# BRM1553mRT Hardware-Software Interface Document (HSID)

## Mil-Std-1553 Multiple RT mode only

Rev 7.1 - added Time Trigger and Time tag clk - Sep 2014

Rev 7.2 – redo the command search time – Aug 2016

## TABLE OF CONTENTS

## TABLE OF FIGURES

# 1 INTRODUCTION

## 1.1 ABOUT THIS MANUAL

This document is the user's manual for BRM1553mRT core. It covers the software interface – modes of operation, memory structure and registers details - for Multiple Remote Terminal (mRT) core. The mRT mode concept of operation is based on DDC's Legacy BC mode in its operation, and was built with the intention to minimize the Software API development.

The mRT was has Error Injection capabilities.

This manual is applicable for various Sital products, including Sital's IP Cores for FPGA.

This manual does not cover the hardware interface for these products.

The hardware interface of each product (IP cores, components, boards and other products from Sital) is covered under each product's Hardware Interface Manual.

## 1.2 TERMS USED IN THIS DOCUMENT

- **IP** – Intellectual Property of Sital technology.
- **Remote Terminal (RT)** – The logic circuit that manages the 1553 communications as a remote terminal as defined in the MIL-STD-1553B standard and implemented by this core.
- **mRT (Multiple RT)** – The IP that supports up to 31 RTs in the same circuit.
- **Bus Controller (BC)** – the system that handles Mil-Std-1553 timing and manages the responses of RT's on the bus.
- **eBC** – Enhanced Bus controller.
- **Sub System** – The whole box that connects to the MIL-STD-1553 bus that contains the IP where part of it is the RT or BC.
- **Host** - the CPU running the Sub System and managing the IP core interface.
- **FPGA** – Programmable device that contains the IP core and user logic and is part of the Subsystem.
- **User Logic** – Logic Circuit that resides in the FPGA that is not the BRM1553D - RT & MT MODE core and connects to it.
- **Message** – the group of command data and status words that compose a 1553 message.
- **Core** – Supplied logic circuit that interfaces to MIL-STD-1553 bus.
- **ICD** - Interface Control Document and refers to the messages contents that run on the 1553 bus.
- **TA** – Terminal Address of the command / status words. Bits 11 to 15.
- **SA** – Sub Address of the command word. Bits 5 to 9.
- **WC** – Word count field of command. Bits 0 to 4.
- **Muxbus** – Time multiplexed bus known as the MIL-STD-1553B Notice 2 bus.
- **SW** – software.
- **BCST** – Broadcast command.
- **Tx** – Transmit.
- **Rx** – Receive.
- **SACW** – The Sub-address control word.
- **LUT** – Look Up Table.
- **EOM** – End of message.
- **EOF** – End Of Frame.
- **ISR** – Interrupt Status Register.
- **ms** – Micro seconds
- us – Micro seconds

# 2 MULTIPLE RT INTRODUCTION

## 2.1 OVERVIEW

The Mil-Std-1553 Multiple Remote Terminals (mRT) allows a subsystem to fulfill some or all of the RT's that are required to reply to the bus controller.

This special mode is designed to simulate system level, real time transactions on the 1553 bus with a single memory management IP.

The mRT mode supports error injection, making it possible to inject errors into the message responses of the RTs as defined in the error injection user's manual.

The concept of the mRT is based on a bus list of message entries preloaded into the memory of the IP. When a 1553 command arrives, the bus list is scanned for a match to this command, and if a match is found, the message is served with the preloaded data for that entry.

Data of message list can be changed during the run, but the switch has to be synchronized with the IP using several data block memory management modes.

The word count field is not compared during the search, so that word count field of the command is ignored for each message. The actual amount of words transferred depends on the word count of each specific message.

If an RT is enabled, but the scan did not match any message in the bus list, the user can select between a reply with zero data, or a reply with status Message Error (ME) set and no data transmitted / stored.

Multiple memory management modes are supported, including single, double buffer, circular or global circular buffer modes for each message.

For each message, an interrupt can be set, if required based on setup of the control field of that message.

The concept of frame time is not applicable for mRT, thus all messages for all minor frames must be loaded in bus list.

The limit on the number of messages depends on the response time length ( <12 us) and the scan speed of the IP. Simple scan starts from the first bus entry and finishes at the last. A quick scan method is supported in the case that the bus entries' commands are loaded to the memory in an incremented order. See quick scan method below.

RT2RT messages are also supported, even if both RTs are simulated by the mRT IP.

## 2.2 BLOCK DIAGRAM

The host processor controlling the core accesses the memory and registers located in the host interface block. By setting the values of the configuration registers and setting up the bus list in memory, the host processor defines which commands would be serviced by the IP.

The following drawing shows the BRM1553mRT Block Diagram:

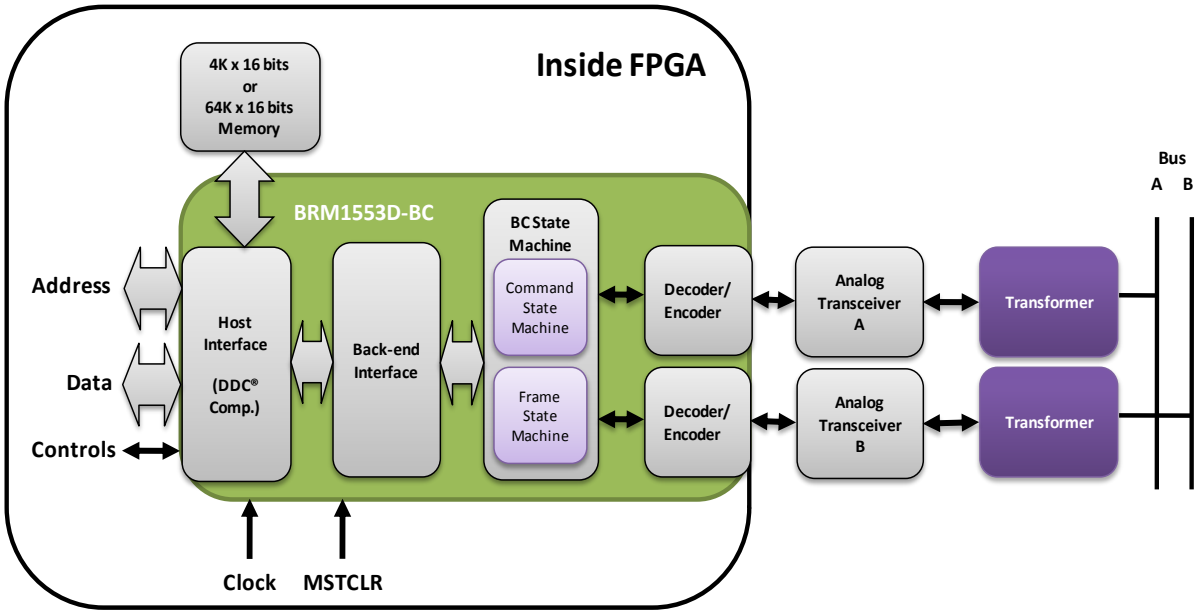



Figure 1: BRM1553mRT Block Diagram

### 2.2.1 THE STATE MACHINES

When mRT is initiated the state machine manages the sequencing of the messages as defined by the configuration registers and message setting in the bus list.

The commands are stored in a bus list. The host points to the beginning of the first command entry in bus list and defines how many entries are supported through 2 registers.

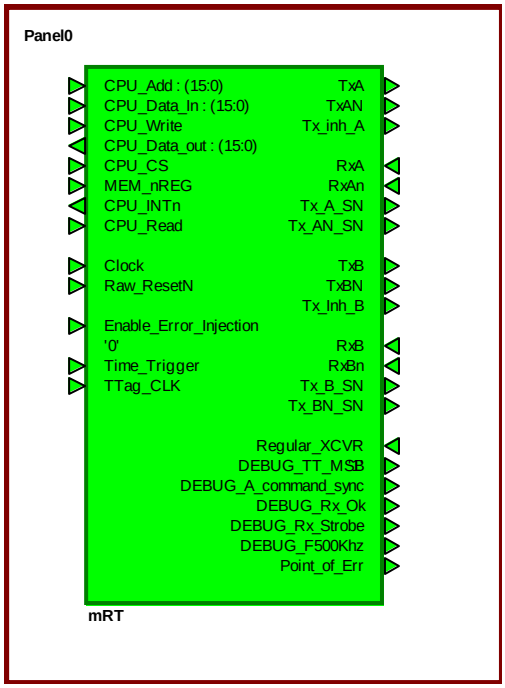
When all data has been loaded for all entries, and the state machines are idle, the host sends a START command. As a response, the frame state machine waits for a coming command from the mRT message state machine. The command is scanned in the bus list for a match.

When a match is found, the commands' data and status are managed to complete that bus list entry. When the message is finished, the frame state machine resets to the first location in list, waiting for the next command.

The mRT message state machine either transmits words through the encoders or receives data words through the decoders. The encoders and decoders interface between the core's 16 bits parallel internal buses and the 1553 serial bus.

Error injection is supported. A set of registers dedicated to error injection, define to which bus list entry the error is injected, to which word, to which bit and what type of error.

### 2.2.2 BRM1553MRT PINS




The core connects with the CPU and User Logic with the following lines:
All CPU side signals take action on the rising edge of clock.

| Signal Name | In/ out | Description |
|---|---|---|
| Clock | In | The core is synthesized to work with a specific clock frequency as requested by the user. This pin should be connected to that clock. The clock should be one of the global clocks of the FPGA. The logic circuits that interface with the core should also work with this clock. |
| Raw_ResetN | In | An active low signal that asynchronously resets all of the FFs in the core. This signal is internally sampled to the clock. This signal does not reset the dual port memory of the message data. This signal can be used to inhibit the core from usage. Once the ResetN is logic high, the Remote terminal starts to operate and respond to the bus based on memory and register settings. |
| CPU_CS | In | CPU Chip select. When logic high, enables CPU accesses. When low, CPU accesses are ignored. |
| CPU_Address | In | (15..0) A 16 bit address used to access the memory array and configuration registers. Memory depth is generically defined when the core is synthesized. Up to 64K x 16 bit memory is possible. |

| | | |
|---|---|---|
| MEM_nReg | In | When logic high connects the CPU address and data lines to the memory array.<br>When low connects the configuration registers. |
| CPU_Data_in | In | (15..0) A 16 bit data bus. Contains the data words written by the CPU on to the memory array and configuration registers.<br>This bus may be configured to support 16 or 32 bits widths. |
| CPU_WE | In | Logic high on this signal when rising edge of clock will write **CPU_Data_in** into the address defined by **CPU_Address** and **MEM_nReg.**<br>When using wide bus (wider than 16 bits) this signal becomes a bus of signals, one for each 16 bits. CPU_WE(0) controls the bits 15 down to 0, CPU_WE(1) controls the bits 31 down to 16,… |
| CPU_RE | In | This signal is not used, and is here as a place holder.<br>Unlike DDC interface, there is NO read and clear option for the interrupt status register.<br>Data is read to the CPU_Data_out lines one clock after an address is set. |
| CPU_Data_out | Out | (15..0) A 16 bit data bus. Contains the data words read by the CPU from the memory array or registers. This bus will display the contents of the memory or register at address - CPU_address with one clock delay between address and data. A read always occur. The CPU should typically ignore this data when a write cycle is executed.<br>This bus may be configured to support 16 or 32 bits widths. |
| CPU_INTn | Out | An active low interrupt line set by the core based on the message settings in the memory, or the configuration registers. Please see configuration methods as defined by configuration register #1. |
| Enable_Error_Injection | In | Logic high allows the insertion of errors based on error registers.<br>Logic low prevents from adding errors. |
| Point_of_Err | Out | This signal can be used as a synchronizing signal for a scope to capture the approximate location of the inserted error. It remains high until end of message, thus trigger on its rising edge.<br>The error is actually placed within 1 us of this rising edge depending on the fault type. |
| Time_Trigger | In | Upon rising edge of this signal the 32 bit time tag counter from registers 4 and 5 are latched to registers 0x14 and 0x15 accordingly. Later on, these latched time tag can be read by the CPU for accurate analysis of the sequence of events. |
| Tag_Clk | In | Upon rising edge of this signal, the time tag register is incremented if the time tag resolution is set to value 7 in configuration register.<br>This signal is internal sampled by the clock signal, so it's maximum frequency has to lower than half the clock frequency. |

The core connects with the transceiver with the following lines for each of the busses (not relevant for BRM1553D-EBR):

| Signal Name | Dir | Description |
|---|---|---|
| Regular_XCVR | In | '1' – Use COTS transceiver.<br>'0' – Use Sital Technology Discrete component transceiver. |
| RxA, RxAn,<br>RxB, RxBn | In | The 1553 receive signals.<br>The transceiver Rx Enable signal, if exist, should be tied to constant enable level and is not used by the core. |
| TxA, TxAn,<br>TxB, TxBn | Out | The 1553 transmitter signals.<br>Drive the bus. Connected to the Tx lines of the transceiver. Bus A signals and Bus B signals. |
| Tx_inh_A,<br>Tx_inh_B | Out | Connected to the tx inhibit line of the transceiver.<br>**This line must be tied to a pull up on the PCB**. This will inhibit transmission for the period of time that the FPGA is not loaded and all of its ports are logic tri-state.<br>If Sital transceiver is used (BRM1553D-RTMT-S), these signals are connected to the En_A and En_B of the transceiver circuit. |
| Tx_A_SN,<br>Tx_AN_SN,<br>Tx_B_SN,<br>Tx_BN_SN, | Out | Relevant for BRM1553D-RTMT-S only.<br>Connects to Sital Technology transceiver signals. Should be left unused if COTS transceiver is used. |

The following are debug pins used for the bring-up stage of the core. All are outputs of the core.

| Signal Name | Debug Pins Description |
|---|---|
| DEBUG_TT_MSB | The MSB of the time tag register. If the TT resolution is 64 microseconds, then this signal is a 0.25 Hz clock. 2 seconds on, 2 seconds off.<br>If this signal does not work, the IP core is reset. Check RAW_ResetN and verify it is logic high. |
| DEBUG_F500Kz | 500KHz clock.<br>If this signal is not 500KHz then the clock fed to the BRM1553D is not the correct clock.<br>Each net list is synthesized to a specific clock. Please see the documentation to verify the target clock for this core. |
| DEBUG_Rx_Strobe | For every 1553 word on bus A OR bus B, this pulse is expected during the end of the parity bit.<br>If this signal does not pulse and 1553 words are on the bus, it means that the plus and minus or Rx and RxN signals are not connected to the 1553 bus.<br>This signal should pulse even if the receive lines are swapped!<br>See DEBUG_A_Command_sync for polarity check. |
| DEBUG_Rx_OK | This signal is the bus A OR bus B decoder error flag. If there is a sync error (inverted sync is not an error) or bi-phase error, or parity error (inverted rx lines makes a parity error!) this signal goes low.<br>Normally for valid words, this signals will always stay high.<br>If there is a bus coupling issue (missing 78 Ohm termination), this signal will go low about 2 us after the end of the 1553 word. |
| DEBUG_A_Command_sync | This is the command sync for bus **A only**!<br>Its should be set (or stay) high just after the mid sync of the command and status words, and go low just after the mid sync of the data words on the bus A.<br>If it goes low after command mid sync, you have the two receive lines swapped (bus A). |

## 2.3 MULTIPLE RT OPERATION MODES

### 2.3.1 BASE OPERATION

The host manages the 1553 bus through the shared memory and several registers. The host should load these registers and memory prior the START command, and then analyze the results in memory after the messages transactions have been finished.

The memory data defines which commands need to be supported. The host stores the commands to be supported in the bus list area of the memory.

The list defines a set of 4-word descriptors. Each descriptor describes a message that needs to be handled by the core.

**Start Pointer** register is initially set by the host and points to the first bus list entry from which the core starts searching the incoming command. The 2 LSBs are ignored because each entry occupies 4 memory words.

Using the **list length** register the host defines the number of bus list entries that the core is required search for a match.

### 2.3.2 MULTI FRAME SETUP

In most avionics systems, not all messages are transmitted every frame. There are messages that need to be transmitted once every 2 frames, others every 4 , 8 …

For mRT there is no notion of minor frames, the mRT should load all messages for all frames to a single bus list.

### 2.3.3 STOP MULTI RT

mRT would work from the Host's START command until the Host's STOP command. Stop is graceful, it will not cut a message in the middle, but rather go to off when the last message has finished. See configuration register options.

### 2.3.4 STOP ON ERROR

When things do go wrong and messages fail, the state machine can be programmed to stop.

The core state machine can be stopped at the end of a failed message.

### 2.3.5 STATUS BITS

Each RT has internal status bits kept in the remote terminal. The host can also set additional status bits to be set per entry. The IP keeps the internal status bits unique for each RT, and capable of setting BCST bit on to ALL in case a broadcast message is on the bus.

### 2.3.6 COMMAND SEARCHING MODES

During run time, when a command is decoded, a search for it in the bus list starts. The search starts from the first entry and runs for bus list length iterations.

The scan has been optimized to run in clock rate. That allows for a large number of commands to be supported. The status response is done automatically, so there is more than 20 microseconds for the scan. If the clock frequency is 33Mhz, then assume 33*20 messages as a minimum, that is more than 600 messages.

However, if status bits are required to be set on the return status word for transmit messages, these have to be ready before status is replied. There are just less than 8 us for this search, or else the status transmitted would be clear status. For 50Mhz IP, these 8 us allow almost 400 messages.

Thus place all transmits command messages in the first 400 messages.

Notice that the commands order in the bus list should be based on the memory block size, biggest first.

# 3 PROGRAMMING AND SETUP

## 3.1 REGISTERS

There are several registers mapped to the register section (when MEM_REG signal is kept low) during a read or write operation.

The following section describes the registers and the operational bits of these registers.

### 3.1.1 REGISTERS MAP

The software interface of the core to the host processor consists of operational registers for normal operation. These registers determine the device configuration, modes of operation, memory structure, interrupt control and status, etc.

The address mapping for the registers is detailed in the following table:

| Address Lines – 16 bit wide | | | | | Register Description (Read/Write) |
|---|---|---|---|---|---|
| **A4** | **A3** | **A2** | **A1** | **A0** | |
| 0 | 0 | 0 | 0 | 0 | Interrupt Enable Register #1 |
| 0 | 0 | 0 | 0 | 1 | Interrupt Status Register #1 |
| 0 | 0 | 0 | 1 | 0 | mRT Configuration Register |
| 0 | 0 | 0 | 1 | 1 | mRT Configuration Register #2 |
| 0 | 0 | 1 | 0 | 0 | Time Tag Register LSBs 15..0 |
| 0 | 0 | 1 | 0 | 1 | Time Tag Register MSBs 15..0 |
| 0 | 0 | 1 | 1 | 0 | Start/Reset Register |
| 0 | 0 | 1 | 1 | 1 | |
| 0 | 1 | 0 | 0 | 0 | RT15 down to RT0 enable register |
| 0 | 1 | 0 | 0 | 1 | RT30 down to RT16 enable register |
| 0 | 1 | 0 | 1 | 0 | Command List pointer |
| 0 | 1 | 0 | 1 | 1 | Command list length |
| 0 | 1 | 1 | 0 | 0 | Global Data pointer for Rx messages |
| 0 | 1 | 1 | 0 | 1 | Global Data pointer for Tx messages |
| 0 | 1 | 1 | 1 | 0 | Messages count |
| 0 | 1 | 1 | 1 | 1 | version of Core |
| 1 | 0 | 0 | 0 | 0 | Error Register #1 |
| 1 | 0 | 0 | 0 | 1 | Error Register #2 |
| 1 | 1 | 0 | 1 | 0 | 0xA001 – Name of IP core – BRM1553mRT |
| | | | | | |

### 3.1.2 INTERRUPT ENABLE REGISTER #1 ADDRESS 0X0

If the Host enables one of the bits below by setting its value to '1', and the specified event occurs, an interrupt will be generated to the Host CPU.

| Bit number | Read/ Write/ Default | What event triggers the interrupt when enabled by '1'. |
|---|---|---|
| **13** | R/W/0 | Transmitter timeout occurred |
| **8** | R/W/0 | State machine time out error. |
| **7** | R/W/0 | Writing to a wrong register |
| **6** | R/W/0 | Time tag counter rollover |
| **4** | R/W/0 | Set if the message that ended had the EOM interrupt enabled. |
| **3** | R/W/0 | Last message in list has been finished. |
| **2** | R/W/0 | 1553 message error occurred. |
| **0** | R/W/0 | EOM occurred. |

*All other bits are not used and will be read as zero. Default Value: 0x0000 – all events masked.

### 3.1.3 INTERRUPT STATUS REGISTER #1 ADDRESS 0X1

This register indicates the cause of an interrupt.

The status bits will be cleared as a result of writing to this register. Writing a '1' to a specific bit clears its value to '0'. If the cause of the interrupt bit has not been solve, this bit will go high again.

When an interrupt occurs, and the register is read, the host services the bits that are high, then writes back the register with the value it received. That way, only the bits that were serviced are cleared, others stay high.

Writing 0xFFFF to this register clears all bits.

| Bit number | Description |
|---|---|
| **15** | This bit is an OR of all interrupts bits that are not masked by interrupt mask register. It reflects the HW interrupt line in level mode, and indicates there is still an interrupt request in the pulse mode. |
| **13** | '1' – Transmitter fail safe timeout. |
| **8** | '1' – State machine timed out on message manage |
| **7** | '1' – Host CPU wrote to an unmapped register. |
| **6** | '1' – 16 LSBs Time Tag Rollover from 0xFFFF to 0x0000. |
| **4** | '1' – Message has ended. This interrupt can be enabled or not for each message with mRT control word bit 4. |
| **3** | '1' – EOM of Last message in bus list (~= BC EOF interrupt) ** |
| **2** | '1' – 1553 message error occurred. |
| **0** | '1' – EOM has occurred. |

*All other bits are not used and will be read as zero.

** it would be useful to place the last message of the major frame as the last bus list entry to get full benefit of this interrupt.

Default Value: 0x0000

### 3.1.4 CONFIGURATION REGISTER #1 ADDRESS 0X2

| Bit number | Read/ Write/ Default | Description |
|---|---|---|
| **15..14** | R/W/0 | These three bits define the size of the Receive Global Circular buffer.<br>0 – 256 words.<br>1 – 1K words.<br>2 – 4K words.<br>3 – 16K words. |
| **13..12** | R/W/0 | These three bits define the size of the Transmit Global Circular buffer.<br>0 – 256 words.<br>1 – 1K words.<br>2 – 4K words.<br>3 – 16K words. |
| **10..8** | R/W/0 | Time Tag resolution:<br>0 – 64 us,<br>1 – 32 us,<br>2 – 16 us,<br>3 – 8 us,<br>4 – 4 us,<br>5 – 2 us,<br>6 – Increment each time a “1” is written to Start Reset Register bit 4.<br>7 – Increments on Rising edge of TTAG_CLK. |
| **6** | R/W/0 | ‘0’ – Time Tag counter controlled by Host only.<br>‘1’ – clears or loads the Time Tag counter LSBs when synchronize with or w/o data mode command is received for ANY simulated RT. |
| **5** | R/W/0 | ‘0’ – generates a 500ns low pulse on the INTn signal on any enabled event.<br>‘1’ – Level mode. INTn stays low until the host clears enable ISR. |
| **4** | R/W/0 | ‘1’ – Enables illegalization. A status ME bit is returned for messages that are not on the bus list, but enabled for simulation.<br>‘0’ – No ME bit is sent. Data for messages not on the bus list, but for RTs that are enabled is dumped for receive, and sent all zero for transmits. |
| **3** | R/W/0 | ‘1’ – Stop mRT at end of a failed message. |
| **2** | Read | ‘1’ – mRT state machine has been started. Stays high until STOP command.<br>‘0’ – mRT is idle (after power up, STOP command or “stop on error” occurred). |
| **1** | Read | ‘1’ – bus active – as long as there is no gap of more than 1 ms, else goes to ‘0’. |
| **0** | Read | ‘1’ - Message active – from start till end of message. |

*All other bits are not used and will be read as zero. Default Value: 0x0000 – all events masked.

### 3.1.5 CONFIGURATION REGISTER #2 ADDRESS 0X3

Place holder for future configurations.

### 3.1.6 TIME TAG REGISTER LSBS – ADDRESS 0X4

### 3.1.7 TIME TAG REGISTER MSBS – ADDRESS 0X5

The time tag register is a 32 bit register. The 16 LSBs are tagged to each bus list entry upon message completion.

When the 16 LSBs cycle from 0xFFFF to 0x0000, the 16 MSBs are incremented by 1 and an interrupt bit is set in the ISR.

The time tag counter can be set by writing to these registers.

All 32 bits can be SW reset by writing to start / reset register.

The time tag counter counts in clock steps of 2us, 4us, 8us, 16us, 32us or 64us or by external clock, or by writing to a bit 4 of the Start Reset Register.

When any enabled RT receives synchronize with data command, the 16 LSBs of the time tag counter is set based on that data only if enabled by configuration register. The counter will reset upon synchronize without data if enabled by configuration register.

| Bit number | Read/ Write/ Default | Description |
|---|---|---|
| **15..0** | Read/Write 0x0 | Read the Time Tag Counter.<br>Write a new value to the time tag counter. |

### 3.1.8 START RESET REGISTER WRITING TO ADDRESS 0X6

This is a write only register.

| Bit number | Read/ Write/ Default | Description |
|---|---|---|
| **6** | Write | '1' stops mRT operation after end of current message. |
| **4** | Write | '1' Increments Time Tag by one when Time Tag resolution is set to 6. |
| **3** | Write | '1' resets the 32 bit Time Tag counter. |
| **1** | Write | '1' mRT START command. Starts the mRT state machine. |
| **0** | Write | '1' Reset core. Reset all registers, FFs in core. Memories are not reset, and should be reset by CPU.<br>Messages are potentially cut in the middle! |

### 3.1.9 RT15 DOWN TO RT0 ENABLE REGISTER ADDRESS 0X8

### 3.1.10 RT30 DOWN TO RT16 ENABLE REGISTER ADDRESS 0X9

31 bits each enables simulation for a specific RT.

RT0 is enabled if Register 8 bit 0 is '1'.

RT15 is enabled if Register 8 bit 15 is '1'.

RT16 is enabled if Register 9 bit 0 is '1'.

RT30 is enabled if Register 9 bit 14 is '1'.

Broadcast messages are supported by default. If the mRT is enabled and a broadcast message arrives, it is searched in the bus list, and if it exists, its data will be stored in the data buffer. If no match is found, data is dumped.

All RTs will be tagged as received a broadcast message, thus, if transmit last command or transmit status mode commands is received as the following command after the broadcast command, the broadcast bit will be set for the status response of that RT. If any other command is following the broadcast command for a specific message, the broadcast bit is cleared for the specific RT and not sent in its status response. (Complicated - But this is the 1553B notice 2 standard requirement).

### 3.1.11 COMMAND LIST POINTER ADDRESS 0XA

After starting execution of the mRT, when a new command arrives, this register points to the first bus entry in the bus list.

Two LSBs are ignored and should be set to "00" for future compatibility.

### 3.1.12 COMMAND LIST LENGTH ADDRESS 0XB

After starting execution of the mRT, when a new command arrives, this register defines the number of bus entries to search for a match.

Each entry is 4 words, but the command list length register defines the number of entries. So, if this register has the value of 3, the bus list is expected to have 12 words.

Value of 0 is legal; expect only responses as in no match condition.

### 3.1.13 GLOBAL DATA POINTER FOR RX MESSAGES ADDRESS 0XC

When data buffer mode is set to 7 for a specific bus list entry, then the data received from the 1553 bus is saved to a memory location pointed by this global data pointer register, rather than to the bus list entry data pointer.

Multiple or all bus list entries can use this common data pointer, by setting data buffer mode to 7.

For receive messages, after a successful EOM, the data pointer of the bus list entry is updated with a pointer to the first data word of the message. This global data pointer would still point to the next “empty” location.

The size of this global data pointer is defined in the configuration register.

The increment after each data word is only for the relevant amount of address LSBits defined by the configuration register. Like all other data pointers, the relevant LSBits cycle back to 0, even if the initial value of the pointer had LSBs other than 0.

For example, if 4K buffer is selected (mode 2), and if the initial buffer defined by the host was 0xBCDE, then this counter would increment all the way to 0xBFFF, and then, on the next increment will become 0xB000, and so on.

### 3.1.14 GLOBAL DATA POINTER FOR TX MESSAGES ADDRESS 0XD

When data buffer mode is set to 7 for a specific bus list entry, then the data transmitted to the 1553 bus is read from a memory location pointed by this global data pointer register, rather than from the bus list entry data pointer.

Multiple or all bus list entries can use this common data pointer, by setting data buffer mode to 7.

For transmit messages, after a successful EOM, the bus list entry data pointer is updated with value pointing to the first data word transmitted in the message. This global data pointer points to the next data word for transmission.

The size of this global data pointer is defined in the configuration register.

The increment after each data word is only for the relevant amount of address LSBits defined by the configuration register. Like all other data pointers, the relevant bits cycle back to 0, even if the initial value of the pointer had LSBs other than 0.

For example, if 256 buffer is selected (mode 0), and if the initial buffer defined by the host was 0xBCDE, then this counter would increment all the way to 0xBCFF, and then, on the next increment will become 0xBC00, and so on.

### 3.1.15 MESSAGE COUNT ADDRESS 0XE

This register is incremented on each successful EOM. It counts only messages that where serviced, messages that are addressed to RTs that are not enabled.

Notice that after the IP core started, it services all broadcast messages, and these are counted.

This register can be written to clear it. When the IP core is stopped, this register retains the message count, and when the IP core is started again, it will continue to increment.

This counter counts up to 64K and then resets to 0.

This register can be used to verify that all messages were serviced.

### 3.1.16 VERSION OF CORE ADDRESS 0XF

This register returns the revision of core.

16 bit date code: 0xDDMY

DD is in the range of 01 to 31, M is in the range of 1 to C, Y is in the range of 0 to F (2000 to 2015)

Example: 0x249D is 24 Sep 2013.

### 3.1.17 ERROR INJECTION ADDRESS 0X10 & 0X11

These registers define the error injection parameters for the BRM1553mRT.

Please refer to “BRM1553D_Ver70_Error_Injection_Mode” document for details on how to inject an error.

### 3.1.18 TIME TAG LATCHED DATA ADDRESS 0X14 AND 0X15

When the external “Time_Trigger” signal rises from ‘0’ to ‘1’, the 32 bit time tag of register 4 and 5 are latched to registers 0x14 and 0x15 accordingly.

The trigger of time tag is usually used in conjunction with a slow pulse that synchronizes the system, such as PPS signal.

When this pulse arrives, it usually interrupts the CPU to perform some computational tasks. One of the parameters is the time tag counter when the trigger occurred. Using these time tag latch registers, relieves the CPU from the need to pick up the time tag register (which constantly increments) but simply have all the time in the world to access these latch registers.

### 3.1.19 TYPE OF IP CORE ADDRESS 0X1A

In order to distinguish between the BRM1553D IPs and the BRM1553mRT IP, register 0x1A is set to 0xA001.

In the BRM1553D this register returns the last 4 digits of the equivalent DDC BU-6xxxx device that it replaces.

The API SW checks this register value when trying to initialize the device to mRT mode.

## 3.2 MEMORY

### 3.2.1 BUS LIST

The Bus list contains all the commands that the core IP should support during run time.

Each command entry is built of 4 words:

1. The configuration and status of this bus list entry.
2. The time tag at which this bus list entry was last serviced.
3. Data pointer which points to the first location of the data that was either received in a receive message or transmitted in the transmit message.
4. The 16 bit 1553 command word of the message that this entry services.

When a new command is received, the IP core searches the bus list, starting from the **bus list pointer** for as many entries as defined by the **bus list length** register. If a match is found, the message is served with the parameters defined in the configuration entry and the data is placed or read from the data pointer location. The 16 LSBs of time tag counter is written to the time tag entry.

The match search logic does not compare the word count field (5 LSBs) of the command in order to support dynamic word count changing by the 1553 bus controller. Instead, the 5 LSBs are loaded from the new command that is serviced so that the host can determine how many words are relevant in the buffer.

### 3.2.2 COMMAND ENTRY

Each bus list entry in the IP is composed of the following 4 words:

| Offset | Description | Comments |
|---|---|---|
| **+0** | Configuration (15..4)<br>Bit 15 – End of message<br>Bit 14 – Set EOM interrupt<br>Bit 13 – Ended Not Ok<br>Bit 12 - Service Request<br>Bit 11 – Busy<br>Bit 10 - Sub System Flag<br>Bit 9 – Dynamic bus control<br>Bit 8 – Terminal Flag | Write and read by host.<br>Set high when entry has finished.<br>Sets bit 4 of ISR if enabled to '1', on End Of Message.<br>Sets if last seen as a bad message. Ignore TTag and Word Count<br>Sets Bit 8 in status word for Service Request bit.<br>Sets Bit 3 in status word for Busy bit.<br>Sets bit 2 in status word for Sub System Flag<br>Sets bit 1 in status word for Dynamic Bus Control.<br>Sets bit 0 in status word for terminal flag. |
| | Bit 6..4 – Data buffer mode | In all modes, Data pointer is incremented on successful messages only. Bad messages will not change the data pointer supporting transmits retries of the same data.<br>0 – Set to single buffer mode. Starts from **Data pointer**.<br>1 – Set to double buffer mode. Starts from **Data pointer**.<br>Data pointer bit 5 is swapped on end of message.<br>2 – Set to circular buffer of 256 words.<br>Only 8 LSBs of data pointer are incremented.<br>Data pointer always points to next write location.<br>3 – Set to circular buffer of 1K words.<br>Only 10 LSBs of data pointer are incremented.<br>Data pointer always points to next write location.<br>4 – Set to circular buffer of 4K words.<br>Only 12 LSBs of data pointer are incremented.<br>Data pointer always points to next write location.<br>5 – Set to circular buffer of 16K words.<br>Only 14 LSBs of data pointer are incremented.<br>Data pointer always points to next write location.<br>6 – For future use. For now acts as 5.<br>7 – Global Buffer mode.<br>There is a global data pointer for Tx and one for Rx.<br>This supports data block transfer wrap around mode. |
| | Entry Status (3..0)<br>Bit 3 – Entry in Progress<br>Bit 2..0 – Msg Count | Updated by core read by host. Write by host is ignored.<br>'1' from start of message till end.<br>Increments by 1 each time this message is finished. Cyclically. |
| **+1** | Time Tag | 16 LSBs or time tag, and read by host.<br>Updated by core when command arrives. |
| **+2** | Data Pointer | Initialized by host, Updated by core.<br>Depends on data buffering mode.<br>Examples below. |
| **+3** | Command | Written by Host read and updated by core. The 1553 command.<br>Word Count field is ignored for match (for non-mode messages).<br>Word count field is updated by the IP core at end of successful message |

### 3.2.3 DATA BUFFERING MODE

#### 3.2.3.1 SINGLE BUFFER - 0

All data words for receive commands are stored at the same location pointed by Data Pointer in the message entry.

All data words for transmit command are read from the same location pointed by Data Pointer in the message entry.

The data pointer for either type of message is not changed, and is identical to the data pointer written by the host in the first place.

It is recommended to allocate 32 words for this mode, but it is not mandatory.

#### 3.2.3.2 DOUBLE BUFFER MODE - 1

This buffering mode is the same as single buffer, except for the core IP inverting the sixth bit of the data pointer, following a successful message. If the message not properly completed, then this bit is not inverted.

It is recommended to set the 5 LSBs of data pointer to zero for memory alignment, but it is not mandatory.

The core IP increments the 6 least significant bits after each data word transaction.

It is recommended to allocate 64 words for this mode, but it is not mandatory.

#### 3.2.3.3 CIRCULAR BUFFER MODE – 2 TO 5

256 words for mode 2, and up to 16K words for mode 6 are provided by this data buffering mode.

Following each successful data read or data write, depending on the command, the data pointer is incremented by one in a cyclic manner, depending on the buffer size.

For 256 word buffer, only the 8 LSBs of data pointer are cyclically incremented.

For 16K word buffer, only the 14 LSBs of data pointer are cyclically incremented.

The data pointer in the bus list entry points to the next word to be written or read in the next message. The word count field of the successfully finished message is updated in the 5 LSBs of the command entry.

#### 3.2.3.4 GLOBAL BUFFER MODE – 7

In order to save space and to support data block transfer wrap around mode, two global data pointers are provided.

Each command entry that points to these global data pointers using mode 7, would store or read its words from these common data buffers.

A possible all data wrap around mode can be setup if both global data pointer for Tx and Rx start at the same address, and if the total word count of transmit commands match the total amount of word count of receive commands.

The global data pointer buffer sizes can be selected between a number of sizes set in the configuration register #1.

# 4 WORK FLOW

## 4.1 OVERVIEW

The host should prepare the registers and memory prior to initiating the START command. The recommended setting is as follows:

1. Set the Interrupt Enable Register, mRT configuration registers, RTs simulation enable registers, and optionally the global data pointer registers.
2. Set the **Command List Pointer register** to the bus list start point.
3. Set the **Command List Length** register to define the number of entries to participate in this Run.
4. Load the bus list with 4 word entries for each ICD command that is required to simulate with the mRT.
5. Setup the data buffer modes and load the data to the relevant buffers.
6. START command.
7. Track the execution of the ICD using a polling approach or by interrupts, and download received data, and upload new fresh data.
8. When it is time to finish, issue a stop command.

## 4.2 SW UPDATING DATA FOR TRANSMIT COMMANDS

In case it is required to change data for an entry (message), there are a few ways to do that, depending on the application.

### 4.2.1 SENSOR DATA SOURCE

For a sensor type of data source, data is assumed to be updated every now and then, not synchronized with the messages rate on the bus. Whenever the sensor has the relevant data, it would like to update it and have it ready to be sent in the next appropriate message.

A method to handle this type of data source is to allocate more than one data buffers for that message in the IP core memory space. Do the following steps:

1. Preset the memory buffer mode for this bus list entry to mode 0 (single buffer mode).
2. Allocate two or more buffers for this message with adequate memory space.
3. Load the first buffer with latest data – and only then…
4. Store the pointer to this first buffer to the data pointer in the bus list entry.
5. Wait until the sensor produces updated data.
6. Load the new data into second buffer.

7. Only AFTER all data is loaded, store the pointer to second buffer to the data pointer in the bus list entry.

The IP core loads the data pointer only when the command has arrived and a match is found with a bus list entry. During the transmission of the message, which could take 680 microseconds, the data pointer increments internally – independent of the data pointer in memory, and thus f the host SW updates a new data pointer in memory, it is not being used until the next message arrives.

Data integrity is maintained, because there are two or more different buffers involved.

In the case that the sensor updates the data faster than the rate of the message, sometimes, some buffers would be overwritten before they are transmitted to the bus. This should be dealt with the system manager.

In the case that the sensor produces the data in a slower rate than the message on the bus, then multiple messages are expected to carry the same data. This should be fine, if such defined.

### 4.2.2 BLOCK DATA TRANSFER

For a data source which has to transfer a block of data words that exceed the word count of a 1553 message, it is desired to transmit the block of data WITHOUT repeating parts of the block more than once.

Such cases could be sensors which produce more than 32 data words structures, or SW loaders that want to flash a new version of SW on to a 1553 subsystem and alike.

A method to handle this type of data source is to allocate a large data buffer for that message in the IP core memory space. The IP core provides various data block sizes that can be defined per bus list entry.

Depending on the respondents of the Host SW and the rate at which the block is going to be read with 1553 messages, allocate a data buffer that is big enough to allow the SW to add more data to the block before the 1553 messages read it.

Do the following steps:

1. Preset the memory buffer mode for this bus list entry to mode 2..5 (block buffer mode).
2. Allocate a data buffer for this message in the relevant size (256 to 16K words).
3. Store a pointer to this buffer to the data pointer in the bus list entry.
4. Manage a SW data pointer that point to the next vacant word in the buffer. Should start with the same value as the bus list entry stored pointer.
5. Load the buffer with data and update the SW pointer to the next vacant location.
6. Wait for no more than 50% of the duration that the 1553 is expected to read the data using transmits commands.
7. Check the data pointer in the bus list entry data pointer.
8. Load new fresh data starting from the SW data pointer but less then the bus list entry data pointer value.
9. Update the new SW data pointer.
10. Repeat these steps as long as required.

Notice that the buffer is cyclic. Make sure the SW data pointer cycles in accordance with the buffer size mode selected.

Notice: Worst case reading of 16K words at a maximum rate of 20us per word, takes more than 320 ms.

## 4.3 READING DATA FROM RECEIVE COMMANDS

In case it is required to read all data for an entry (message), there are a few ways to do that, depending on the application.

### 4.3.1 SINGLE BUFFER DATA SOURCE - 0

For a messages that their data is not important, but needs to be saved in a buffer in the IP memory, select a single data buffer.

If you manage a few such receive commands, all can point such “garbage” buffer.

Data will be overwritten any time that message is on the bus.

### 4.3.2 DOUBLE BUFFER DATA SOURCE - 1

For messages that produce data in a rate which is less than the host CPU accesses the IP core, use the double buffer.

For Airborne real time applications this is a typical case.

Follow these steps:

1. Preset the memory buffer mode for this bus list entry to mode 1 (double buffer mode).
2. Allocate 64 words buffer for this message aligned to 6 zero LSBs.
3. Store the pointer to this buffer to the data pointer in the bus list entry.
4. Wait until the update period of time.
5. Read the bus list entry data pointer value. The 6 LSBs could be either 0, or 32 (0x20).
6. If it is equal to the last time, wait ½ of the period, else
7. Read the data in accordance with the command word count (1 to 32).
8. Remember the data pointer and go to stage 4 as long as required.

Since there are two buffers, and the core IP updates the data pointer at EOM, and the core IP automatically ping pongs between them, the host will always read consistent data. In worse case, the host reads the buffer when a new message is updated to the adjacent buffer.

### 4.3.3 BLOCK DATA TRANSFER- 2..5

For messages that transfer a block of data words that exceed the word count of a 1553 message, it is desired to receive the block of data WITHOUT data loss.

Such cases could be sensors which produce more than 32 data words structures, or SW loaders that want to flash a new version of SW on to a 1553 subsystem and alike.

A method to handle this type of data source is to allocate a large data buffer for that message in the IP core memory space. The IP core provides various data block sizes that can be defined per bus list entry.

Depending on the respondents of the Host SW and the rate at which the block is going to be written with 1553 messages, allocate a data buffer that is big enough to allow the SW to read data from the block before the 1553 messages overrides it.

Do the following steps:

1. Preset the memory buffer mode for this bus list entry to mode 2..5 (block buffer mode).
2. Allocate a data buffer for this message in the relevant size (256 to 16K words).
3. Store a pointer to this buffer to the data pointer in the bus list entry.
4. Keep a SW data pointer that point to the next unread word in the buffer. Should start with the same value as the stored pointer.
5. Wait for no more than 50% of the duration that the 1553 is expected to load the data using receive commands.
6. Check the data pointer in the bus list entry data pointer.
7. Read all new data from the SW data pointer location until the bus list entry data pointer value.
8. Update the new SW data pointer.
9. Repeat these steps as long as required.

Notice that the buffer is cyclic. Make sure the SW data pointer cycles in accordance with the buffer size mode selected.


17 Atir Yeda St., Kfar-Saba, ISRAEL 44643

Email: info@sitaltech.com

Website: http://www.sitaltech.com