries to transmit and the bus to transmit on is determined by the bc_retry_buf.

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

See chapter 7, “RAM Usage”, rt_interrupt_enables for a complete description of these bits.

out_word_count (Word 0x4E; Bytes 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 (Word 0x78; Bytes 0xF0 - 0xF1)

The nrsp bit is programmed by the host software during BC setup . The host should only write zeros to all the other bits of the bc_flags register and should not write to this register while the BC is running. The other bits are used internally by microcode and must be cleared by the host software during BC initialization.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					nrsp	retry occ	Reserved			stop -flag	roc	Reserved Reserved			

Field	Definition
neom	do not process end of message for bc
roc	message status retry bit
stop-flag	hold off stopping BC until interrupt is posted
retryocc	retry occured
nrsp	retry on no response

do not process eom for BC (neom)

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

message status retry (roc, retryocc)

These bits are for microcode sequencing when a retry message has occurred. The roc bit is used while processing a command to be output. The retryocc bit is set when a retry is enabled and is used to set the retry_occured bit in the bc_interrupt_status2 word at the BC end of message routine. The application should not write to these bits nor should it care what state they are in.

stop flag (stop-flag)

This bit is used by the microcode to hold off stopping the BC until the interrupt is posted.

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. The BC retries a message after waiting a minimum of the programmed response time-out plus eight (8) microseconds.

BC Minor Frame Overflow Pointer (Word 0x39; Bytes 0x72 - 0x73)

The BCMinor Frame Overflow Pointer (BCMFOptr) is used to store the BC message block pointer (bcmsgpts) that the overflow occurred on. This gives the application additional information if this undesired event occurs.

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

bc_control_save (Word 0x7A; Bytes 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 (Word 0x7B; Bytes 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.

broadcast command save (Word 0x26; Bytes 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 (Word 0x28; Bytes 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 (Word 0x29; Bytes 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 (Word 0x4A; Bytes 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 RT-to-RT messages, this pointer is used for the transmitting RT.

This pointer is a 19-bit value with the three least significant bits truncated. See chapter 7, “RAM Usage” for details.

rt_message_pointer (Word 0x49; Bytes 0x92 - 0x93)

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

For RT-to-RT messages, this pointer is used for the transmitting RT.

This pointer is a 19-bit value with the three least significant bits truncated. See chapter 7, “RAM Usage” for details.

rt_message_pointer1 (Word 0x22; Bytes 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.

This pointer is a 19-bit value with the three least significant bits truncated. See chapter 7, “RAM Usage” for details.

rt_message_pointer2 (Word 0x4B; Bytes 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.

This pointer is a 19-bit value with the three least significant bits truncated. See chapter 7, “RAM Usage” for details.

rt_control_pointer (Word 0x48; Bytes 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 (Word 0x4F; Bytes 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 (Word 0x50; Bytes 0xA0 – 0xA1) and rt_error_inject_pointer_save2 (Word 0x51; Bytes 0xA2 - 0xA3)

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

rt_address_buffer_base_address (Word 0x54; Bytes 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 (Word 0x55; Bytes 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 (Word 0x34; Bytes 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 (0x6C – 0x6F). A logic zero enables the RT; a logic one 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

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

savedis0 (Word 0x8; Bytes 0x10 - 0x11) and savedis1 (Word 0x9; Bytes 0x12 - 0x13)

When the Remote Terminal is running, these two registers store the 32 RT disable bits upon reception of a transmitter shutdown mode code. They are restored upon reception of an override transmitter shutdown mode code. If the mode command is on bus A, then the disable bus B bits (0x6C – 0x6F) are stored; if the mode command is on bus B, then the disable bus A bits (0x68 – 0x6B) are stored. A logic zero enables the RT; a logic one disables the RT. These bits are not stored if the bus receiving the transmitter shutdown mode code is not enabled.

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.

User register 1 (Word 0xA; Bytes 0x14 - 0x15) and User register 2 (Word 0xB; Bytes 0x16 - 0x17)

These two registers may be used for user-specific purposes for Bus Monitor operation. While the Bus Monitor is running, User register 1 is stored at bm_message_buffer word offset 0x85 and User register 2 is stored at bm_message_buffer word offset 0x86 upon reception of a command word. These registers allow custom tagging of the monitored messages.

bm_message_pointer (Word 0x4D; Bytes 0x9A - 9B), bm_message_pointer_save (Word 0x4C; Bytes 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.

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 (Word 0x56; Bytes 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															

bmflags (Word 0x79; Bytes 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**
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 (Word 0x57; Bytes 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															

bm trigger event type (Word 0x24; Bytes 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_trigger_event_register (Word 0x33; Bytes 0x66 - 0x67)

This register is used by microcode for temporary storage of the event register.

Playback Registers

playback start pointer (Word 0x10; Bytes 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 (Word 0x11; Bytes 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 (Word 0x12; Bytes 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 (Word 0x13; Bytes 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 (Word 0x14; Bytes 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 (Word 0x15; Bytes 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 (Word 0x16; Bytes 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 (Word 0x1F; Bytes 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 (Word 0x4E; Bytes 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 (Word 0x7B; Bytes 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.

WCS Def File Revision Number (Word 0x0; Bytes 0x00 – 0x01) and WCS Source File Revision Number (Word 0x01; Bytes 0x02 – 0x03)

The microcode consists of an instruction definition file and source code. There is a revision number associated with each of these files, which is loaded into these two file registers once after applying power to the board. The microcode and associated revision is linked to a specific LPU Revision number, so you can ignore the microcode revision and use only the LPU Revision Register to reference which firmware is loaded in the board. The API initialization routine may clear these two registers.

The WCS revision numbers are split into a single digit major component and a two-digit minor component. In addition, a single digit build number may be programmed here during engineering development. Each component may be a value from zero through

nine decimal, although the build number may also be a hexadecimal value from 0xa through 0xf. For most model boards, the WCS is in software-programmable logic devices contained in the API.

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

command_word1 (Word 0x20; Bytes 0x40 - 0x41) and command_word2 (Word 0x21; Bytes 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 (Word 0x53; Bytes 0xA6 - 0xA7)

This register is described in the section “error_injection_word (Enhanced zero-crossing)” in chapter 7, “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															

interrupt_status1 and interrupt_status2 (Word 0x59; Bytes 0xB2 - 0xB5)

These two registers are used by the firmware to store temporary message status for the rt_interrupt_status, bc_interrupt_status and bm_interrupt_status words. See chapter 7, “RAM Usage”, rt_interrupt_enables for a complete description of these bits.

orphaned_hardware_bits (Word 0x5B; Bytes 0xB6 - 0xB7)

Miscellaneous control and status bits.

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

Field	Definition
res	reserved
rtts	rt load time tag on sync
fet	fire external trigger if sync mode code
mfo	minor frame overflow
bbserr	bc busy bit set on minor frame overflow
hpfo	high priority frame overflow

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

The system design must ensure that frames do not overflow, taking into account extra words associated with RT to RT messages, retries (up to eight retries per message), high word error injection and long programmable responses. However, if the Bus Controller firmware determines that there is insufficient time to transmit and process all the messages in a frame, it will set the minor frame overflow bit, and not process the remaining message(s) of the frame. In cases where the the timing is within tens of microseconds, the minor_frame_overflow bit is set yet the message might still go out. The minor_frame_overflow bit must be reset by the software and doesn't give any indication of which messages overflowed and when they overflowed; the Bus Monitor message buffer must be examined to determine that information.

The algorithm is as follows. The message is assumed to be the worst-case message, an RT-to-RT, so two command words, two status words and two 30-microsecond response times are included in the estimate. The number of data words are extracted from the command word. In estimating the time required, each 1553 word is weighted as 0.8125 (~ 20/25) of a tick, since a word transmission is 20 microseconds and the minor frame timer has a 25 microsecond resolution. The programmed one-microsecond resolution intermessage gap time that follows the current message is multiplied by 0.0391 (~ 1/25) of a tick. Two additional clock ticks (50 microseconds) are allowed for timing variations in firmware. The estimated time ticks are then added and compared to the time ticks remaining.

bc_busy_set_on_minor_frame_overflow

Set if the bc_busy bit is set at the start of a minor frame. This differs from the minor_frame_overflow bit (desribed above), for which the firmware predicts that all the messages in a bus list will not have sufficient time to be transmitted prior to the minor frame expiring (synchronizing frames is given priority to transmitting messages for chich there is insufficient time). The bc_busy_set_on_minor_frame_overflow detects that the firmware has not completed the previous frame, indicating that a message actually transmitted into the next frame or that the firmware is somehow out of sync with the bus list.

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.

Temporary registers (Word 0x27, 0x7E and 0x7F; Bytes 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 during MIL-STD-1553 operation.

word sequencer bits (Word 0x7C; Bytes 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 (Word 0x7D; Bytes 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 (Word 0x7E; Bytes 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_interrupt_status, rt_ interrupt_status and bm_interrupt_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
MASK001F	3D	7A-7B	001F
MASK003F	3E	7C-7D	003F
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

Register Name	Word Address	Byte Address	Hex Data
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 7

RAM Usage

There is a 1 MB physical SRAM for each 1553 channel on all 1553 products listed in this manual. 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.

For ISA-1553, Q104-1553, PCCARD-D1553, RPCCARD-D1553, 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.

RAM allocation is performed by the application. Some structures must be allocated to the first 128K-bytes of memory (see the next section, 19-Bit RAM Pointers). Refer to the BusTools/1553-API manual for the structures used by the API.

19-Bit RAM Pointers

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.

- 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/host reads)

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

The 1553 hardware may be set-up to simulate any or all RT's on a 1553 channel. The memory buffers described here and the registers described in 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_options
- rt_status_word
- rt_last_command_word
- rt_bit_word

rt_options

This word contains control options that may be programmed for chosen RT's.

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		BITo	dbcf	itf	edbc	rtmon

Field	Definition
rtmon	RT Monitor mode
edbc	enable dynamic bus control
itf	inhibit terminal flag
dbcf	dynamic bus control flag
BITo	Built-In-Test (BIT) Option
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 Abaco Systems' API.

- do not want the overhead of parsing out the messages.

There is a bit for each RT that determines if the 1553 channel on 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 bit is set by the API to allow the RT to respond to a dynamic bus control mode command with the dynamic mode code acceptance bit set in its status word. The mode code must also be legalized in the RT Control Buffer for this to occur. It is the responsibility of the API to clear the acceptance bit in the RT address buffer and to transfer control to the RT.

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. The API must clear this bit during initialization.

Built-In-Test (BIT) Option

This mode is available on V5.0 firmware and later. When this bit is set, the BIT word response is from data word 0 of the RT Message buffer. Otherwise it's from the The rt_bit_word in the rt_address_buffer.

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.

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 interrupt_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)

Buffer Word#	Enabled RT	Function
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 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_interrupt_status1	rt_interrupt_status1	rt_interrupt_status1
5	rt_interrupt_status2	rt_interrupt_status2	rt_interrupt_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

Word	BC-RT (receive)	RT-BC (transmit)	Broadcast
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 32)	rt_data_words (0 to 33)	rt_data_words (0 to 33)
10+n+1	rt_status_word*	reserved	reserved
(10+n+2) to 46	reserved	reserved	reserved
47	Reserved for API use	Reserved for API use	Reserved for API use

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.

* 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_interrupt_status, rt_interrupt_enables, rt_ interrupt_status, bm_interrupt_enables, and bm_interrupt_status words (See the section, “rt_interrupt_enables” in chapter 7, “RAM Usage”, for a complete description of these bits) .

rt_interrupt_enable1 (and interrupt_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
er	early response
lr	late response
ira	incorrect rt address
bus	bus A or B
res	reserved
wb	response on wrong bus
bce	bit count error
nig	no/short intermessage gap

rt_interrupt_enable2 (and interrupt_status2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
res	bmo	te	tb	me	nres	rtry	nres 2	iw2	no cmd	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
nocmd	no command seen by decoder
iw2	transmit command of an RT-to-RT is invalid
nres2	no-response on receive command of RT-to-RT
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

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.

Note: high_word status is not detected for a channel that is programmed in flash as a single RT (sRT mode) to meet RT Validation.

invalid_word

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

inverted_sync

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

mid_bit_error

This bit is set if a Manchester II error is detected on bits 2 through 16, or on a parity bit, or if there is a low bit count (see bit_count_error description).

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 µs to 2 µs 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_interrupt_status as well.

late_response , no_response and no_response2

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

The response time is measured from the mid-parity bit crossing of the last word from the 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_interrupt_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.

To differentiate between a no-response of the transmitting RT from the receiving RT on an RT-to-RT transmission, the no_response2 bit is used. If the transmitting RT fails to respond, the no_response bit is set and the no_response2 bit is clear. If the transmitting RT responds but the receiving RT fails to respond, both the no_response and the no_response2 bits are set. *This*

might seem counterintuitive, but was done this way for backward compatibility with existing software.

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. The command word will still determine the setting of the bus bit for a response on wrong bus.

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

Bit count errors will usually be identified by the presence of the mid-bit error bit; however, if another word immediately follows it, then the bit-count error status bit may be set and will have precedence over the mid-bit error bit. This bit is set when a sync bit is followed by at least two, but 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_interrupt_status words to their initialized states. In sRT (RT Validation) mode, the reset remote terminal and broadcast reset remote terminal mode commands will enable both Bus A and Bus B for the RT address selected for RT Validation in hardware register word 0x1B. This mode is available on V5.0 firmware and later.

self_test

This bit is set upon reception of a initiate self-test mode command.

mode_code

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

nocmd

This is a BC status bit and should be ignored for the RT function.

invalid_word_2

The iw2 bit is set if the transmit command word of an RT-to-RT message has an invalid Manchester or parity bit error or if the receive/transmit command word combo does not meet MIL-STD-1553 specifications (receive followed by transmitetc.).

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. This bit is only valid for the BC.

message_error_bit

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

trigger_begin, trigger_end, bus_monitor_overflow

These three bits are used by the API for Bus Monitor functions. The microcode clears these bits when writing message status to the RT, BC, and BM message buffers at the end of every message.

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

rt_interrupt_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 interrupt status and so some of these bits may not have meaning for the RT.

rt_time_tag

The Time Tag Counter (also called “tag timer”) is incremented by a board-internal, free-running, one-microsecond clock (or by an external 1PPS clock if the IRIG option has been selected). The 45-bit tag timer is loaded into three words of the RT message buffer at the reception of the mid-zero crossing of the parity bit of each enabled command word. The counter has one microsecond resolution, giving a maximum time of approximately nine years. The LSB of the tag timer is stored at the LSB of rt_time_tag0 word; the MSB is stored at bit 12 of rt_time_tag2 word.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rt_time_tag0															
rt_time_tag1															
0	0	0	rt_time_tag2												

rt_command_word, rt_status_word, and rt_data_words

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

to-RT commands, separate RT message buffers exist for transmitting and receiving RTs, if they are both enabled.

rt_command_word

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

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

rt_status_word

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

The capability exists 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. High and low priority lists may not be programmed simultaneously.

Aperiodic message lists may contain linked BC message blocks that could be any combination of: 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 high priority aperiodic message pointer (HPmsgpts) 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 messages for the given frame. If the high priority list isn't finished by the end of a minor frame, the remaining messages are transmitted on the subsequent frame(s) until the list has been completely transmitted. The hardware compares the time required to transmit each message with the remaining time in the frame to determine if the message is transmitted in the current frame or the next frame.

Low Priority

The low priority aperiodic message pointer (LPmsgpts) 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 or intermessage_gap_time
4	data_pointer_a	bit_mask	reserved	ignored
5	data_pointer_b or repeat_rate	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_interrupt_status1	reserved	reserved	ignored
10	bc_interrupt_status2	branch_msg_pointer	reserved	ignored
11	next_msg_pointer	next_msg_pointer	next_msg_pointer	next_msg_pointer
12	bc_time_tag0	bc_time_tag0	bc_time_tag0	bc_time_tag0
13	bc_time_tag1	bc_time_tag1	bc_time_tag1	bc_time_tag1
14	bc_time_tag2	bc_time_tag2	bc_time_tag2	bc_time_tag2
15	reserved or start_frame	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.

Note: Insertion of a conditional branch may cause an intermessage gap time of up to twenty-five microseconds to be inserted.

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 must be programmed, respectively. A NOOP may also be inserted to add to or adjust message timing. See the Timed NOOP description in a following paragraph.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rapi	tno	BC_	noh	res	rtry	res	bus	emf	dta	rtrt	min	mnf	ctl1	ctl0	noo

	p	Halt	i								t	r			p

Field	Definition
noop	NOOP message block
ctl[1..0]	message control bits
mnfr	start minor frame
mint	interrupt on this message
rtrt	RT-to-RT message format
dta	use data buffer A
emf	end minor frame
bus	select bus: 0 = bus A, 1 = bus B
rtry	retry enable
nohi	No hardware interrupt
BC_Halt	enable external trigger starts BC and clears discretes
tnop	Timed NOOP option (honor intermessage gap time)
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 or Halt BC
11	Conditional

A value of 00 is not valid unless the noop_message_block bit is active (clear).

BC Halt

This bit is only applicable when the message_control bits are "10" (Stop BC or Halt BC). If the BC_Halt bit is set then it clears the

bc_trigger, allowing for a retrigger on an external pulse. In addition, the discrete output register bits are cleared.

---

start_minor_frame

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

If generating minor frame timing from the card, the start minor frame bit *must* be set at the first message block of each minor frame. It *must* be cleared for all other message blocks in each frame. It must also be cleared for all aperiodic messages. The microcode tests this bit at the start of processing each message block.

If generating frame timing externally by starting and stopping each frame, the start minor frame bit is not used; however, it should never be set on a Stop BC message block. Externally generated frame timing is more complicated to generate and is not the recommended method.

---

interrupt_on_this_message

For BC Messages, 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. For Conditional, Stop BC, and NOOP message blocks, setting this bit will cause a bus controller control block interrupt (ctl) to be posted in the Interrupt Queue when the message block is executed.

---

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.

retry_enable

The retry_enable bit allows the application to enable retries on a message-to-message basis. The retry_enable bit must be set if a retry is to occur on selected status word bits or on bus errors. The desired status word bits must be enabled by setting bits of the sw_retry file register, and the desired message error bits must be enabled by setting bits of the isw_retry file register; except for the no_response, which is enabled by a bit in the bc_flags register. Refer to the previous chapter for details of the file registers.

Retry_occured status is presented in the bc_interrupt_status2 word.

The number and type of retries is programmed in the bc_retry_buf. The bc_retry_buf must be allocated and programmed if retries are enabled. Also, the bc_retry_buffer_ptr_save file register must be set-up by software to point to the first word of the bc_retry_buf.

For the Message-to-Message Timing mode, the gap preceding each retry is eight (8) microseconds. For the Fixed Message Timing and Frame Start Timing modes (options in 1553_control) the retry gap is determined by microcode execution time. For most messages, this will be four to five microseconds. For all timing modes, the gap following the last retry will be the intermessage gap programmed in the BC Message buffer.

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

The retry_enable bit is ignored for Conditional, Stop BC, and NOOP message blocks.

---

no_hardware_interrupt

This bit allows the message to be logged in the interrupt queue (when the interrupt_on_this_message bit is set and the appropriate bits in the bc_interrupt_enables register are set to cause an interrupt, but to not cause a hardware interrupt. The default for “no_hardware_interrupt” is a logic “0” which generates a hardware interrupt for the message if the message generates an interrupt in the queue.

Timed_NOOP (tnop)

This bit implements a Timed NOOP, used to extend the intermessage gap time between the messages preceding and following the Timed NOOP in the BC bus list.. Be sure to enter an intermessage_gap_time at word 3 of the BC_Message_Block.

Inserting NOOP commands into a buslist will not normally affect the intermessage gap timing unless the Timed NOOP enable bit is set. The Time entry will add from 0 to 65,535 microseconds between the preceding message and the next message. Timed NOOPs may be entered sequentially between messages in order to extend the gap further.

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

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

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 time between messages (intermessage gap time) is programmed into this register. Valid values are from six to 65,535 microseconds, with the exceptions shown in the following table.

Message Format	Minimum Inter-message Gap Time (guaranteed)
All non-Broadcast messages	5.0 us

Message Format	Minimum Inter-message Gap Time (guaranteed)
Broadcast Commands followed by 2 or more data words	6.0 us
Broadcast messages followed by a single data word (including mode codes)	15.0 us
Broadcast mode codes w/o a data word (except those listed below)	40.0 us
Broadcast Inhibit Terminal Flag and Override Inhibit Terminal Flag mode codes	45.0 us

This table shows guaranteed minimum times; smaller values may work in specific applications. The 16-bit register 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. 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.

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

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.

For no-response time-out, allow 12 to 14 microseconds longer after the programmed resonse time-out time before the next message is transmitted onto the 1553 bus.

If the *Fixed Message Timing* bit in the 1553_control_register is set (chapter 6), the intermessage gap time register is programmed to be a “message-to-message time” which is measured from the start of the current command to the start of the next command in the bus list. The table above should still be referenced to determine the minimum intermessage gap times that are added to the message times. It is up to the system designer to ensure that there is sufficient time to process all messages without overruning the frame.

Note: The default message timing mode for UCA uses an intermessage gap time that is not completely deterministic. The encoder uses a 2 MHz clock tick to output the

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

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.

If the BC Message Scheduling option is selected, then only buffer A is valid.

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
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_interrupt_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 interrupt status so some of these bits may not have meaning for the BC.

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

bc_ interrupt_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	no cmd	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
nocmd	no command seen by decoder
res	reserved
rtb0	rt busy bit 0 (for API use)
rtb1	rt busy bit 1 (for API use)
rtry	retry occurred
nres	no response
me	message error bit
tb	API use: trigger begin

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

nocmd

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

The logic detection mechanism for the missing command is as follows. When the 1553 encoder transmits a command word, it monitors a feedback line from the decoder that a valid sync plus two data bits have been detected. If this signal is not present by the end of the encoder's serial shifting out the command word, the nocmd status bit is set and the message is complete, with the microcode posting an interrupt in the interrupt queue, if so enabled.

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

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															

bc_time_tag

The Time Tag Counter (also called “tag timer”) is incremented by a board-internal, free-running, one-microsecond clock (or by an external 1PPS clock if the IRIG option has been selected). The 45-bit tag timer is loaded into three words of the BC message buffer at the reception of the mid-zero crossing of the parity bit of each enabled command word (the receive command for RT-to-RT transfers). The counter has one microsecond resolution, giving a maximum time of approximately nine years. The LSB of the tag timer is stored at the LSB of bc_time_tag0 word; the MSB is stored at bit 12 of bc_time_tag2 word.

The firmware also stores the time tag of the most recently recorded message in BC Control Blocks (Conditional, NOOP or STOP). This could be helpful in post data analysis or engineering troubleshooting.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
bc_time_tag0															
bc_time_tag1															
0	0	0	bc_time_tag2												

repeat_rate and start_frame

These two words are only used if the BC Message Scheduling option is selected (see Chapter 5). They represent binary count values.

test_word_address (2 words)

These two words contain a single pointer to the word to be tested for conditional message flow.

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 Software User's Manual" 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 to BC_RETRY_BUF(7) is for the eighth 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 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

Field	Definition
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 that future firmware revisions 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 88-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. A separate message status word for each 1553 word (command, status, or data) is stored in the buffer following the 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 88-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_interrupt_status1	bm_interrupt_status1	bm_interrupt_status1
4	bm_interrupt_status2	bm_interrupt_status2	bm_interrupt_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_interrupt_status_cw1	bm_interrupt_status_cw1	bm_interrupt_status_cw2
10	see note below	bm_response_time1	bm_command_word2
11	see note below	bm_status_word1	bm_interrupt_status_cw1

Word	BC-RT (receive)	RT-BC (transmit)	RT-RT
12	see note below	bm_interrupt_status_sw1	bm_response_time1
13	see note below	see note below	bm_status_word1
14	see note below	see note below	bm_interrupt_status_sw1
15-83	see note below	see note below	see note below
84	Reserved	Reserved	Reserved
85	User register 1	User register 1	User register 1
86	User register 2	User register 2	User register 2
87	Reserved	Reserved	Reserved

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

Words 10 through 78 for BC-RT are in the following sequential order:

- bm_data_words (1553 data word followed by message status word)
- bm_response_time1 (1 word)
- bm_status_word1 (1553 status word followed by message status word)

Words 13 through 78 for RT-BC are in the following sequential order:

- bm_data_words (1553 data word followed by message status word)

Words 15 through 83 for RT-RT are in the following sequential order:

- bm_data_words (1553 data word followed by message status word)
- bm_response_time2 (1 word)
- bm_status_word2 (1553 status word followed by message status word)

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_interrupt_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_ interrupt_status, bc_interrupt_enables and bc_interrupt_status words. See chapter 7, “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_interrupt_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_interrupt_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

Field	Definition
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
bus	bus A or B
res	reserved
wb	response on wrong bus
bce	bit count error
nig	no/short intermessage gap

bm_interrupt_enable2 (and bm_interrupt_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	no cmd	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
nocmd	no command seen by decoder
res	reserved
rtb0	rt busy bit 0 (for API use)
rtb1	rt busy bit 1 (for API use)
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

The Time Tag Counter (also called “tag timer”) is incremented by a board-internal, free-running, one-microsecond clock (or by an external 1PPS clock if the IRIG option has been selected). The 45-bit tag timer is loaded into three words of the BM message buffer at the reception of the mid-zero crossing of the parity bit of each enabled command word (the receive command for RT-to-RT transfers). The counter has one microsecond resolution, giving a maximum time of approximately nine years. The LSB of the tag timer is stored at the LSB of bm_time_tag0 word; the MSB is stored at bit 12 of bm_time_tag2 word.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
bm_time_tag0															
bm_time_tag1															
0	0	0	bm_time_tag2												

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. Response time is measured at the receiver of the Bus Controller, referenced from the mid-parity bit time to mid-sync time of the status word.

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_interrupt_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
Field	**Definition**
br	broadcast command received
reserved	reserved bits
sr	service request
in	instrumentation
me	message error
rt_address	RT address bits

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

Bus Monitor Triggering Description

When the bus monitor is running, it captures 1553 messages and stores them in a circular buffer in RAM. The interrupt queue data is updated by 1553 bus messages as long as the bus monitor interrupt is enabled. The bus monitor may also be set up for a start trigger, a stop trigger, or an external input as a trigger source. When triggering is not used, the interrupt is usually enabled as part of the bus monitor initialization routine.

Messages occurring both before and after the trigger are stored in the circular buffer. The host can determine the message that the trigger occurred on by a bit set in the interrupt queue (See the section, “Interrupt Queue”, later in this chapter).

With start trigger enabled and the interrupts disabled, the interrupt queue isn’t updated for 1553 bus messages until the start trigger event or events occur. A trigger event may be any selected command, status, or data word on the 1553 bus, a logical AND/OR of multiple bus events, an ARMING/ARMED bus event combination, or an external pulse.

With stop trigger enabled and the interrupts enabled, the interrupt queue is updated for 1553 bus messages until the stop trigger event or events occur. Trigger events are the same as for start trigger.

With an external input as the trigger source, the trigger is a start trigger or a stop trigger, depending on whether or not interrupts are enabled when the input pulses.

Start trigger, stop trigger, and external input as the trigger source may be set up simultaneously. If the interrupts are enabled, the stop trigger events are active. If the interrupts are disabled, the start trigger events are active. There is no way to tell from the interrupt queue if a trigger is a start trigger, a stop trigger, or whether the trigger is due to 1553 bus events or an external input pulse. Software intervention in real-time can be implemented to start, stop, or re-trigger in any desired order.

Other options provide the ability to trigger on errors and the ability to generate an external output pulse when the trigger event occurs and an event counter.

bm_trigger_buffer

The bus monitor trigger buffer consists of 43 words: a header word, two event registers, five words for each of three start events, five words for each of three stop events, and eight reserved words. This buffer must be programmed during board setup, but may be modified during real-time provided the appropriate enable bits are turned off in the bm_trigger_header.

- bm_trigger_header
- event_registers (1 start event register, 1 stop event register)
- bm_trigger_event_buffer (40 words)

Word	Description
0	bm_trigger_header
1	start_event_register
2	stop_event_register
3	start event 1 word_offset
4	start event 1 word_to_compare
5	start event 1 bit_mask
6	start event 1 initial_count
7	start event 1 count
8	start event 2 word_offset
9	start event 2 word_to_compare
10	start event 2 bit_mask
11	start event 2 initial_count
12	start event 2 count
13	start event 3 word_offset
14	start event 3 word_to_compare
15	start event 3 bit_mask
16	start event 3 initial_count
17	start event 3 count
18 - 22	reserved
23	stop event 1 word_offset

Word	Description
24	stop event 1 word_to_compare
25	stop event 1 bit_mask
26	stop event 1 initial_count
27	stop event 1 count
28	stop event 2 word_offset
29	stop event 2 word_to_compare
30	stop event 2 bit_mask
31	stop event 2 initial_count
32	stop event 2 count
33	stop event 3 word_offset
34	stop event 3 word_to_compare
35	stop event 3 bit_mask
36	stop event 3 initial_count
37	stop event 3 count
38 - 42	reserved

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								xot	res	terr	ext	res	henb	senb	inte

Field	Definition
inte	bm_interrupt_enable
senb	enable start trigger
henb	enable stop trigger
res	reserved bit
ext	enable external trigger
terr	trigger on errors
xot	external output on trigger
reserved	reserved bits

bm_interrupt_enable

When the bus monitor is capturing 1553 messages, the interrupt queue isn't updated for 1553 bus messages until bm_interrupt_enable is set. The software should program bm_interrupt_enable prior to executing the run command. If implementing start trigger, bm_interrupt_enable is cleared by software, then set by the hardware when all of the trigger events occur in the given order. If implementing stop trigger, bm_interrupt_enable is cleared by the hardware when the enable_stop_trigger bit is set and all of the events occur in the given order.

The bm_interrupt_enable bit may also be armed by an external TTL input if the enable_external_trigger bit is set. It is then inverted (start or stop) when the next message is input on the 1553 bus.

The bm_interrupt_enable bit is normally set by the software (default) when triggering isn't used.

enable_start_trigger

If the Bus Monitor is to be start triggered by a 1553 bus event, the software sets the enable_start_trigger bit, and if not already clear, it also clears the bm_interrupt_enable bit. Once the trigger event(s) occur, the firmware posts a trigger interrupt and sets the bm_interrupt_enable bit. Interrupts will continue to be posted in the interrupt queue.

enable_stop_trigger

If the Bus Monitor is to be stop triggered by a 1553 bus event, then the software sets the enable_stop_trigger bit, and if not already set, it also sets the bm_interrupt_enable bit. Once the trigger event(s) occur, the firmware posts a trigger interrupt and clears the bm_interrupt_enable bit. Interrupts are no longer posted in the interrupt queue.

enable_external_trigger

This bit provides the capability to use a trigger source hardwired externally to the board. The trigger occurs once the first 1553 message is input on the bus (any message) following the input pulse (an active high signal with a minimum 400 ns duration). The effect of the external trigger is to start interrupt posting (if bm_interrupt_enable is previously clear) or to stop interrupt posting (if bm_interrupt_enable is previously set). In either case, bm_interrupt_enable is inverted on triggering.

trigger_on_errors

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

external_output_on_trigger

Setting this bit causes a 50 microsecond active HIGH pulse on the EXT OUT port of the VME-1553, QVME-1553, QVME2-1553, PCI-1553, ISA-1553, PMC-1553, PCCard-1553, or cPCI-1553 model board when the 1553 bus trigger or external input trigger occurs. For QPCI-1553, QPCX-1553, R15-AMC, R15-EC, RPCIe-1553, AMC-1553, QPM-1553, QPMC-1553, QCP-1553, and Q104-1553 model boards the EXT OUT port may be assigned to discrete[7], discrete[8] and/or 485 channel[1] and setting the bit causes a 50 microsecond active LOW pulse on the EXT OUT port.

On the RXMC-1553 and RXMC2-1553 this output triger may be assigned to any of the available I/O's (see the "External Trigger Out" section in chapter 10, "IRIG, Discretes and Differential I/O").

On the R15-LPCIe, there are dedicated trigger outputs for each 1553 channel.

start_event_register and stop_event_register

The event register determines what sequence of events must occur and keeps track of the events when they do occur. There are two event registers, one for the start_trigger function and one

for the stop_trigger function. Software must set-up these registers prior to enabling the start or stop trigger bits.

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

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

ev1, ev2, and ev3

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

trigger_function

These four bits determine the trigger function/sequence.

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

* When the event is any 1553 data word or a 1553 status word of a receive command, it must be linked to a command word. In

such cases, the data or status word shall be event 2 and the command word shall be event 1.

bm_trigger_event_buffer

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

- word_offset
- word_to_compare
- bit_mask
- initial_count
- event_count

word_offset

This is the word offset of the particular message in the bm_message_buffer of the word to be tested for the trigger event. For offset values, see the section, “bm_message_buffer”, in chapter 7. Here are some example offsets:

Word	Offset
Command word	0x8
Status word of a transmit message	0xb
Data word 1 of a transmit command	0xd
Data word 32 of a transmit command	0x4b
Data word 1 of a receive command	0xa
Data word 32 of a receive command	0x48
Status word of a receive command with 1 data word	0xc
Status word of a receive command with 32 data words	0x4a
Data word 1 of an RT-to-RT command	0xf
Message Error (ME) bit set in any status word (this is not supported by Abaco Systems’ API)	0x4

Generally, event 1 is a command word, and event 2 may be another command word or a status or data word that is linked to the command word of event 1. Obviously, the data word needs to be linked to a specific command, although the word count and

transmit/receive bits of the command word may be masked off. However, since status words for a specific RT are not stored in the bm_message_buffer at fixed locations, as their locations depend on the transmit/receive bit, and in the case of receive commands, also on the word count, they are normally also linked to specific command words.

word_to_compare

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

bit_mask

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

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

initial_count and event_count

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

Note: Both the initial_count and the event_count must be programmed with 0x1 for the first event (command word) of a linked event: use the initial_count and event_count of the second event (data or status word) for the number of times that you want the specified event to occur before making it cause a trigger.

Both the initial_count and the event_count must be programmed with 0x1 for the armed (second) event of an arming/armed event sequence. The first event must occur as many

times as its event_count before the second event is counted. The second event must then occur once before trigger occurs.

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

1553 playback message code format

The Playback Data Buffer uses the following message code format.

Message Code Format		
Bit(s)	**Function**	**Description**
15	Bus	Valid only during xmit message codes 0 = 1553 bus A used 1 = 1553 bus 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 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 clear 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 set 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

Message Code Format		
Bit(s)	**Function**	**Description**
		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. Only the bit values of the second word are transmitted: bits after bit 16 have an undetermined value. ▪ 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)

An example of coding might start with the following words:

0x2800	Gap message code
0x0028	Gap between words of 20 microseconds (40 x 0.5 us)
0x0000	Gap between words, upper bits
0x4001	Xmit message code: 1 data word
0x0021	Command Word (RT=0, Rcv, SA=1, WC=1)
etc.	

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 word format is defined seperately for standard errors and for enhanced zero-crossing.

error_injection_word (standard errors)

The standard format is as follows.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
err_enc			ira	pres	wce	ncd	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)
ncd	non-contiguous data (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 = Mid-Parity Zero-Crossing error (BC, RT) 100 = Transmit 1553 on the Wrong Bus (RT) 101 = Not allowed (see Enhanced Zero-Crossing) 110 = Bi-phase error (BC, RT) 111 = Not allowed (see Enhanced Zero-Crossing)

error_info

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

bit_count_error

The hardware normally transmits a sync pulse followed by 16 data bits and ending with a parity bit for each word sent out on the 1553 bus. When the bit_count_error is selected, the five LSBs of the

error_info bits are loaded into a bit_count register representing the bit count in binary. The counter accepts values from zero to 31. The bit_count_error is programmed on a word-per-word basis.

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

inverted_sync

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

The six err_info bits are used to invert any combination of half-bit times (500 ns) for the six half-bit times of the sync bit. A logic low on the respective err_info bit will invert the sequential half-bit time of sync. Only the err_info bit patterns listed below have been verified.

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

111100	111011 (0x3B)
110000	110111 (0x37)
111001	111110 (0x3E)
011000	011111 (0x1F)
000111	000000 (0x0) – This inverts a command sync

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

000011	111011 (0x3B)
001111	110111 (0x37)
000110	111110 (0x3E)
100111	011111 (0x1F)

111000	000000 (0x0) – This inverts a data sync

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

parity_error

This error transmits a word with an inverted parity bit. non_contiguous_data (data gap)

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

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

word_count_error

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

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

programmable_response

This RT selection allows the application to enter a programmable response time of up to 31½ microseconds, measured from the mid-parity time of the receiver to the mid-sync time of the status word, programmed as a binary number in the error_info bits, with the LSB equal to 500 ns. An entry in the error_info of less than 4-microseconds constitutes an *early response*. An entry greater than 14-microseconds constitutes a *late response* according to the MIL-

STD-1553B Specification. Values of less than 8½ microseconds *are not guaranteed* to be attainable by the hardware.

respond_to_wrong_address

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

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.

This method of generating zero-crossing errors is only for firmware versions earlier than V5.0. V5.0 and later firmware does not support it.

bi_phase_error

A bi-phase bit error is when there is no zero-crossing for the entire bit time. This error injection inhibits a zero crossing for the selected bit. It may be programmed as a logic high or a logic low. Only logic low is supported for firmware versions earlier than V5.0.

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

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

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

clearing err_info[5] will force the signal to a logic low state throughout the bit time. err_info[5] is ignored for firmware versions earlier than V5.0.
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 a bi-phase error is injected onto one of the first two bits, as it depends on timing and the states of the first two bits. When the bit doesn't cross zero, it may stay in one state for 1½ microseconds and the decoder might try to re-establish sync. 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.

wrong_bus

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

error_injection_word (Enhanced zero-crossing)

This mode is available on V5.0 firmware and later. Setting the three MSB's to either 111 or 101 determines *Enhanced Zero-Crossing* and the word format is as follows.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	TC	1	ZC_Earl y	ZC_Offset						ZC_HalfBit					

Field	Definition
ZC_HalfBit	Affected word half-bit time of the
ZC_Offset	Offset of the transition
ZC_Early	Transition is early or late
TC	Transmitter compensated (active low)

0.5, 1.0, 1.5 or 2.0 us. Applying Enhanced Zero-Crossing to a word will offset the selected transition by the amount determined in the ZC_Offset field.

The transition may be programmed to occur early of the expected transition by setting the ZC_Early bit, or late of the expected transition by clearing the ZC_Early bit.

The six-bit ZC_Offset resolution (LSB) is 6.25 ns, which allows for up to a 400 ns offset. Note: the Q104-1553, Q104P-1553 and QcP-1553 boards use a 40 MHz clock and the LSB is ignored: the actual resolution of the five-bit field remaining is 12.5 ns.

Early offsets are determined by subtracting the desired offset amount from 80. For instance, for a 150 ns early offset (24 x 6.25 ns), program the six bits with 0x38 (56), which is 80 minus 24.

Late offsets are programmed with the desired value. For instance, for a 150 ns late offset (24 x 6.25 ns), program the six bits with 0x18 (24).

The ZC_HalfBit field determines the half-bit time of the word that the injected transition will occur. The 1553 word consists of forty half-bit times of 500 ns each (six for the Sync time, 32 for the 16 bits and two for the parity bit). These half-bit times are designated 1 through 41, where 41 is the same as 1 of the next consecutive word. You cannot inject an early transition on the start of a command sync. To move up the start of a data sync, inject the early crossing at the end of the parity bit of the previous word.

Early and late transitions may be programmed in the half-bit time indicated:

```
ZC_HalfBit = 4;  // mid-sync
ZC_HalfBit = 7;  // end-sync
ZC_HalfBit = (( BitCount * 2 ) + 6 ); // mid-bit
ZC_HalfBit = (( BitCount * 2 ) + 7 ); // end-bit
ZC_HalfBit = 40; // mid-parity
ZC_HalfBit = 41; // end-parity/start data sync
```

The first bit transmitted on the bus is BitCount = 1 up to the last bit before parity is BitCount = 16.

Only one transition per message may be injected with a zero-crossing error.

When using enhanced zero-crossing for RT validation, the affected half bit might require adjustment for differences of the MIL-STD-1553 transmitter outputs when the inputs are stable for greater than 500 ns (1.0, 1.5 or 2.0 us). Programming the TC bit to a logic zero determines *Transmitter Compensated Enhanced Zero-Crossing* which adjusts the early transition or the transition following a late

transition by moving it approximately 30 ns later so that plus or minus 150 ns will be met at the UUT. Generally, compensation will be required if testing an RT near the receiver margins of plus or minus 150 ns, Setting the TC bit to a logic one will not apply transmitter compensation.

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 message, BM message, RT message, tag timer load, BM trigger or input trigger), uses the message_pointer to gather further information using error_detect status, clears the (BC, BM, or RT) interrupt_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							ctl	res	ti	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
ti	trigger input
res	Reserved
ctl	bus controller control block interrupt

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.

remote terminal interrupt

If this bit is set, a 1553 message has been input to the RT buffer.

bus monitor interrupt

If this bit is set, a 1553 message has been input to the enabled BM buffer.

bus controller interrupt

If this bit is set, a 1553 message has been input to the BC buffer.

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”.

trigger input

If this bit is set, a trigger input on the selected I/O port has ocurred. The enabled_trigger_interrupt bit in the 1553_control_register must be set to enable this interrupt source.

bus controller control block interrupt

This bit is set when the interrupt is enabled for a NOOP, BC Conditional Branch Block or BC Stop Block has been executed. It is set regardless of whether the branch is taken for a Conditional Branch Block.

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 8

PCI Configuration Space

PCI Configuration Space

This chapter applies to PCI-1553, cPCI-1553, PMC-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, Q104-1553-P (PCI option), QCP-1553, QPCI-1553, QPCX-1553, RPCIe-1553, RXMC-1553, RXMC2-1553, R15-EC, and R15_LPCIe model boards only.

The PCI configuration space is fully described in the PCI Local Bus Specification, Revision 2.2. 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).

Vendor ID

Vendor ID is 13C6 hex is used for Abaco Systems' Santa Barbara Avionics group.

Device ID And Subsytem Device ID

Product	Device ID (hex)	Subsystem Device ID (hex)
PCI-1553, PMC-1553, cPCI-1553, QPM-1553, QPMC-1553 and Q104-1553-P	1553	1553
AMC-1553	1008	1008
QPCI-1553	1554	0000
QPCX-1553	1556	1556
QCP-1553	1555	1555

Product	Device ID (hex)	Subsystem Device ID (hex)
R15-EC	1557	1557
R15-AMC	1558	1558
RXMC-1553	1559	1559
RPCIe-1553	155A	155A
R15-LPCIe	155B	155B
RXMC2-1553	155C	155C

Note: For AMC-1553, QPM-1553 and QPMC-1553 from firmware version 3.94 through 4.00, and for Q104-1553-P firmware versions prior to 4.10, the Subsystem Device ID is 0000 hex.

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 applicable 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, unless the device was shipped with firmware that supports 66MHz. Contact the factory for 66MHz-capable configurations.
- 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 firmware only. Microcode and 1553 Local Processing Unit (LPU) firmware is identified separately.

Class Code

For all boards the Class Code is FF0000 hex.

Subsystem Vendor ID

The Subsystem Vendor ID is 13C6 hex.

Note:

For QPCI-1553 the Subsystem Vendor ID is 0000 hex..

For QPMC-1553 for firmware versions 3.94 and 3.96, the Subsystem Vendor ID is 1553 hex.

Base Address Registers

For 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. The R15-AMC, AMC-1553, QPM-1553, QPMC-1553 , Q104-1553-P (PCI models) and R15-EC (ExpressCard model) and RPCIe-1553 and R15-LPCIe (PCIe models) use an 8M-byte BAR and no I/O space. The boards resides in 32-bit address space.

For QPCI-1553 models, the PLX9030 PCI Interface chip is used, so two memory address spaces are required, and no PCI I/O space is required. In addition to the PCI configuration space used by the operating system for board configuration, two Base Address

Registers are used. The first, BAR0, occupies 128 bytes, and is used for a mapping of PLX9030 local control registers. BAR2 maps an 8MB window for the physical mapping to QPCI-1553 memory.

For QCP-1553 models, the PLX9056 PCI Interface chip is used, so two memory address spaces and one PCI I/O space are required. In addition to the PCI configuration space used by the operating system, three Base Address Registers are used. BAR0 occupies 512 bytes, and is used for mapping PLX9056 local control registers. BAR1 occupies 256 bytes of PCI I/O space, mapping the same local control registers, as required by PLX. BAR2 occupies an 8MB window for mapping QCP-1553 local address space.

For QPCX-1553 models, the PLX9056 PCI Interface chip is used, so two memory address spaces and one PCI I/O space are required. In addition to the PCI configuration space used by the operating system, three Base Address Registers are used. BAR0 occupies 512 bytes, and is used for mapping PLX9056 local control registers. BAR1 occupies 256 bytes of PCI I/O space, mapping the same local control registers, as required by PLX. BAR2 occupies an 8MB window for mapping QPCX-1553 local address space.

For Wintel PC systems, the address spaces are mapped by the BIOS and the Base Address Registers are programmed without user intervention. For VxWorks systems, the API takes care of the mapping and programming of the Base Address Registers. In all cases, 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”, 1553_control_reg (cpu_interrupt and cpu_interrupt_enable), for interrupt usage.

Interrupt Line Register

This value is system dependent. When the system determines that the PCI device requests that an interrupt be assigned to it, the system responds by writing the interrupt number to this register.

CHAPTER 9

IP-D1553 ID PROM

ID PROM

This chapter applies to IP-D1553 model boards only.

Every IP contains an ID PROM to assist in software configuration. Abaco Systems' 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

Address	Function	Value
0x1F	Hardware ECO Level	**
0x21	PLD Major	**
0x23	PLD Minor	**
0x25	PCB Revision	**
0x27-3F	Unused	0x00

* Value is 0x0C for IP-D1553-1M, 0x0D for IP-D1553-1S, 0x0E for IP-D1553-2M and 0x0F 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 10

IRIG, Discretes and Differential I/O

IRIG, Discretes, and Differential I/O Memory Space

This chapter describes the IRIG, Avionics Discretes and Differential I/O capabilities of the Q104-1553, Q104-1553-P, QPCI-1553, QPCX-1553, QCP-1553, R15-AMC, AMC-1553, QPM-1553, QPMC-1553, QVME-1553, IP-D1553, PCCARD-D1553, RPCCARD-D1553, R15-EC, RPCIe-1553, R15-LPCIe and PCCard-1553 model boards .

Memory Address Byte Offset	Memory Address Word Offset	Write Only	Register
0x100 - 0x101	0x80	W only	irig_dac
0x102 - 0x103	0x81	R/W only	irig_control
0x104 - 0x105	0x82	W only	irig_toy[15..0]
0x106 - 0x107	0x83	W only	irig_toy[31..16]
0x108 - 0x109	0x84	R/W	discrete_out [16..1]
0x10A - 0x10B	0x85	R/W	discrete_out [18..17]
0x10C - 0x10D	0x86	R/W	discrete_oe [16..1]
0x10E - 0x10F	0x87	R/W	discrete_oe[18..17]
0x110 - 0x111	0x88	R/W	discrete_control (RT Address)
0x112 - 0x113	0x89	R/W	discrete_control (Ext I/O)
0x114 - 0x115	0x8a	R/W	diff I/O control
0x116 - 0x117	0x8b	R/W	diff termination control
0x118 - 0x119	0x8c	R only	discrete_in [16..1]
0x11A - 0x11B	0x8d	R only	discrete_in [18..17]
0x11C - 0x11F	0x8e – 0x8f	R , R/W	rt_add_read
0x120 - 0x123	0x90 – 0x91	R/W	sw_rt_add
0x124 – 0x12b	0x92 – 0x95	R only	DAC Ready

Memory Address Byte Offset	Memory Address Word Offset	Write Only	Register
0x130-0x137	0x98-0x9B	R/W	External Trigger Out Select
0x138 – 0x15f	0x9C – 0xaf	Undefined	Reserved
0x160 – 0x161	0xb0	R/W	Temp Sensor Read Command
0x162 – 0x163	0xb1	W	Temp Sensor Write Command
0x164 – 0x17f	0xb2 – 0xbf	Undefined	Reserved
0x180 - 0x1bf	0xc0 – 0xdf	Undefined	System Use Only
0x1c0 – 0x1c7	0xe0 – 0xe3	R	Board Specific
0x1c8 – 0x1ff	0xe4 – 0xff	Undefined	Reserved

IRIG Signaling

The R15-AMC, AMC-1553, QPM-1553, QPMC-1553, QVME-1553, QVXI-1553X, QPCI-1553, QPCX-1553, QCP-1553, Q104-1553, Q104-1553-P, PCCARD-D1553, RPCCARD-D1553, R15-EC, RPCIe-1553, R15-LPCIe and IP-D1553 have optional IRIG-B capability, while the PCCard-1553 always includes IRIG-B capability. All models are capable of decoding AM or DC level-shifted IRIG, with the exception of the PCCard-1553 which receives only DC level-shifted IRIG. IRIG time may be received on the signals IRIG IN and IRIG IN Return. The following IRIG formats are accepted:

Format	Modulation Frequency	Frequency/ Resolution	Coded Expressions
B	0, 1	0, 2	0, 1, 2, 3

All boards are also equipped with an IRIG output , which outputs DC level-shifted IRIG. When enabled by the software, IRIG OUT transmits IRIG B002 data from the onboard IRIG encoder. IRIG OUT signal can source/sink 16mA at valid TTL levels.

The PCCARD-1553 supports only DC IRIG timecode reception and TTL DC IRIG output. Since there is no dedicated IRIG driver on the card, drive strength of the output is less than that of other cards.

irig_dac (Word 0x80, Bytes 0x100 – 0x101)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused								Irig_dac_data[7:0]							

The irig_dac sets the threshold level for detecting either AM or DC level shifted IRIG data. The threshold is 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 9B hex (155 decimal). This register is valid for all boards with the exception of the PCCARD-1553.

The PCCard-1553 receives only DC Level Shifted IRIG. The IRIG calibration routine is unnecessary for this card.

irig_control register (Word 0x81, Bytes 0x102 – 0x103)

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 is receiving data based on the threshold set with the Irig_dac. This bit is cleared when the read is completed. This value is always set for the PCCARD-1553, as it receives only DC level shifted IRIG.

This bit is used in conjunction with the IRIG_DAC to calibrate the internal threshold of the Amplitude Modulation signal-conditioning circuitry. By scanning the threshold range for upper and lower bounds of valid signal, the threshold may be set for approximately 80% of the valid signal range for maximum reliability.

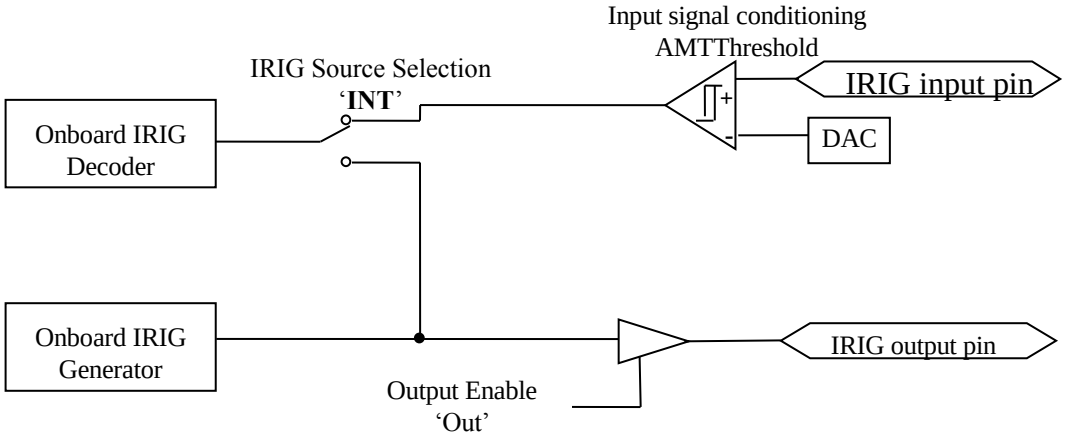
out Read/Write. This bit is the output enable for the IRIG output. When set, the internal IRIG encoder outputs valid IRIG data based on the TOY data. When cleared, the output is driven low.

For PCCARD-1553 boards, setting this bit routes IRIG signals through the card's External In and External Out signal pins, respectively. Care must be exercised when using IRIG or Trig I/O, as the pins are dual-purpose. Power-up default is to Trig I/O. This bit is known as irig_status[1] in the firmware.

int Read/Write. This bit determines the IRIG source in use. When set, the IRIG source is the onboard IRIG generator. When INT is set, the External IRIG Input is ignored. When this bit is clear, the External IRIG Input is routed to the onboard IRIG Decoder. This bit is known as irig_status[2] in the firmware.

val Read only. Set if IRIG time code input is valid. Otherwise, clear. Note that it may take over 1 second for the IRIG decoder to update the Valid flag after connection and calibration of a valid IRIG signal.

IRIG Receiver / Generator Implementation



'IRIG Status' Register Bit Function Definitions

OUT=1 for IRIG output enabled	INT=0 for External IRIG Input
OUT=0 for IRIG output driven low	INT=1 for internal IRIG loopback

IRIG Time of Year (TOY) (Words 0x82 – 0x83, Bytes 0x104 – 0x107)

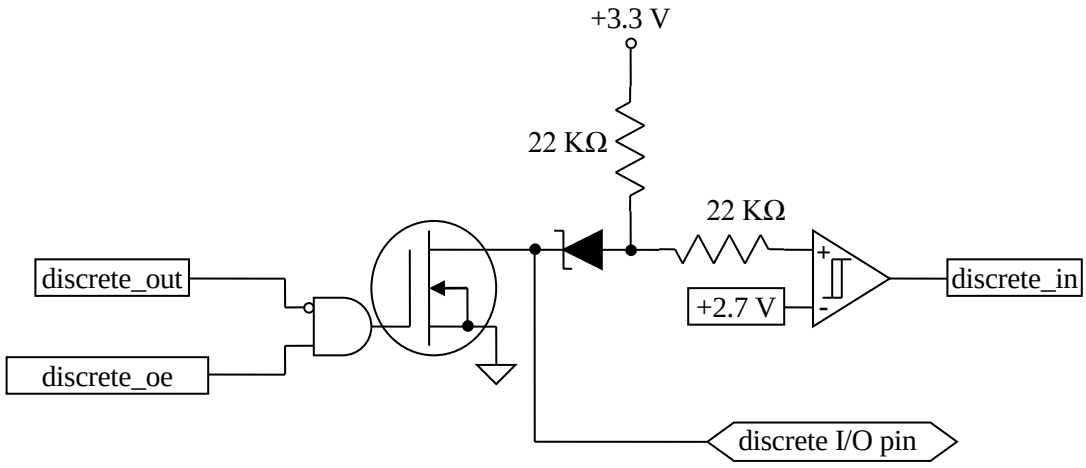
29:28	27:24	23:20	19:18	17:14	13:11	10:7	6:4	3:0
100s Of Days	Tens Of Days	Units Of Days	Tens Of Hours	Units Of Hours	Tens Of Minutes	Units Of Minutes	Tens Of Seconds	Units Of Seconds

The IRIG Time of Year is presented as 30 bits of information with the fields shown above. All fields reset to 0x0. 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.

Avionics Discrete Signals

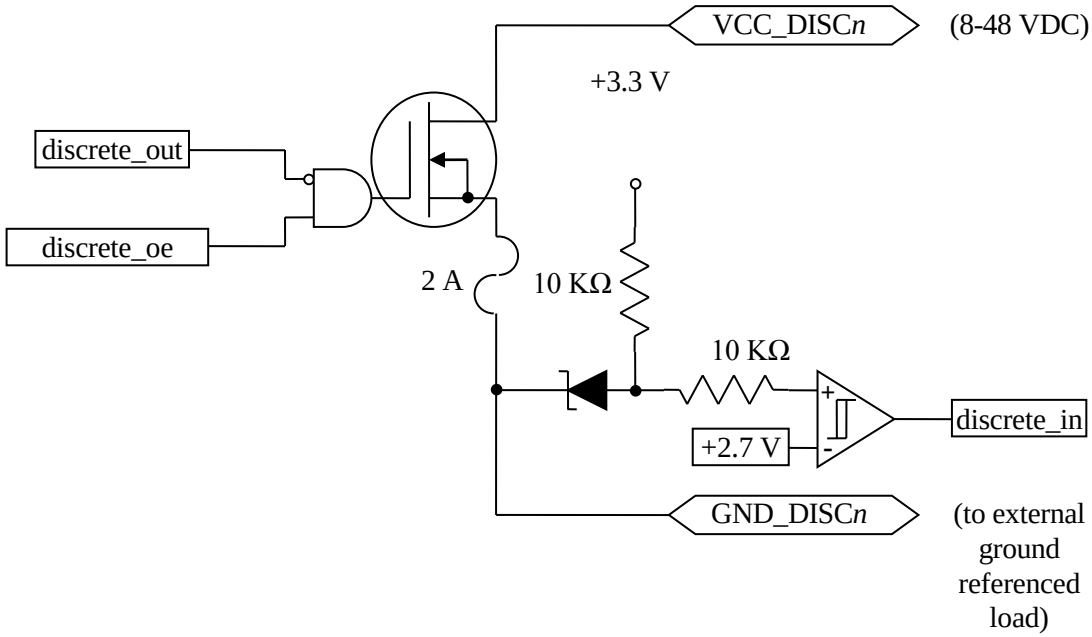
Open/Ground Discrete Outputs

The R15-AMC, AMC-1553, QPM-1553, QPMC-1553, QVME-1553, QVXI-1553X, QPCI-1553, QPCX-1553, QCP-1553, Q104-1553, Q104-1553-P, R15-EC, RVME-1553, RPCIe-1553, R15-LPCIe, RXMC-1553, RXMC2-1553, PCCARD-D1553, and RPCCARD-D1553 have *up to* 18 bi-directional individually configurable open/ground discrete pins used for general I/O, hardwired RT addressing or external triggering. The open/ground discrete outputs are low side n-channel FET switches capable of sinking 1 Amp through an external high side load, while the inputs are single ended, protected (50V max), with a logic threshold of approximately 2.7 V.



28V/Open Discrete Outputs

The RXMC can have discretes 1-4 configured as 28V/open discretes as an option, available for use as general I/O or external output triggering. The 28V/open discrete outputs are high side n-channel FET switches capable of sinking 1 Amp through an external ground referenced low side load, while the inputs are single ended, protected (50V max), with a logic threshold of approximately 2.7 V. The VCC_DISC pins should have 8 to 48 VDC applied while the external ground referenced load should be connected to the GND_DISC pin. The FET and PWB are protected by a factory replaceable 2A fuse.



Both open/ground and 28V/open discretes have the following truth-table functionality when used as outputs:

discrete_oe	discrete_out	discrete I/O pin
0	X	FET OFF
1	0	FET ON
1	1	FET OFF

While configured as outputs (discrete_oe = 1) discrete_in can still be read, although the behaviour will differ between open/ground and 28V/open discretes as follows:

With open/ground discretes when the FET is on reading discrete_in should return a “0”. When the FET is off, reading discrete_in will generally return a “1” (because of the weak 22 KΩ pullup resistor) but the load attached to the discrete I/O pin must also be taken into consideration. Note that some designs have 10K resistors verses the 22K shown.

For 28V/open discretes, when the FET is on, discrete_in should return a “1”. When the FET is off and a sufficient external load is applied, discrete_in returns a “0”.

The discrete I/O for both open/ground and 28V/open have the following truth-table functionality when used as inputs:

discrete_oe	discrete I/O pin	discrete_in

0	> 2.7 VDC	1
0	< 2.7 VDC	0

The following table shows discrete usage on each specific board:

Product	Number of Discretes available	Notes
R15-AMC, AMC-1553, QPM-1553, QPMC-1553, QCP-1553 RPCIe	18 -- discrete[18..1]	discrete[6..1] shared with CH1 terminal addressing discrete[8..7] shared with external triggering discrete[14..9] shared with Ch 2 terminal addressing
QPM-1553 (-H option), QPCI-1553, QPCX-1553, Q104-1553, Q104-1553-P	10 -- discrete[10..1]	discrete[6..1] shared with CH1 terminal addressing discrete[8..7] shared with external triggering
QVME-1553	4 -- discrete[4..1]	dedicated discretes, (the QVME has dedicated terminal addressing and external triggering I/O for each channel)
PCCARD-D1553 RPCCARD-D1553 R15-EC	2 -- discrete[8..7]	discrete[8..7] shared with external triggering.
RXMC-1553	0, 4 or 12	discrete[4..1] may be ordered as either open/ground or 28V/open discrete[12..5} are always 28V/open any discretes maybe used as an output trigger (the RXMC has dedicated terminal addressing and external triggering input for each channel)
RXMC2-1553	12 discretes	Dedicated discretes, any discretes maybe used as an output trigger (the RXMC has dedicated terminal addressing and external triggering input for each channel)
R15-LPCIe	14 discretes	Discretes 1-12 are single ended and discretes 13 & 14 are differential but can be used as single ended since the negative inputs are biased at 1.8V.

Avionics discretes may be used for general purpose use, for triggers, or for hardwired RT Addresses. In addition, one or two RS-485 transceivers are present on many board models and can be used either as differential I/O pairs or as low-side switches combined with an input comparator. The following paragraphs describe all of these functions in more detail.

General Purpose Avionics Discretes

The board has three registers to control and access the general purpose Avionics discretes: discrete_oe, discrete_out and discrete_in. These registers are wide enough to support the number of discretes available, as indicated in the above table, although only word accesses are supported in the architecture. Reading of reserved bits returns unknown values, and it is recommended to pad them with logic zeros when writing to the word, to ensure software compatibility across Abaco Systems model boards listed in this manual.

discrete_out (Words 0x84 – 0x85, Bytes 0x108 – 0x10b)

Two words are allocated for the discrete outputs:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
discrete_out[16..1]															
reserved														discrete_out [18..17]	

Field **Definition**

discrete_out [18..1] Avionics discrete output

discrete_out[18..1]

When a discrete_out bit is set the discrete FET switch is turned off. When cleared the FET switch is turned on. If the corresponding discrete_oe bit is not set, the discrete FET switch is always off.

Note: Discrete outputs may also be used for external trigger or external clock functions as described later in this chapter.

discrete_oe (Words 0x86 – 0x87, Bytes 0x10C – 0x10F)

Two words are allocated for the discrete output enable bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
discrete_oe[16..1]															
reserved														discrete_oe [18..17]	

Field **Definition**

discrete_oe [18..1] discrete output enable (data direction)

discrete_oe[18..1]

Setting the discrete_oe bit drives the corresponding discrete_out FET switch. Clearing the bit (power up default) prevents the corresponding discrete from turning on its FET switch. If a discrete is to be used as an input, the corresponding discrete_oe bit must be clear.

Hardwired RT Address

The boards listed in the table support hardwired RT addressing. When enabled hardwired RT addressing allows the RT to respond within 150 milliseconds of power-up with the Busy Bit set to messages addressed to it, as described in MIL-STD-1760D. Hardwired RT addressing is enabled differently on various boards. The following table summarizes how it is enabled for all boards that support hardwired RT addressing and which channels are supported. Refer to the MIL-STD-1553 hardware installation guide for the I/O pin locations:

1553 Board	Hardwired RT Enable Method	Inputs	Channels Supported	Modes Supported
QPCI-1553 QPCX-1553	shorting block installed at JP1	Shared discretes	Channel 1	-
Q104-1553 Q104-1553-P	shorting block installed at JB2	Shared discretes	Channel 1	-
QCP-1553	shorting block installed at JB2 *-or-*	Shared discretes	Channel 1 Channel 2	-

1553 Board	Hardwired RT Enable Method	Inputs	Channels Supported	Modes Supported
	HW_RT pin grounded on I/O connector			
AMC-1553 QPM-1553 QPMC-1553 RPCIe-1553	~HWRT_EN pin grounded on I/O connector	Shared discretes	Channel 1 Channel 2	-
QVME-1553 QVME2-1553	shorting block installed at JB1, location HW_RT.	Dedicated inputs	All	-
RXMC	OFFSET flash based bit “use_rtan” set to 0 in the channel OFFSET register	Dedicated inputs	All	Fixed, Offest
RXMC2	OFFSET flash based bit “use_rtan” set to 0 in the channel OFFSET register	No inputs, flash based RT addressing only	All	-
R15-LPCIe	OFFSET flash based bit “use_rtan” set to 0 in the channel OFFSET register	Dedicated inputs	All	Fixed, Offest

Certain boards also can be ordered with a factory installed RT Enable option, consult sales regarding this option.

Note: The RXMC has individual hardware RT enable bits for each channel in flash that may be programmed by the user, all other boards use a single jumper/pin to enable hardwired RT addressing on all capable channels.

When Hardwired RT for a board is enabled as described above and correct parity is present, the board responds to any message to the Hardwired RT address with a “busy’ response until you start the RT function running.

Hardwired RT Addressing input polarity is Active High. Onboard pull-ups assure that floating inputs are read as logic one, and grounded inputs are read as logic zero. Parity evaluation is odd;

that is, the total of inputs that read logic one [parity inclusive] must be an odd number, or the card does not respond to any bus traffic.

Boards with the method listed as “Shared discretes” use six Avionics discrete I/O lines for each channel of Hardwired RT Addressing. With these boards the following Avionics discrete pin to hardware RT address and parity bits assignment applies (refer to the MIL-STD-1553 Hardware Installation Guide for board Avionics discrete pin locations):

1553 Channel	Parity Bit	Address Bit 4	Address Bit 3	Address Bit 2	Address Bit 1	Address Bit 0
1	ADISC6	ADISC5	ADISC4	ADISC3	ADISC2	ADISC1
2	ADISC14	ADISC13	ADISC12	ADISC11	ADISC10	ADISC9

Boards with the method listed as “Deidicated inputs” have inputs solely used for hardware RT address and parity lines.

During initialization, the API checks the discrete control register (0x110) for the [rta4..rta1] bits to be set, indicating that the board is using Hardwired RT addressing. If these bits are set, the API checks the corresponding parity error bit in the discrete control register for parity errors. If there is a parity error, the API initialization returns with an error. Otherwise, the API reads and stores encoded RT address.

discrete_control register – Hardwired RT Address (Word 0x88, Bytes 0x110 – 0x111)

This register controls hardwired RT Addressing.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved				cb4	pe4	rta4	cb3	pe3	rta3	cb2	pe2	rta2	cb1	pe1	rta1

Field	Definition
rta[2..1]	use hardwired rt address (for channels 1 - 2)
pe[2..1]	hardwired rt address parity error (for channels 1 - 2)
cb[2..1]	clear rt busy bit (for channels 1 - 2)
rta[4..3]	use hardwired rt address (for channels 3 - 4)
pe[4..3]	address parity error (for channels 3 - 4)

cb[4..3] clear rt busy bit (for channels 3 - 4)

Caution: discrete_control register assignments must be made *BEFORE* setting any BC, RT, or BM run bits to ensure proper bus triggering behavior.

use_hardwired_rt_address

The use_hardwired_rt_address bit (rta) is a read only bit that is set by installing an on board jumper (QPCI, QPCX, Q104, Q104-P, QCP, and QVME-1553), by tying an off board input signal to GND (QPMC) or by having valid address and parity present at board configuration (RXMC-1553, RXMC2-1553 and R15-LPCIe). For all products, the jumper, input signal or valid address and parity enables all of the rta bits for that product. The following table summarizes the source and functionality for each product with terminal addressing capability:

Product	terminal address source			
	Ch 1	Ch 2	Ch 3	Ch 4
QPM-1553 (-H opt.), QPCI-1553, QPCX-1553, Q104-1553, Q104-1553-P	discrete[6..1]	n/a	n/a	n/a
R15-AMC, AMC-1553, QPM-1553, QPMC-1553, QCP-1553	discrete[6..1]	discrete[14..9]	n/a	n/a
QVME-1553	P2 connector	P2 connector	P2 connector	P2 connector
RXMC-1553	Front/Rear I/O Connector	Front/Rear I/O Connector	n/a	n/a
RXMC2-1553	flash	flash	flash	flash
R15-LPCIe	Front I/O Connector or flash	flash	n/a	n/a

If this bit is set and the address evaluation with parity results in an error (i.e. parity is “even” instead of being “odd”) the initialization returns an error and the problem must be corrected. If there is no

parity error, the software may still overwrite the external RT address to a different address with the sw_rt_add registers, but it cannot drive these six lines. Hardwired RT address is obtained in under 150 milliseconds in order to allow for RT Busy Bit response, as described in MIL-STD-1760D. Standard Abaco Systems' boards listed in this manual power up in transformer-coupled mode.

hardwired_rt_address_parity_error

This bit is read only. If the use_hardwired_rt_address bit is active, the API may interrogate this bit for valid parity. If parity is not valid (i.e., this bit is "1"), the external RT address strap is ignored and the six discretes may be used for other input functions. This bit reads back as "0" if the use_hardwired_rt_address bit is "0".

The Hardwired RT Address and Parity input polarity is high true. A floating [disconnected] input reads back Logic One because of on-card pull-up resistors, while a grounded input reads back Logic Zero. Parity evaluation is odd; that is, the total of inputs that read Logic One [parity inclusive] must be an odd number, else the Parity Error is set.

clear_rt_busy_bit

The busy bit in the 1553 status word is set on power up to meet the response timing of MIL-STD-1760D. The software needs to clear the busy bit when it has completed initialization by writing a "1" to the clear_rt_busy_bit. This bit is read/write.

Triggers

The R15-AMC, AMC-1553, QPM-1553, QPMC-1553, QPCI-1553, QPCX-1553, QCP-1553, PCCARD-D1553, RPCCARD-D1553, R15-EC, RPCIe-1553, Q104-1553-P, and Q104-1553 cards have two discrete lines (discrete 7 and discrete 8) that are programmable for either input or output triggers: TRIGIN and TRIGOUT. You may also connect an RS-485/422 differential I/O signal to the differential trigger ports on these boards for use as differential triggers.

Some boards have dedicated TRIGIN or EXTTRIG signals for an external trigger input. These are a TTL signal that is activated by a low to high transition. The board determines the minimum high

pulse width required for detection. See the applicable hardware reference chapter for specific input duration requirements. The input trigger has one of four functions that are programmed in software: BC trigger, RT trigger, BM trigger, and Load Tag Timer.

- **BC Trigger:** After sensing an active input, the hardware sets the bc_run bit, starting the Bus Controller. Holding the external input signal active, or pulsing the external input signal has no effect on the board, unless the software clears the bc_run bit. If the board is currently running as a BC, pulsing this signal has no effect. You can synchronize multiple buses to within 40 microseconds using this input.
- **RT Trigger:** RT trigger is essentially the same as the BC, but activates all RTs that are previously set up by the application when bit 13 of 1553_control_register1 is set.
- **BM Trigger:** After sensing the input to be active, the hardware starts capturing 1553 data, if the software sets the enable_external_trigger bit in the BM trigger header word.
- **Load Tag Timer:** After sensing the input active, the tag timer is loaded with a pre-programmed value in the Time Tag Counter Load register (TTCL) if bit 6 of 1553_control_register1 is set. A single tag timer is used for the Bus Monitor and all Remote Terminals.

TRIGOUT is an active high TTL signal that goes active for 50 microseconds. This output is under Software control. It may be enabled for BM trigger or RT Synchronize Mode Code.

BM Trigger Output: The BM trigger output fires when the BM trigger occurs. You must enable the external_output_on_trigger flag stored in the bm_trigger_buffer in RAM.

RT Synchronize Mode Code Output: This signal fires when a simulated RT receives a Synchronize Mode Code, providing you set the fire external trigger (FET) bit of the orphaned_hardware_bits register. For a Broadcast Synchronize Mode Code, the output is active for at least 50 microseconds and up to 250 microseconds.

The discrete control register for “External I/O” is described below.

If using the RS-485/422 differential I/O signal as a trigger out, be sure to also enable the RS-485 transmitter as described in the

"differential output control (Word 0x8a, Bytes 0x114 – 0x115)" register description later in this chapter.

discrete_control register – Ext I/O (Word 0x89, Bytes 0x112 – 0x113)

This 16-bit register controls the external trigger(s)/clock(s). The register powers up with all zeros. This register is applicable to the R15-AMC, AMC-1553, QPM-1553, QPMC-1553, QPCI-1553, QPCX-1553, QCP-1553, PCCARD-D1553, RPCCARD-D1553, R15-EC, RPCIe-1553, Q104-1553-P, and Q104-1553 products. This register is not applicable to the R15_LPCIe, QVME or QVME2-1553 which has dedicated triggers.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ch4_in_src [1..0]		ch3_in_src [1..0]		ch2_in_src [1..0]		ch1_in_src [1..0]		trig_out_src[1][4..1]				trig_out_src[0][4..1]			

Field	Definition
trig_out_src[0][4..1]	clk/trigger 0 output source enables (channels 1 – 4)
trig_out_src[1][4..1]	clk/trigger 1 output source enables (channels 1 – 4)
ch1_in_src[1..0]	channel 1 clk/trigger input source
ch2_in_src[1..0]	channel 2 clk/trigger input source
ch3_in_src[1..0]	channel 3 clk/trigger input source
ch4_in_src[1..0]	channel 4 clk/trigger input source

Caution: Discrete_control register assignments must be made *BEFORE* setting any BC, RT, or BM run bits to ensure proper bus triggering behavior.

trig_out_src[0][4..1] and trig_out_src[1][4..1]

Two discretes may simulataneously be driven by enabled triggers (or output clocks) from any or all 1553 channels.

Each of four bits, trig_out_src[0][4..1], corresponding to each 1553 channel, are set to logic 1 if that channel's trigger / clock is to drive discrete[7] (discrete[1] for the PCCARD-D1553, RPCCARD-D1553, and R15-EC). The four bits are OR'ed together.

Each of four bits, trig_out_src[1][4..1], corresponding to each 1553 channel, are set to logic 1 if that channel's trigger / clock is to drive discrete[8] (discrete[2] for the PCCARD-D1553, RPCCARD-D1553, and R15-EC). The four bits are OR'ed together.

To drive the discrete outputs, the corresponding discrete output enable bit, discrete_oe[8..7], must also be set.

Caution: Setting multiple 1553 channels to trigger out on the same discrete output may cause unexpected results. Trigger outputs pulse for 50 microseconds duration, but multiple triggers may overlap or occur one right after the other with much less than 50 microseconds between pulses. Also, a trigger driving a discrete results in an active-low, open-drain trigger output that has a weak (22KΩ) on-board pullup. When used as an output trigger, the discrete used may require an additional external pullup resistor for high-frequency switching events.

ch1_in_src[1..0], ch2_in_src[1..0], ch3_in_src[1..0] and ch4_in_src[1..0]

If external trigger or external clock function is selected, discretes[8..7] are mapped to either EXT_CLK_IN / TRIG_IN or EXT_CLK_OUT / TRIG_OUT for any or all 1553 channels depending on the discrete direction and the programmed values in the discrete_control (Ext I/O) register.

The input trigger and input clock to the lpu share a common signal. The source of this signal is selected with these two bits. Each 1553 channel has a corresponding set of assignment bits. The sources may be the RS-485[0] receiver (trig_in), discrete[7] line or discrete[8] line.

On the PCCARD-D1553, RPCCARD-D1553, and R15-EC boards, the differential input selection is replaced by the TTL Trigger Input and will route the board's TTL Trigger Input to the selected channel.

At Power-up, no source is selected. Selection is decoded as follows.

Bit 1,0 value	Source
00	None (default)
01	Trig_in (RS-485[0] differential input)
10	discrete[7]
11	discrete[8]

differential I/O

The Q104-1553, Q104-1553-P, QCP-1553, QPCI-1553 and QPCX-1553 boards have two differential ports (one output and one input); the R15-AMC, AMC-1553, QPM-1553 and QPMC-1553 have one differential port (input-only). The R15-LPCIe has two differential ports which are bi-directional and controlled as a single ended discrete. Other UCA-based 1553 board products do not have differential I/O.

differential output control (Word 0x8a, Bytes 0x114 – 0x115)

The host programs the following register during system setup. This register powers up with zeros for all defined bits. All reserved and unused bits should be programmed with logic zeros during system initialization. Reserved bits contain undetermined values when read.

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

Field	Definition
clk_out_src[4..1]	ext_clk_out source(s)
txen	RS-485[1] transmit enable

clk_out_src[4..1]

RS-485[1] transmitter may be driven by trigger out / clock out lines from any combination of 1553 channels as determined by the clk_out_src[4..1] bits. Each bit represents the connection of 1553 channel, 1-4; a logic one enables (connects) that channel's line onto the RS-485 transmitter. If multiple 1553 channels drive the RS-485/422 transmitter, the signals are logically OR'ed together. This enabling provides only the path to the external outputs. Programming of the triggering and clock output functions is determined in a particular channel's Hardware Registers. This feature is not available on the QPMC-1553 since it only has one RS-485 component, assigned strictly as an input to the card.

Since there is no differential transmitter on either the PCCARD-D1553 / RPCCARD-D15533 or R15-EC boards, selecting the RS-485 output routes the trigger to the TTL Trigger Out signal. The transmit enable bit is a don't care since the RS-485 driver component is not included. The TTL Trigger Out signal corresponds to the state of the enabled output regardless of the state of the txen bit.

txen

This signal directly drives the RS-485[1] transmitter enable signals. It powers up in the off position (logic zero) and must be turned on (logic one) before RS-485/422 transmission can take place.

Diff_termination_en (Word 0x8b, Bytes 0x116 – 0x117)

The R15-LPCIe has 2 differential discretes with switchable termination. This register independently control the termination for each differential discrete and is also used by the RXMC2-1553 to control the termination of the Ext_TT _CLK (bit 0)and EXT_TT_RST (bit 1).

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

Field	Definition
diff_term_en [2..1]	discrete output enable (data direction)

diff_term_en[2..1]

Setting the diff_term_en bit low disables the channels 120 ohm termination. Setting the bit high enables the termination. The power up default is all termination is off (i.e. 0x00).

discrete_in (Words 0x8c - 0x8d, Bytes 0x118 – 0x11b)

Two words are allocated for the discrete inputs:

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

reserved	discrete_in [18..17]

Field	Definition
discrete_in [18..1]	Avionics discrete input

discrete_in[18..1]

If the voltage present at the discrete I/O pin is less than +2.7 VDC, the discrete_in bit reads a “0” while voltages greater then +2.7 VDC return a “1”.

Note: Discrete inputs may be used for Hardwired RT addresses, but it is recommended to read these through the rt_address_read location rather than through discrete_in (refer to the discussion of “Hardwired RT Address” later in this chapter. Likewise, use of discrete inputs for external trigger or external clock functions are described later in this chapter.

discrete_in (Word 0x8c, Byte 0x118 – 0x119) for Hardwired RT Address

If discrete inputs are used to define Hardwired RT Addressing for 1553 channel 1, the following discretes are used:

Bit name	RT Address Function
discrete_in[1]	RT_ADD1_0
discrete_in[2]	RT_ADD1_1
discrete_in[3]	RT_ADD1_2
discrete_in[4]	RT_ADD1_3
discrete_in[5]	RT_ADD1_4
discrete_in[6]	RT_ADD1_P

On boards where discrete inputs are used to define Hardwired RT Addressing for 1553 channel 2, the following discretes are used:

Bit name	RT Address Function
discrete_in[9]	RT_ADD2_0
discrete_in[10]	RT_ADD2_1
discrete_in[11]	RT_ADD2_2
discrete_in[12]	RT_ADD2_3
discrete_in[13]	RT_ADD2_4
discrete_in[14]	RT_ADD2_P

Hardwired RT addresses may be overwritten by the host through the sw_rt_add register. It is recommended that the hardwired RT

address be read through the rt_address_read location rather than through discrete_in.

rt_address_read, ch 1 and ch 2 (Word 0x8e, Bytes 0x11C – 0x11D)

The hardwired RT address for channels 1 and 2 may be read at this location. As soon as a write occurs to any sw_rt_add register the value presented here will be the same as the contents of the sw_rt_add register for ch1 and ch2.

The Hardwired RT Address and Parity input polarity is high true. A floating [disconnected] input reads back Logic One because of on-card pull-up resistors, while a grounded input reads back Logic Zero. Parity evaluation is odd; that is, the total of inputs that read Logic One [parity inclusive] must be an odd number, else the Parity Error is set.

On both the RXMC-1553 and RXMC2-1553 the value read is the latched value seen at the board pins at power-up. The actual value present at the pins (unlatched) may be read by writing a “1” to the “sel” bit for the corresponding channel. The “sel” bit is cleared upon power-up. On boards other than these the “sel” bit is not implemented.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rsvd	ch2							rsvd	ch1						
	sel	par	rta4	rta3	rta2	rta1	rta0		sel	par	rta4	rta3	rta2	rta1	rta0

rt_address_read, ch 3 and ch 4 (Word 0x8f, Bytes 0x11E – 0x11F)

The hardwired RT address for channels 3 and 4 may be read at this location. As soon as a write occurs to any sw_rt_add register the value presented here will be the same as the contents of the sw_rt_add register for ch3 and ch4. This is a read-only location.

The Hardwired RT Address and Parity input polarity is high true. A floating [disconnected] input reads back Logic One because of on-card pull-up resistors, while a grounded input reads back Logic Zero. Parity evaluation is odd; that is, the total of inputs that read Logic One [parity inclusive] must be an odd number, else the Parity Error is set.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved		ch4						reserved		ch3					
		par	rta4	rta3	rta2	rta1	rta0			par	rta4	rta3	rta2	rta1	rta0

sw_rt_add, ch 1 and ch 2 (Word 0x90, Bytes 0x120 – 0x121)

When hardwired RT addressing is enabled the hardwired RT address for channels 1 and 2 may be overwritten at this location. This register is read/write and is initially cleared upon power-up. Any write to this register causes the given address to overwrite the current hardwired address.

In sRT (RT Validation) mode, before writing to the sw_rt_add register, hardware register word 0x1B for the affected channel(s) must be loaded by the driver with the same RT address as programmed in this register.

Note: Any write to Word 0x90 OR 0x91 results in RT Addresses for all channels to take on the values provided in Words 0x90 and 0x91.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved		ch2						reserved		ch1					
		par	rta4	rta3	rta2	rta1	rta0			par	rta4	rta3	rta2	rta1	rta0

sw_rt_add, ch 3 and ch 4 (Word 0x91, Bytes 0x122 – 0x123)

When hardwired RT addressing is enabled the hardwired RT address for channels 3 and 4 may be overwritten at this location. This register is read/write and is initially cleared upon power-up. Any write to this register will cause the given address to overwrite the current hardwired address.

In sRT (RT Validation) mode, before writing to the sw_rt_add register, hardware register word 0x1B for the affected channel(s) must be loaded by the driver with the same RT address as programmed in this register.

Note: Any write to Word 0x90 OR 0x91 will result in RT Addresses for all channels to take on the values provided in Words 0x90 and 0x91.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved		ch4						reserved		ch3					
		par	rta4	rta3	rta2	rta1	rta0			par	rta4	rta3	rta2	rta1	rta0

External Trigger Out, (Words 0x98 - 0x9b, Bytes 0x130 – 0x137)

On the RXMC and RXMC2 these registers allow any of the optional I/O to act as an output trigger from any 1553 channel. To enable an I/O to function as an output trigger, the corresponding bit must be selected here. This configures the bit as an output (i.e., for discretes it is not necessary to set the discrete_oe register bit) and drives it active when it's 1553 channel generates an output trigger. For PIO and the 485 I/O the output trigger is active high while for the Avionics discretes it turns the output FET on (for OPEN/GND discretes this results in an active low signal, for 28V/OPEN discretes the signal is active high). You can select multiple bits to function as triggers for a given channel and the same bits may be selected to be the ouput trigger for both channels.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1553 Channel 1 (Word 0x98, Bytes 0x131-130)															
1553 Channel 2 (Word 0x99, Bytes 0x133-132)															
1553 Channel 3 (Word 0x9a, Bytes 0x135-134)															
1553 Channel 4 (Word 0x9b, Bytes 0x137-136)															
485[4:1]				discretes[12:5] or PIO[7:0]								discretes[4:1]			

Temp Sensor Read Command (Word 0xb0, Bytes 0x160 – 0x161)

This register allows the reading of various registers within the MAX6658 device through the Read Byte format only. Refer to the MAX6658 datasheet for more information.

The Format and description of this register is as follows:

Write Function:

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

Bit	Definition	R/W
0:7	**CMD[0:7], Command Byte**. Writing a command to this register provides various data from within the MAX6658.	W

Read Function:

Note After writing the Command Byte, you must wait a minimum of 700 us before the data is available to read.

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

Bit	Definition	R/W
0:7	**RD[0:7], Read Data**. Reads data as addressed by the Command Byte above.	R
8	**RD[8], Over Temp Alarm_I**. Asserts when temperature is above the software programmed threshold.	R
9	**RD[9], Alert_I**. Asserts when temperature exceeds user-set limits (high or low temperature).	R

Temp Sensor Write Command (Word 0xb1, Bytes 0x162 – 0x163)

The R15-LPCIe contains a Maxim MAX6658 Temperature Sensor which can provide the board temperature. Alarm set points are available. This register allows the writing of commands to the device via the Write Byte format only. Refer to the MAX6658 datasheet for more information.

The Format and description of this register is as follows:

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

Bit	Definition	R/W
0:7	**CMD[0:7], Command Byte**. Selects the register you are writing to within the MAX6658.	W
8:15	**WDB[0:7], Write Data Byte**. Used to set thresholds, configuration masks and sampling rate.	W

Board Specific Data, (Words 0xe0 – 0xe3, Bytes 0x1c0 – 0x1c7)

These 4 read only registers provide board specific data as identified below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Wd 1 (Word 0xe0, Bytes 0x1c1-1c0)															
Data Wd 2 (Word 0xe1, Bytes 0x1c3-1c2)															
Data Wd 3 (Word 0xe2, Bytes 0x1c5-1c4)															
Data Wd 4 (Word 0xe3, Bytes 0x1c7-1c6)															

For the R15-LPCIe:

Data Word	R15-LPCIe Data
Data Wd 1[15:0]	FPGA Revision
Data Wd 2[15:0]	ext_rt_addr0[5:0] (unused bits = 0)
Data Wd 3[15:0]	NOT USED (0xFFFF)
Data Wd 4[15:0]	NOT USED (0xFFFF)

CHAPTER 11

Special Options

Available Options for 1553 Universal Core Architecture Boards

Abaco Systems offers several special-order options that are not included on standard boards. This chapter discusses only those options which affect the memory map.

The "**-H**" option (available on QPM-1553) provides for eight Common-Mode RS-485 transceiver general purpose differential I/O ports. This is in addition to the input-only port described in the *Discrete Control Register – Ext I/O (0x112 – 0x113)* section.

The RXMC-1552 *nx-n***6**" option RXMC boards provides for the four least significant bits described in the following as Common-Mode RS-485 transceiver general purpose differential I/O ports

The memory locations used by these options are given in the table below.

Memory Address Byte Offset	Memory Address Word Offset	Read (R)/ Write (W)	Register
0x114 - 0x115	0x8a	W only	diff I/O control (-H option)
0x124 - 0x125	0x92	R/W	RS-485 Transmitters
0x126 - 0x127	0x93	R/W	RS-485 Transmitter Enables
0x128 - 0x129	0x94	R Only	RS-485 Receivers
0x12c - 0x12d	0x96	R Only	Reserved

General Purpose Differential I/O: RS-485 registers (Words 0x92 – 0x95, Bytes 0x124 – 0x12b)

The "**-H**" option (available on QPM-1553) includes eight differential I/O ports. The RS-485 transceivers provide differential signaling, either as an input or output. Each three-state differential driver / differential receiver pair form a Differential Channel and operate from the 5V power supply. There are separate words for transmit data, transmit enables and receive data. The Differential receive data is always readable. The Differential transmit may be written at any time, but is valid only when the transmit enable enable is set.

RS-485 Transmit Data (Word 0x92, Bytes 0x124 – 0x125)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								485_d[8:1]							

RS-485 Transmit Enables (Word 0x93, Bytes 0x126 – 0x127)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								485_de[8:1]							

RS-485 Receive Data (Word 0x94, Bytes 0x128 – 0x129)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								485_r[8:1]							

When the transmit enable bit is set, the corresponding transmit register bit is output to the RS-485 transmitter. When the enable bit is zero, the transmitter is disabled in the high-impedance (Z) state. The truth table is shown below. An "X" is a "don't care"; **485+** and **485-** are the differential wire pairs.

485_de	485_d	485+	485-
0	X	Z	Z
1	0	Low	High
1	1	High	Low

When the transmitter enable bit is cleared to a logic "0", the corresponding receiver data bit reflects the current input on the RS-485 receiver, with respect to the voltage at the differential

inputs, as shown in the table (**VID** = the voltage difference between the **485+** and **485-** wire pairs).

Differential Input (VID)	485_r
> + 0.2 V	1
- 0.2 V < VID < + 0.2 V	1
< - 0.2 V	0
Open Circuit	1
Short Circuit	1
Idle (terminated) bus	1

VID between -0.2 V and +0.2 V is a fault and 485_r settles at this value after several hundred microseconds.

When the receiver enable bit transitions to zero, the receiver register is latched and will stop inputting from the RS-485 receiver.

Note: The logic utilizes high-resolution clocked registers rather than true latches.

On the "**-H**" option QPM-1553 each RS-485/422 transceiver has a 1.9 V bias on its negative terminal to allow the positive terminal to be used either differentially or as a single ended TTL discrete.

differential output control (Word 0x8a, Bytes 0x114 – 0x115)

The "**-H**" option (available on QPM-1553) allows one of the eight transceivers, RS-485[1], to alternatively be used as an output trigger. See the section, "**Error! Reference source not found.**", for further information on this feature. A "conflict of use" warning bit that is set when both functions are enabled at once and is described below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										cou	txe n	clk_out_src_[4..1]			

Field	Definition
clk_out_src[4..1]	See Chapter 10 for description.
txen	See Chapter 10 for description.
cou	RS-485[1] conflict of use warning (–H option build)

cou

This bit is valid only for the "**-H**" option build (eight general-purpose rs-485 differential transceivers).

This bit is a "conflict of use" warning, which is set by the firmware when there is an attempt to use RS-485[1] for multiple purposes. Specifically, when the txen bit is set at the same time as the 485_de[1] bit is set (RS-485 Transmit Enable) in the rs485 register 0x126, this bit is set. When either transmit enable bit is turned off, the firmware resets this warning bit to a logic "0".