**PROJECT REPORT**

on

# *"SOFTWARE SIMULATOR FOR INDIGENOUS MIL-STD-1553B PROTOCOL CONTROLLER ASIC"*

SUBMITTED FOR THE PARTIAL FULFILLMENT FOR THE DEGREE OF

**BACHELOR OF ENGINEERING**

in

**COMPUTER SCIENCE AND ENGINEERING**

by


**Mr. MOHD SHOZAB ABDUL SHAMIM**
**Mr. SHRITEJ ANIL LAKADE**
**Mr. SHEELAJ SURESH PATWARDHAN**
**Mr. PRASHIK RUPRAO GADGE**


UNDER THE GUIDANCE OF

Prof N. D. SHELOKAR


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SIPNA COLLEGE OF ENGINEERING AND TECHNOLOGY, AMRAVATI**

(AN ISO 9001:2015 CERTIFIED)

**NAAC ACCREDITED WITH GRADE "A+"**

**SANT GADGE BABA AMRAVATI UNIVERSITY, AMRAVATI.**


**2023-24**

भारत सरकार
अंतरिक्ष विभाग
विक्रम साराभाई अंतरिक्ष केंद्र
तिरुवनंतपुरम - 695 022, भारत
दूरभाष : 0471-2562444 / 2562555
फैक्स : 0471-2705345

Government of India
Department of Space
**Vikram Sarabhai Space Centre**
Thiruvananthapuram - 695 022, India
Phone : 0471-2562444 / 2562555
Fax : 0471-2705345

# FLIGHT PROCESSOR SYSTEM SOFTWARE DIVISION

## CERTIFICATE OF PROJECT WORK

This is to certify that the project report titled **"SOFTWARE SIMULATOR FOR INDIGENOUS MIL-STD-1553B PROTOCOL CONTROLLER ASIC"** is the authentic record of the project work done by **Mr. MOHD SHOZAB ABDUL SHAMIM** (Reg. No. VP20231211VS74) with **Mr. SHRITEJ ANIL LAKADE** (Reg. No. VP20231211VS73), **Mr. SHEELAJ SURESH PATWARDHAN** (Reg No. VP20231211VS75) and **Mr. PRASHIK RUPRAO GADGE** (Reg No. VP20231211VS76) under my supervision and guidance at **Vikram Sarabhai Space Centre, Thiruvananthapuram,** and submitted to the Department of Computer Science and Engineering, **Sipna College of Engineering and Technology, Amravati** in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

15/01/2024

Place: VSSC

Date: 15-01-2024

GOVERNMENT OF INDIA DEPARTMENT OF SPACE ★ VIKRAM SARABHAI SPACE CENTRE ★

Name: Shri. Syamlal L S

Designation: Sci/Engr SD

FPSD/FCG/AVN

# ACKNOWLEDGEMENT

We would like to express our profound gratitude to our mentor Shri. **SYAMLAL L S** of **FPSD/FCG/AVN** for the time and effort he provided throughout our project work duration. Your mentorship, useful advice and suggestions were really helpful to us during the project's completion. In this aspect, we are eternally grateful to you.

We like to express our special thanks to Smt. **ANJANA S J** of **FPSD/FCG/AVN** for her guidance to the completion of our project.

We would also like to extend our gratitude to **Vikram Sarabhai Space Centre, Thiruvananthapuram, Kerala** for providing us with this opportunity.

Additionally, we express our sincere thanks to **Prof. N. D. Shelokar**, our project guide in college along with **Dr. V. K. Shandilya**, Head of Department, Computer Science and the other staff members of the department for their kind co-operation.

We express our sincere thanks to **Dr. S. M. Kherde**, Principal, and **Shri. Jagdish Motilalji Gupta**, Chairman, Sipna College of Engineering & Technology, for their appreciation and kind words towards our team. We are also thankful for our friends and parents whose well wishes are always with us.

Lastly, we are grateful for the appreciation award given to us at the **Vidyotan 2k24**, for this project.

Mr. MOHD SHOZAB ABDUL SHAMIM
Mr. SHRITEJ ANIL LAKADE
Mr. SHEELAJ SURESH PATWARDHAN
Mr. PRASHIK RUPRAO GADGE
(Final Year CSE)

# ABSTRACT

MIL-STD-1553B is a data communication protocol published by the United States Department. It was originally designed as an avionic data bus for use with military avionics but has also become commonly used in spacecraft on-board data handling (OBDH) subsystems, both military and civil uses.

The MIL-STD-1553B protocol is implemented using ISRO's INDIGENOUS MIL-STD-1553B PROTOCOL CONTROLLER ASIC. In this project, we have created a simulator that behaves just like the physical controller ASIC. The purpose of this project was to create software that could be used to mimic the behaviour of the controller ASIC so that a large amount of test cases could be carried out on the simulator instead of physical hardware.

This project has been done for Flight Processor System Software Division in Indian Space Research Organization (VSSC), Thiruvananthapuram, Kerala as part of project work in the period of December 2023 to January 2024.

# TABLE OF CONTENTS

## LIST OF FIGURES

# CHAPTER 1

# Introduction

## 1.1 Introduction:

**MIL-STD-1553B** is a military standard defined by the U.S. Department of Defence that specifies the mechanical, electrical, and operational characteristics of a serial data transmission bus. It's a digital time division command/response multiplexed data bus. The 1553 data bus is a dual-redundant, bi-directional, Manchester II encoded data bus with a high bit error reliability.

The standard is organized similar to most military standards with a foreword, scope, referenced document section, definitions, general requirements, the appendix, and a tri-service Notice 2. It allows two types of data bus interface techniques; direct coupling and transformer coupling. Subsystems and 1553 bus elements are interfaced to the main data bus by interconnection buses (called "stubs").

All bus communications are controlled and initiated by a main bus controller. It's now widely used in avionics, aviation, and spacecraft data processing for the military as well as civil purposes.

# CHAPTER 2

## Literature Review

## 2.1 Literature Review:

### 2.1.1 MIL-STD 1553B Bus Protocol:

MIL-STD-1553B is a military standard that defines the method for the interconnection of digital electronic subsystems in aircraft, spacecraft, and other vehicles. It was originally developed by the United States Department of Defence and has become a widely accepted standard in the aerospace industry. This standard specifies the electrical and protocol characteristics for a data bus used for communication between different components within a vehicle. It employs a dual-redundant bus architecture, ensuring reliability and fault tolerance critical for military and aerospace applications.

It was originally designed as an avionic data bus for use with military avionics, but has also become commonly used in spacecraft on-board data handling (OBDH) subsystems, both military and civil, including use on the James Webb space telescope. It features multiple (commonly dual) redundant balanced line physical layers, a (differential) network interface, time-division multiplexing, half-duplex command/response protocol, and can handle up to 31 Remote Terminals (devices); 32 is typically designated for broadcast messages.

### 2.1.2 History:

MIL-STD-1553B is a set of rules for how electronic devices should talk to each other in military and aerospace applications. It all started in the late 1960s when the US Department of Defence realized that there needed to be a dependable and consistent way for electronic systems in planes, spacecraft, and military vehicles to communicate.

In the mid-1960s, the US Air Force, Navy, and NASA teamed up to create MIL-STD-1553. Their main aim was to create a common set of rules that could work across various military devices, ensuring that they could all understand each other and work together well. The first version, MIL-STD-1553A, was released in 1973. It laid out the basics for digital communication using a special type of wiring setup, making sure that data transfer would be reliable.

This standard has been crucial in ensuring that different military systems can work together smoothly by providing a common language for communication.

Later on, a more advanced version called MIL-STD-1553B was introduced in 1978. This update made the standard even better by refining the rules, improving how errors are detected and handled, and making it work for an even wider range of military and aerospace systems. As time passed, MIL-STD-1553B became popular not only in the US military but also in international aerospace and defence industries.

People liked it because it was strong, reliable, and could handle tough conditions. It became the top choice for making different parts of military aircraft, helicopters, spacecraft, missiles, and other defence systems talk to each other. Even though it's been around for many years, MIL-STD-1553B is still used in modern systems because it has a proven history of working well.

### 2.1.3 MIL-STD-1553B Architecture:

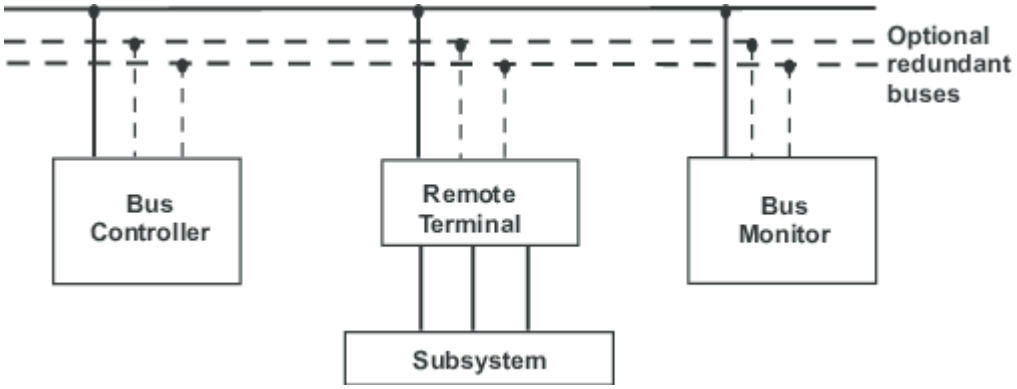



**Fig 2.1: Architecture of MIL-STD-1553B Protocol**

# CHAPTER 3

# Indigenous MIL-STD-1553B protocol controller ASIC

## 3.1 Indigenous MIL-STD-1553B protocol controller ASIC:

The Indigenous MIL-STD-1553B protocol controller ASIC is developed by the Indian Space Research Organization for the implementation of MIL-STD-1553 data communication protocol.

The Indigenous MIL-STD-1553B protocol controller ASIC provides a complete, flexible interface between a microprocessor and a MIL-STD-1553A, B Notice 2, McAir, or STANAG 3838 bus, implementing Bus Controller (BC), Remote Terminal (RT) and Monitor Terminal (MT) modes.

The protocol controller ASIC can provide advanced BC features like programmable gap times and programmable response time out. Along with this, the ASIC also provides the advanced RT feature of Double Buffering. The protocol can support various messaging formats like BC to RT, RT to BC, BC to RT's (Broadcast), RT to RT, RT to RT's (Broadcast), etc.

This protocol controller chip is able to detect various type of errors like the sync error, parity error, Manchester encoding error, status response timeout error, inter word gap error and loopback error.

The interrupt features provided by this protocol controller are as follows:

- End of message
- BC status set/ RT selected mode code interrupt
- BC end of frame
- Transmit time out, etc

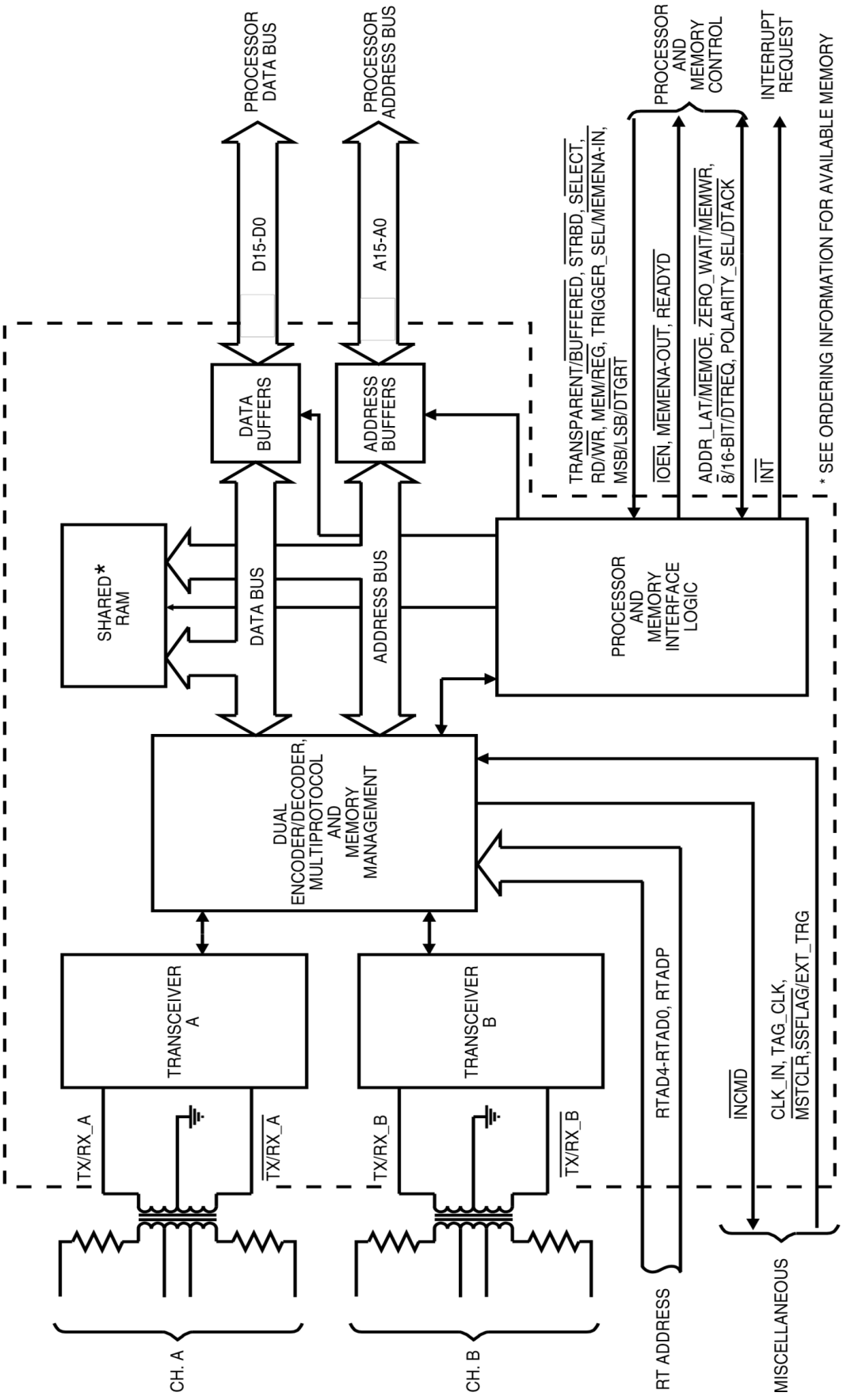



**Fig 3.1: Chipset architecture**

# CHAPTER 4

# Registers

## 4.1 Registers:

The software interface of the Indigenous MIL-STD-1553B protocol controller ASIC to the host processor consists of internal operational registers for normal operation, an additional 8 test registers, plus 64K x 16 of shared memory address space. The 4K x 16 of internal RAM resides in this address space.

The definition of the address mapping and accessibility for the 17 non-test registers, and the test registers, is as follows:

**4.1.1 Interrupt Mask Register** is used to enable and disable interrupt requests for various conditions.

**4.1.2 Configuration Registers #1** and **#2** are used to select the mode of operation and for software control of RT Status Word bits, Active Memory Area, BC Stop-on-Error, RT Memory Management mode selection, and control of the Time Tag operation.

**4.1.3 Start/Reset Register** is used for "command" type functions, such as software reset, BC/MT Start, Interrupt Reset, Time Tag Reset, and Time Tag Register Test. The Start/Reset Register includes provisions for stopping the BC in its auto-repeat mode, either at the end of the current message or at the end of the current BC frame.

**4.1.4 BC/RT Command Stack Pointer Register** allows the host CPU to determine the pointer location for the current or most recent message when the controller is in BC or RT modes.

**4.1.5 BC Control Word/RT Sub-address Control Word Register:** In BC mode, it allows host access to the current, or most recent BC Control Word. The BC Control Word contains bits that select the active bus and message format, enable off-line self-test, masking of Status Word bits, enable retries and interrupts, and specify MIL-STD-1553A or -1553B error handling. In RT mode, this register allows host access to the current or most recent Sub-address Control Word. The Sub-address Control Word is used to select the memory management scheme and enable interrupts for the current message. The read/write accessibility can be used as an aid for testing.

**4.1.6 Time Tag Register** maintains the value of a real-time clock. The resolution of this register is programmable from among 2, 4, 8, 16, 32, and 64 µs/LSB. The TAG_CLK input signal also may cause an external oscillator to clock the Time Tag Register. Start-of-message

(SOM) and End-of-message (EOM) sequences in BC, RT, and Message Monitor modes cause a write of the current value of the Time Tag Register to the stack area of RAM.

**4.1.7 Interrupt Status Register** mirrors the Interrupt Mask Register and contains a Master Interrupt bit. It allows the host processor to determine the cause of an interrupt request by means of a single READ operation.

**4.1.8 Configuration Registers #3, #4,** and **#5** are used to enable many of the advanced features. These include all the enhanced mode features; that is, all the functionality beyond that of the previous generation product, for all three modes, use of the Enhanced Mode enables the various read-only bits in Configuration Register #1. For BC mode, the enhanced mode features include the expanded BC Control Word and BC Block Status Word, additional Stop-On-Error and Stop-On-Status Set functions, frame auto-repeat, programmable inter-message gap times, automatic retries, expanded Status Word Masking, and the capability to generate interrupts following the completion of any selected message. For RT mode, the enhanced mode features include the expanded RT Block Status Word, the combined RT/Selective Message Monitor mode, internal wrapping of the RTFAIL output signal (from the J′ chip) to the RTFLAG RT Status Word bit, the double buffering scheme for individual receive (broadcast) sub-addresses and the alternate (fully software programmable) RT Status Word. For MT mode, use of the enhanced mode enables the use of the Selective Message Monitor, the combined RT/Selective Monitor modes, and the monitor triggering capability.

**4.1.9 Data Stack Address Register** is used to point to the current address location in shared RAM used for storing message words (second Command Words, Data Words, RT Status Words) in the Selective Word Monitor mode.

**4.1.10 Status Word Register** and **BIT Word Registers** provide read-only indications of the controller's RT Status and BIT Words.

**4.1.11 Time Tagging:**

The ASIC includes an internal read/writable Time Tag Register. This register is a CPU read/writable 16-bit counter with a programmable resolution of either 2, 4, 8, 16, 32, or 64 µs per LSB. Also, the Time Tag Register may be clocked from an external oscillator. Another option allows software-controlled incrementing of the Time Tag Register. This supports self-testing for the Time Tag Register. For each message processed, the value of the Time Tag

register is loaded into the second location of the respective descriptor stack entry (“TIME TAG WORD”) for both BC and RT modes.

Additional provided options will: clear the Time Tag Register following a Synchronize (without data) mode command or load the Time Tag Register following a Synchronize (with data) mode command; enable an interrupt request and a bit setting in the Interrupt Status Register when the Time Tag Register rolls over from 0000 to FFFF. Assuming the Time Tag Register is not loaded or reset, this will occur at approximately 4-second time intervals, for 64 µs/LSB resolution, down to 131 ms intervals, for 2 µs/LSB resolution.

Another programmable option for RT mode is the automatic clearing of the Service Request Status Word bit following the ASIC's response to a Transmit Vector Word mode command.

### 4.1.12 Interrupts:

The controller components provide many programmable options for interrupt generation and handling. Individual interrupts are enabled by the Interrupt Mask Register. The host processor may easily determine the cause of the interrupt by using the Interrupt Status Register. The Interrupt Status Register provides the current state of the interrupt conditions. The Interrupt Status Register may be updated in two ways. In the standard interrupt handling mode, a particular bit in the Interrupt Status Register will be updated only if the condition exists and the corresponding bit in the Interrupt Mask Register is enabled. In the enhanced interrupt handling mode, a particular bit in the Interrupt Status Register will be updated if the condition exists regardless of the contents of the corresponding Interrupt Mask Register bit. In any case, the respective Interrupt Mask Register bit enables an interrupt for a particular condition.

### 4.1.13 Address mapping:

| ADDRESS MAPPING | | | | | | |
|---|---|---|---|---|---|---|
| ADDRESS LINES | | | | | | REGISTER DESCRIPTION/ACCESSIBILITY |
| HEX | A4 | A3 | A2 | A1 | A0 | |
| 00 | 0 | 0 | 0 | 0 | 0 | Interrupt Mask Register (RD/WR) |
| 01 | 0 | 0 | 0 | 0 | 1 | Configuration Register #1 (RD/WR) |
| 02 | 0 | 0 | 0 | 1 | 0 | Configuration Register #2 (RD/WR) |
| 03 | 0 | 0 | 0 | 1 | 1 | Start/Reset Register (WR) |
| 03 | 0 | 0 | 0 | 1 | 1 | BC/RT Command Stack Pointer Register (RD) |
| 04 | 0 | 0 | 1 | 0 | 0 | BC Control Word*/RT Sub-address Control Word Register (RD/WR) |
| 05 | 0 | 0 | 1 | 0 | 1 | Time Tag Register (RD/WR) |
| 06 | 0 | 0 | 1 | 1 | 0 | Interrupt Status Register (RD) |
| 07 | 0 | 0 | 1 | 1 | 1 | Configuration Register #3 (RD/WR) |
| 08 | 0 | 1 | 0 | 0 | 0 | Configuration Register #4 (RD/WR) |
| 09 | 0 | 1 | 0 | 0 | 1 | Configuration Register #5 (RD/WR) |
| 0A | 0 | 1 | 0 | 1 | 0 | Data Stack Address Register (RD)* |
| 0B | 0 | 1 | 0 | 1 | 1 | BC Frame Time Remaining Register (RD)* |
| 0C | 0 | 1 | 1 | 0 | 0 | BC Time Remaining to Next Message Register (RD)* |
| 0D | 0 | 1 | 1 | 0 | 1 | BC Frame Time*/RT Last Command/MT Trigger Word* Register (RD/WR) |
| 0E | 0 | 1 | 1 | 1 | 0 | RT Status Word Register (RD) |
| 0F | 0 | 1 | 1 | 1 | 1 | RT BIT Word Register (RD) |

**Fig 4.1: Address Mapping of registers**

# CHAPTER 5

# Bus Controller

## 5.1 Bus Controller:

### 5.1.1 BC Memory Organization:

The below figure illustrates a typical memory map for BC mode. It is important to note that the only fixed locations for the controller in the Standard BC mode are for the two Stack Pointers (address locations 0100 (hex) and 0104) and for the two Message Count locations (0101 and 0105). The maximum size of a BC message block is 38 words, for an RT-to-RT Transfer of 32 Data Words (Control + 2 Commands + Loopback + 2 Status Words + 32 Data Words).

| BC MEMORY MAP | |
|---|---|
| **ADDRESS (HEX)** | **DESCRIPTION** |
| 0000 | Command Stack Pointer |
| 0001 | Message counter |
| 0002-0003 | Unused |
| 0004-01FF | Message Block Descriptor |
| 0200-0FFF | Message Blocks |

**Fig 5.1: Memory mapping of Bus Controller**

### 5.1.2 BC Memory Management:

The following figure illustrates the controller's BC memory management scheme. One of the BC memory management features is the global double buffering mechanism. This provides for two sets of the various BC mode data structures: Stack Pointer and Message Counter locations, Descriptor Stack areas, and BC message blocks.

The BC may be programmed to transmit multimessage frames of up to 512 messages. The number of messages to be processed is programmable by the Active Area Message Count location in the shared RAM, initialized by the host processor. In addition, the host processor

must initialize another location, the Active Area Stack Pointer. The Stack Pointer references the four-word message block descriptor in the Stack area of shared RAM for each message to be processed. The BC Stack size is programmable with choices of 256, 512, 1024, and 2048 words.

The third and fourth words of the BC block descriptor are the Intermessage Gap Time and the Message Block Address for the respective message. These two memory locations must be written by the host processor prior to the start of message processing. Use of the Intermessage Gap Time is optional. The Block Address pointer specifies the starting location for each message block. The first word of each BC message block is the BC Control Word.

At the start and end of each message, the Block Status and Time Tag Words write to the message block descriptor in the stack. The Block Status Word includes indications of message in process or message completion, bus channel, status set, response timeout, retry count, status address mismatch, loop test (on-line self-test) failure, and other error conditions. The 16-bit Time Tag Word will reflect the current contents of the internal Time Tag Register. This read/writable register, which operates for all three modes, has programmable resolution of from 2 to 64 μs/LSB. In addition, the Time Tag register may be clocked from an external source.

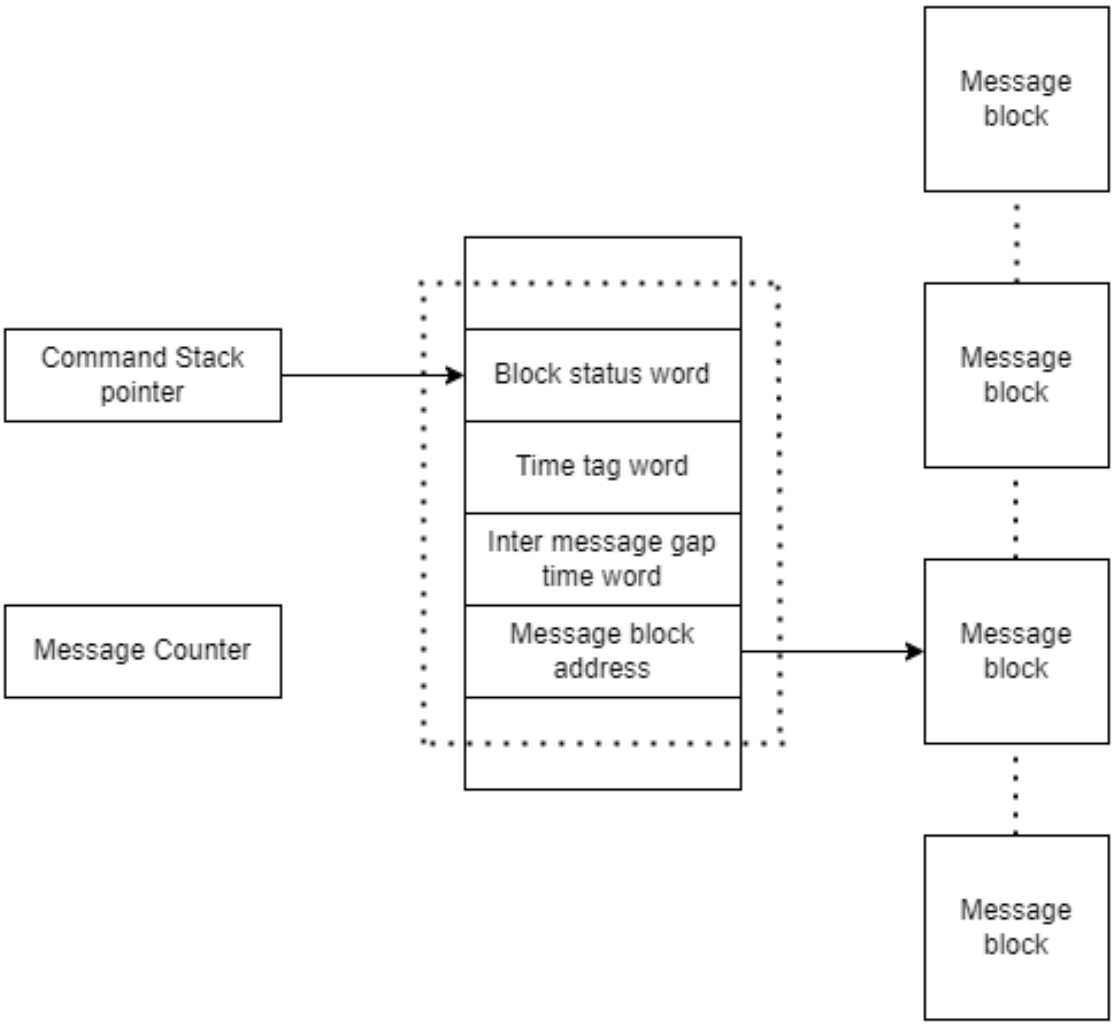



**Fig 5.2 Memory Organization for BC Mode**

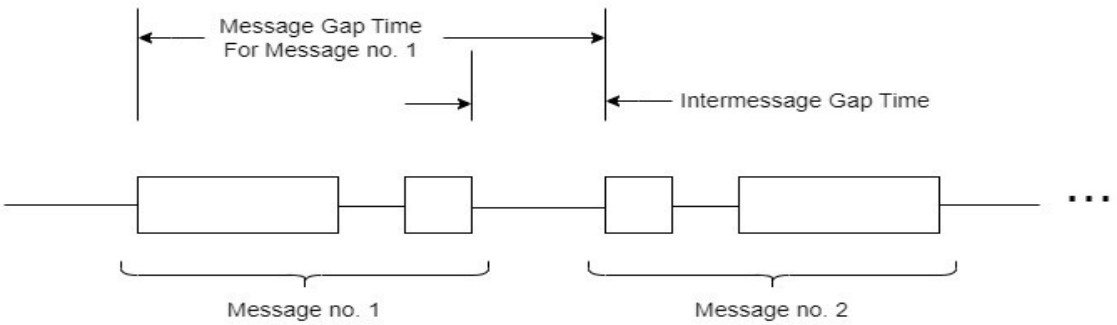



**Fig 5.3: BC Message Gap Timing**

# CHAPTER 6

# Remote Terminal

## 6.1 Remote Terminal:

### 6.1.1 RT Memory Organization:

The below figure illustrates a typical memory map for the controller in RT mode. As in BC mode, the two Stack Pointers reside in fixed locations in the shared RAM address space: address 0100 (hex) for the Area A Stack Pointer and address 0104 for the Area B

Stack Pointer. Besides the Stack Pointer, for RT mode there are several other areas of the ASIC address space designated as fixed locations. All RT modes of operation require the Area A and Area B Lookup Tables. Also allocated are several fixed locations for optional features: Command Illegalization Lookup Table, Mode Code Selective Interrupt Table, Mode Code Data Table, and Busy Bit Lookup Table. It should be noted that any unenabled optional fixed locations may be used for general purpose storage (data blocks).

| RT MEMORY MAP | |
|---|---|
| **ADDRESS (HEX)** | **DESCRIPTION** |
| 0000 | Command Stack Pointer |
| 0001-0003 | Unused |
| 0004-01FF | Message Block Descriptor |
| 0200-03FF | Transmit Buffer |
| 0400-05FF | Receive Buffer (1) |
| 0600-07FF | Receive Buffer (2) |
| 0800-0FFF | Unused |

**Fig 6.1: Memory mapping of Remote Terminal**

### 6.1.2 RT Memory Management:

One of the salient features of the ASIC is the flexibility of its RT memory management architecture. The RT architecture allows the memory management scheme for each transmit, receive, or broadcast subaddress to be programmable on a subaddress basis.

Also, in compliance with MIL-STD-1553B Notice 2, the controller provides an option to separate data received from broadcast messages from non-broadcast received data.

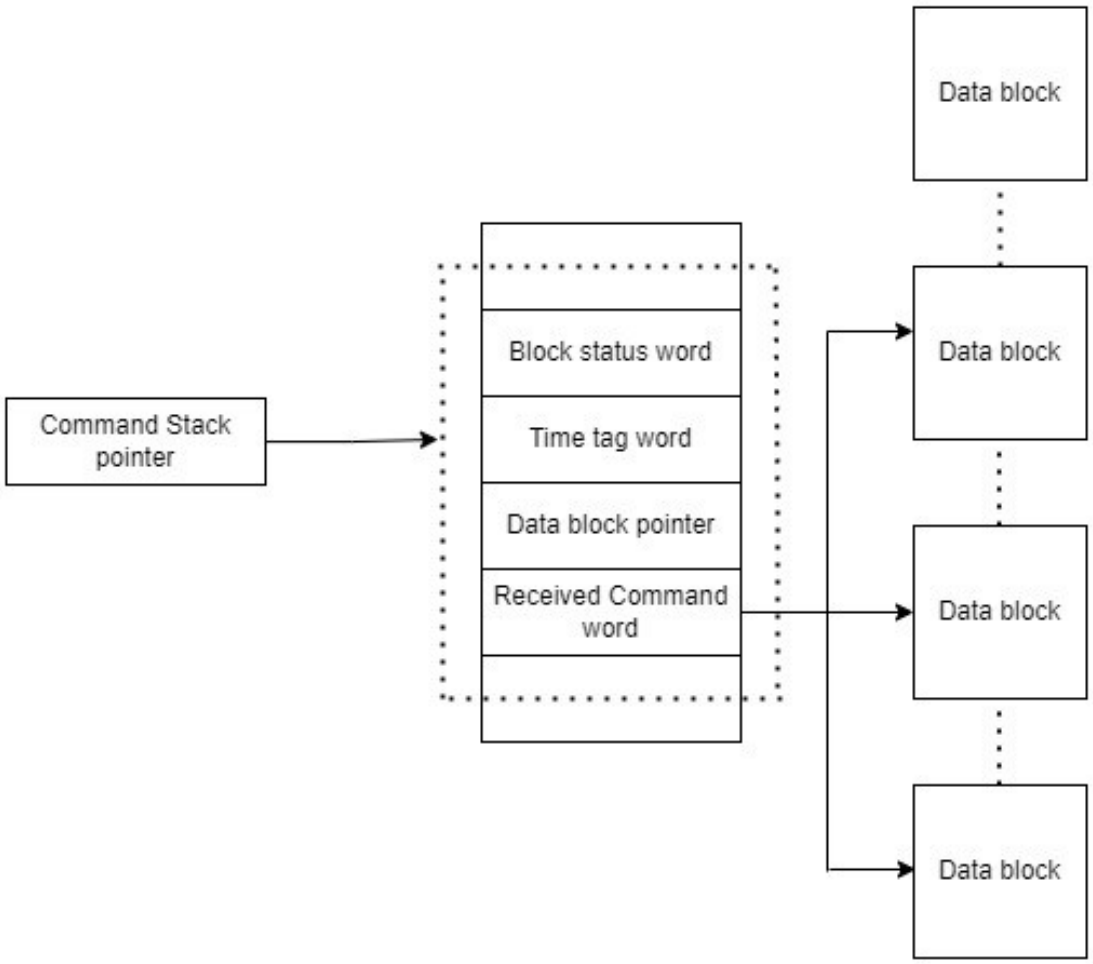



**Fig 6.2: Memory Organization for RT mode**

The structure of command word, data word, and status word is as follows:

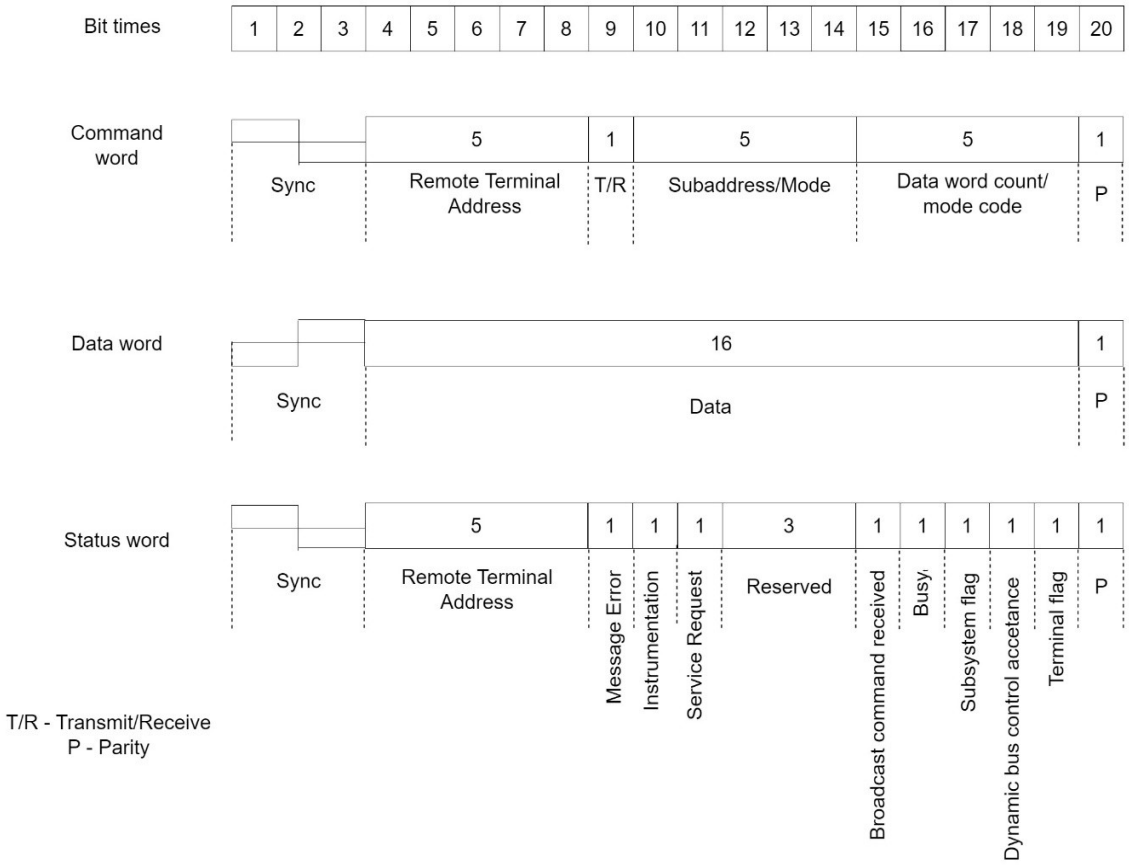



**Fig 6.3: Structure of Command word, Data word and Status word**

# CHAPTER 7

# Message formats

## 7.1 Message formats:

### 7.1.1 BC-TO-RT Message Format:

The command word is written to the bus for a specific RT. This RT gets the receive command from the Bus Controller. The transmit/receive bit of the command word is set as 0. The command word will be written at 4th word of the block descriptor block of the RT i.e. the Received Command Word. The block address pointer is incremented so that it points to the first data word in the message block. This data word is written to the bus. The receiving RT will receive this data word from the bus and write it in its respective buffer. This continues till the word counter indicates the number of data words is over. Then the looped back last word is written to the BC message block. Here the BC will wait for the status word from the RT. The status word for the RT is given by,

Status word = first word of the message descriptor block in RT i.e. Block Status Word.

| **BC-to-RT Transfer** |
|---|
| Control word |
| Receive Command Word |
| Data Word #1 |
| Data Word #2 |
| • • • |
| Last data word |
| Last data word looped back |
| Status Received |

**Fig 7.1: BC-to-RT Message Format**

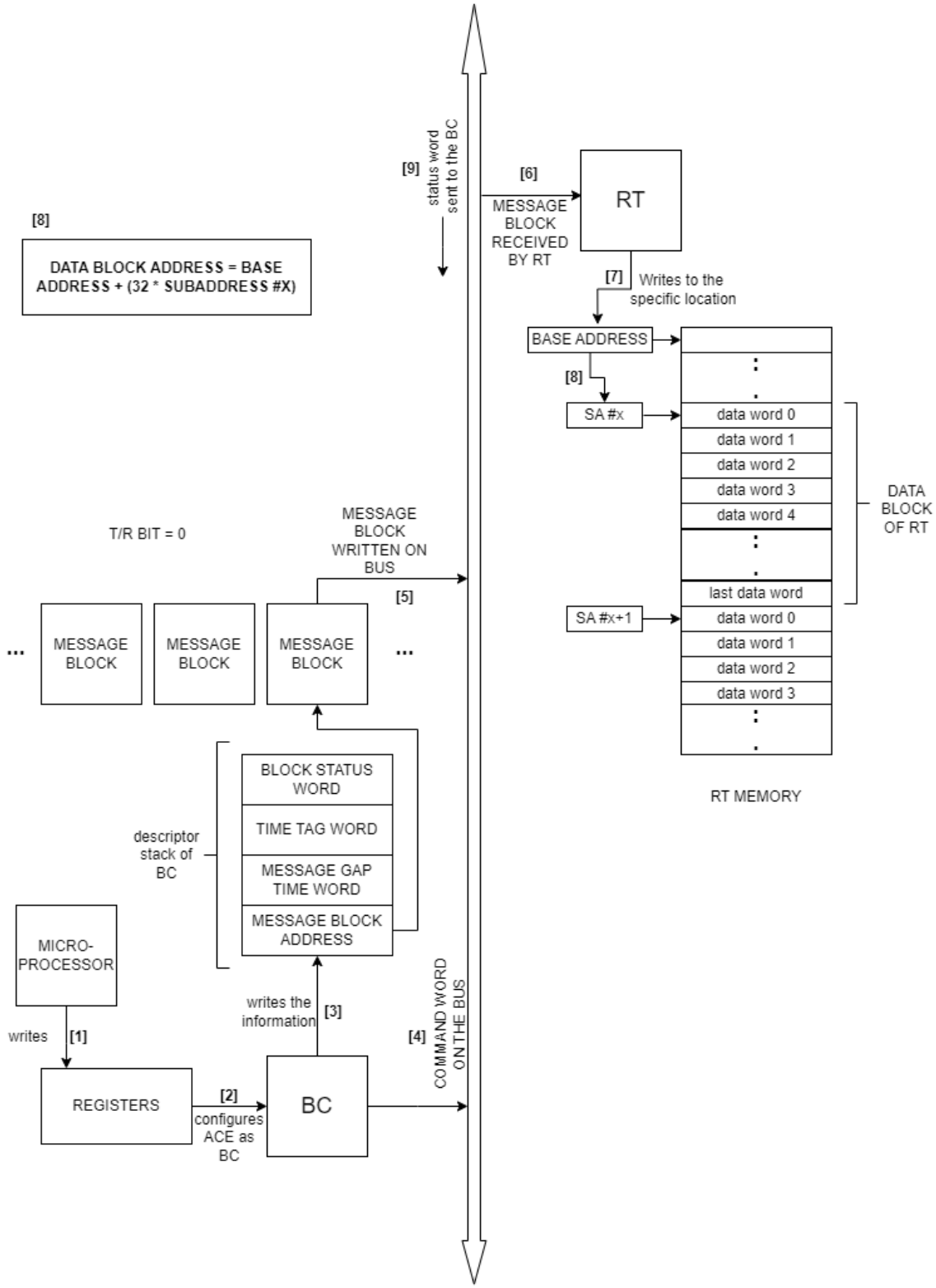



**Fig 7.2: BC-to-RT Message Transfer**

### 7.1.2 RT-TO-BC Message Format:

At first, the BC sends the command word on the bus. The T/R bit of the command word is set to 1 to indicate a transmit command to the RT. The address of the required remote terminal is mentioned in the command word itself.

After the required RT gets the transmit command, the RT sends the status word to the BC and the start of the message transfer begins.

The number of the data words needed to be sent is mentioned in the first 5 bits of the command word. After the status word is sent, the successive transfer of the data words begins.

| RT-to-BC Transfer |
|---|
| Control word |
| Transmit Command Word |
| Transmit Command Looped Back |
| Status Received |
| Data Word #1 |
| Data Word #2 |
| • • • |
| Last data word |

**Fig 7.3: RT-to-BC Message Format**

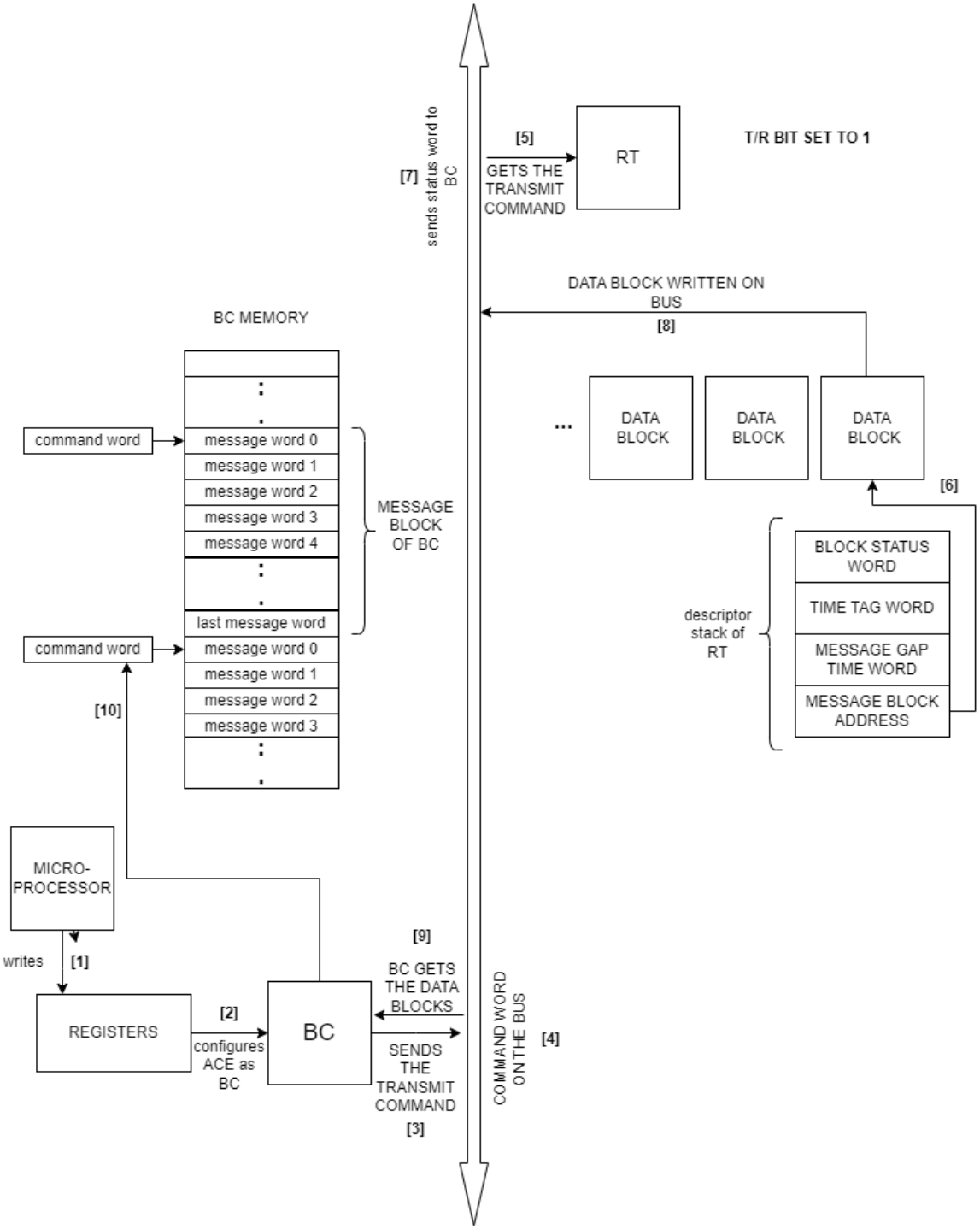



**Fig 7.4: RT-to-BC Message Transfer**

### 7.1.3 BROADCAST Message Format:

In the context of broadcast message transfer, the Bus Controller initiates communication by sending a broadcast command word over the bus. This command word typically contains information specifying that the subsequent data transfer is intended for all or a specific subset of RTs on the bus.

As the broadcast command is processed, successive data words are transmitted on the bus, and each RT interprets the information relevant to its designated subaddress. Unlike point-to-point communication, where a specific RT is addressed in the command word, broadcast messages allow multiple RTs to simultaneously receive and act upon the transmitted data.

In the broadcast command word, all the bits of the address field (bit 15 – bit 11) are set to 1.

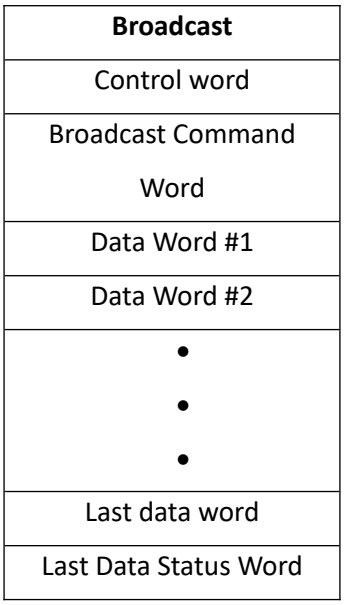



**Fig 7.5: BROADCAST Message Format**

The working of the broadcast message format is illustrated in the following figure:

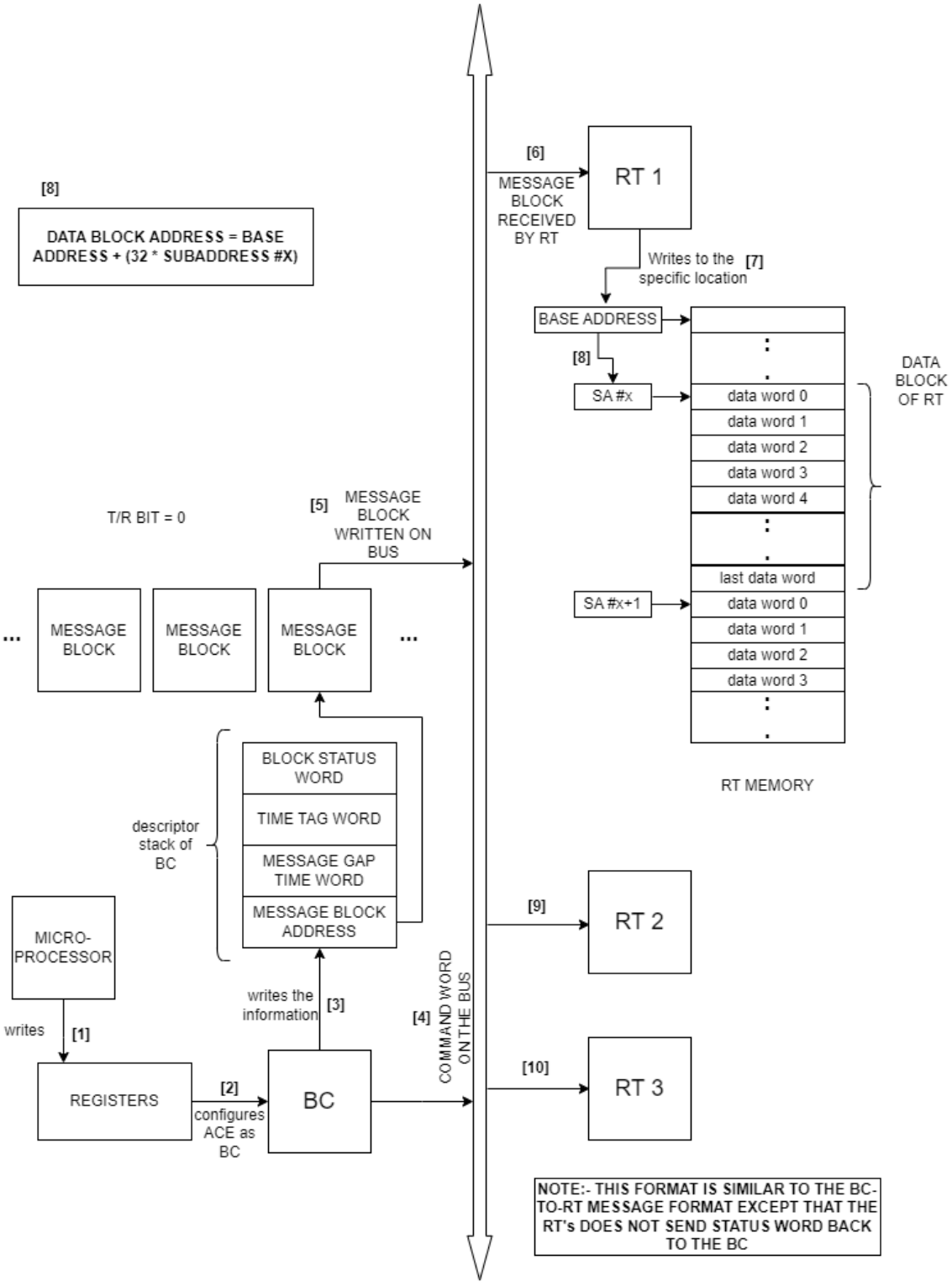



**Fig 7.6: BROADCAST Message Transfer**

# CHAPTER 8

## Software interface

## 8.1 Software interface:

For the software interface of the MIL-STD-1553B protocol, we require the following registers:

- Start Reset Register:

  The start reset register is used for command type functions, such as status word, inter message gap time resolution, start of operations in BC mode and also software reset operations.

- Configuration Registers:

  The configuration registers are used to select the ASIC's mode of operation and also various other functionalities like the mode selection, start on trigger, or stop on trigger.

- Block Status word:

  The block status word contains the bits that are used to denote the start of the message, end of the message, composite error, loop back error, etc.

- Inter Message Gap Time Word:

  The inter message gap time word is used to write the time of the start and end of the message. This word is updated at the start and the end of every message block.

# CHAPTER 9

## Simulator/ implementation

## 9.1 Simulator/ implementation:

- In the software implementation of the protocol, we created a class of the ASIC which can be configured as a Bus Controller or as a Remote Terminal by the values set in the registers. The registers will be configured by the microprocessor.
- The memory of the ASIC will also be set according to the BC or the RT.
- The mode of message format will be decided by the T/R bit of the Command word. If this bit is set to 1 then it indicates transmit, else it indicates receive. According to the message format, the transfer of data takes place.
- The command word will be used to decide the address, the Transmit/Receive mode of the message transfer, the subaddress of the respective Remote Terminal, and also the number of data words required to be sent.
- The first 5 bits (bit 11 – bit 15) of the command word are used to denote the Remote Terminal Address, the next bit i.e. bit 10 is used to represent the Transmit/Receive mode of the message transfer. The next 5 bits (bit 5 – bit 9) are used to represent the subaddress of the respective Remote Terminal. And last 5 bits are used for the data word count (bit 0 - bit 4).

The UML diagram for the software simulator is given as follows:

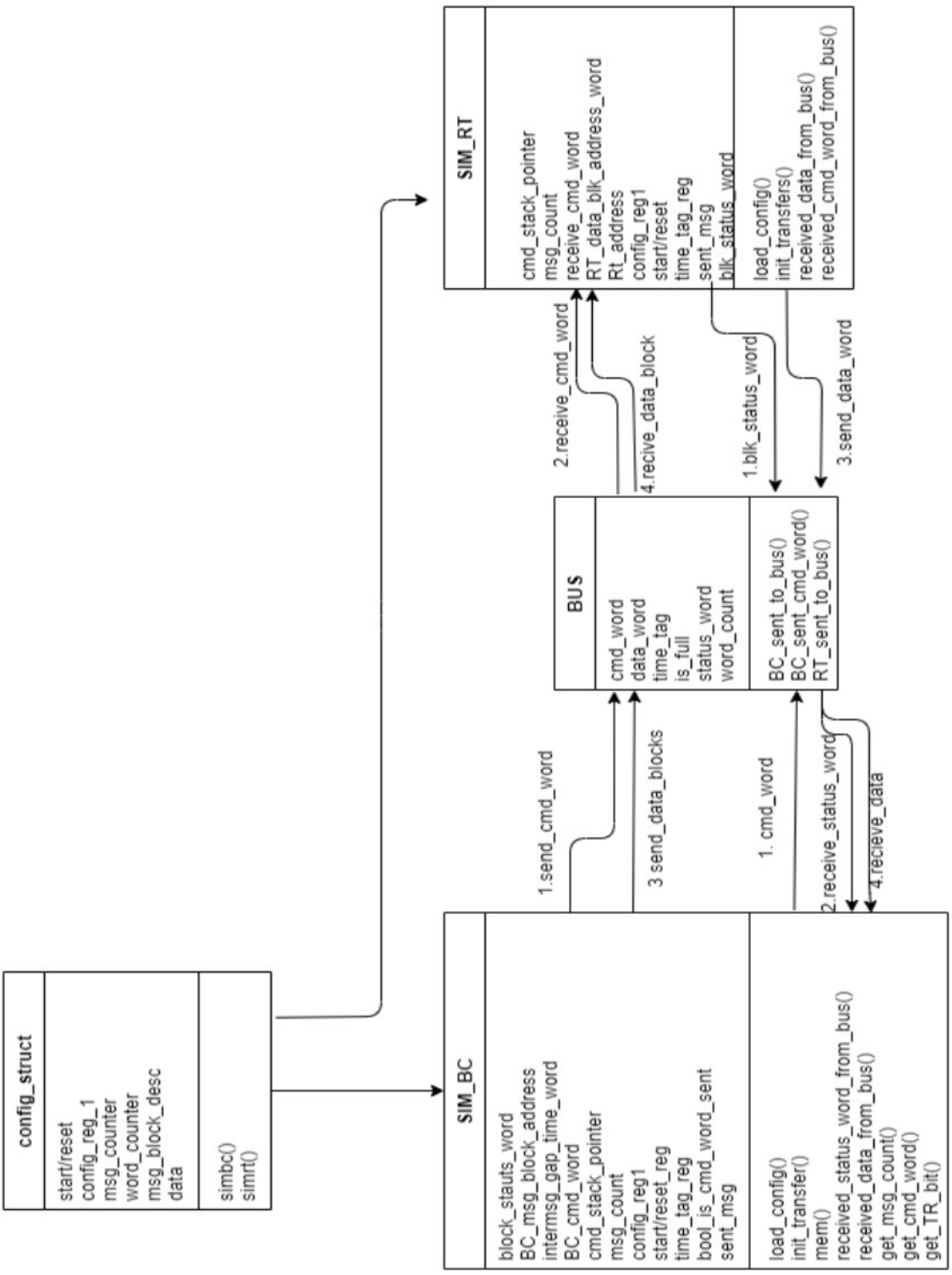



**Fig 9.1: UML diagram of the software simulator**

# CHAPTER 10

# Test Cases

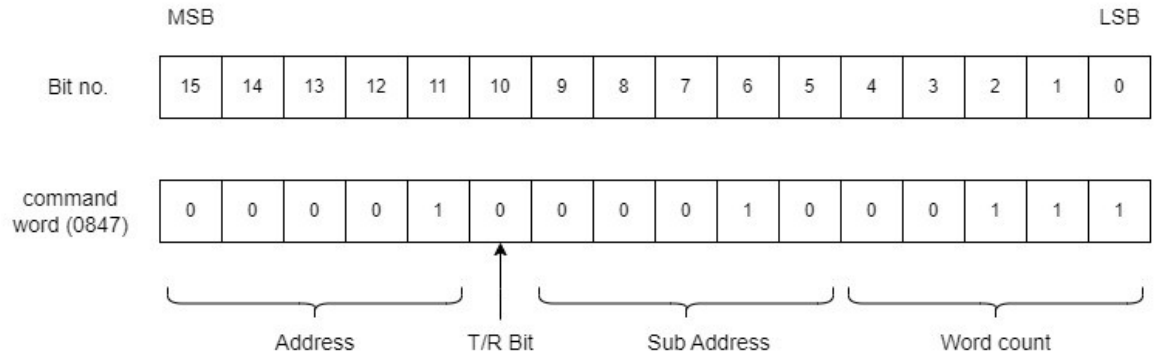
## 10.1 Test Cases:

### 10.1.1 Test Case 1: Single message BC-TO-RT transfer

Consider a BC, with the following configurations:

| ADDRESS | DATA | |
|---|---|---|
| 0004 | 4000 | Message Block Descriptor |
| 0005 | 0000 | |
| 0006 | E000 | |
| 0007 | 0200 | |
| : | : | |
| 0200 | 0847 | ← Command Word |
| 0201 | AAAA | Message Block |
| 0202 | BBBB | |
| 0203 | CCCC | |
| 0204 | DDDD | |
| 0205 | EEEE | |
| 0206 | FFFF | |
| 0207 | 1234 | |

When the BC is in INIT mode, the command word is analyzed and then it is written to the bus. The command word (0847) contains the following information:




From the above command word, we can say that the address of the Remote Terminal is 1, the T/R bit is set to 0 i.e. receive, the corresponding RT sub address is 2 and the number of data words to be sent is 7. The test case can be illustrated as follows:

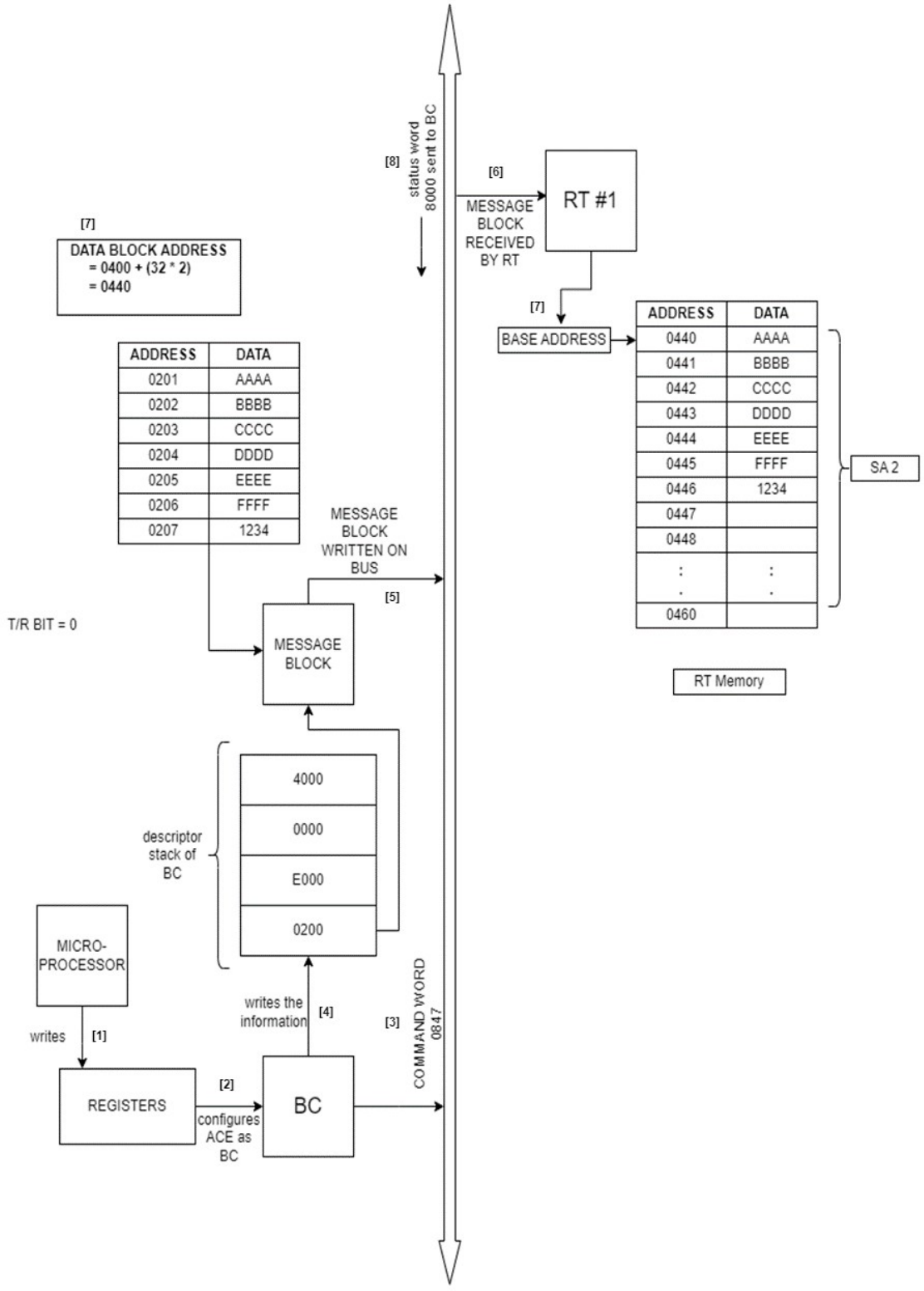



**Fig 10.1: Illustration of Test Case 1 (BC-TO-RT)**

**Terminal output:**

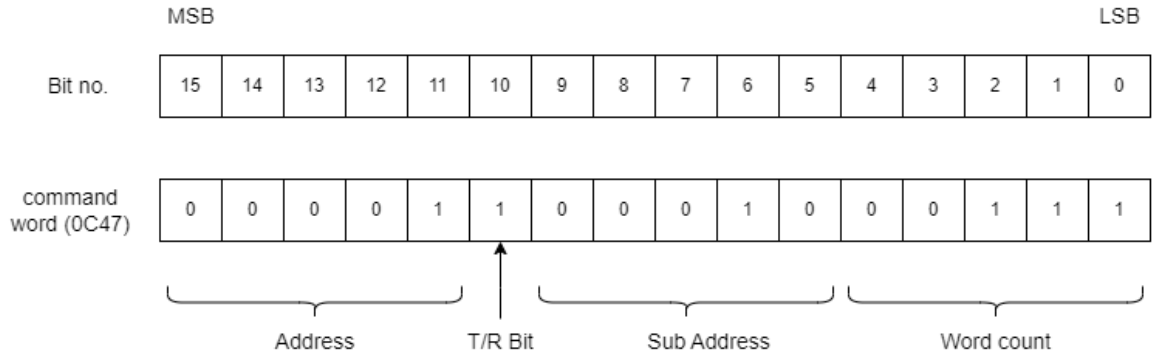
```
Begin sending message ##########################################
Writing to BUS------> Command Word-----> 847
Writing to BUS------> Address-----> 1
Writing to BUS------> Word Count-----> 7
Writing to BUS------> T R bit-----> 0
Writing to BUS------> Sub Address-----> 2

BC to RT transfer ---> Sending Data word 'AAAA' on receive buffer address 440 Time taken 20us
BC to RT transfer ---> Sending Data word 'BBBB' on receive buffer address 441 Time taken 40us
BC to RT transfer ---> Sending Data word 'CCCC' on receive buffer address 442 Time taken 60us
BC to RT transfer ---> Sending Data word 'DDDD' on receive buffer address 443 Time taken 80us
BC to RT transfer ---> Sending Data word 'EEEE' on receive buffer address 444 Time taken 100us
BC to RT transfer ---> Sending Data word 'FFFF' on receive buffer address 445 Time taken 120us
BC to RT transfer ---> Sending Data word '1234' on receive buffer address 446 Time taken 140us
End of message ##########################################
Block status word recieved---> 8000
Time Taken for msg 1: 140us
```

### 10.1.2 Test Case 2: Single message RT-TO-BC transfer

Consider a RT, with the following data

| ADDRESS | DATA |
|---|---|
| 0400 | 1111 |
| 0401 | 2222 |
| 0402 | 3333 |
| 0403 | 4444 |
| 0404 | 5555 |
| 0405 | 6666 |
| 0406 | 7777 |
| 0407 | 8888 |

When the BC is in INIT mode, the command word is analyzed and then it is written to the bus. The command word in BC (0C47) contains the following information:



From the command word we can say that, address of the Remote Terminal is 1, the T/R bit is set to 1 i.e. transmit, the corresponding RT sub address is 2 and the number of data words to be sent is 7.

The test case can be illustrated as follows:

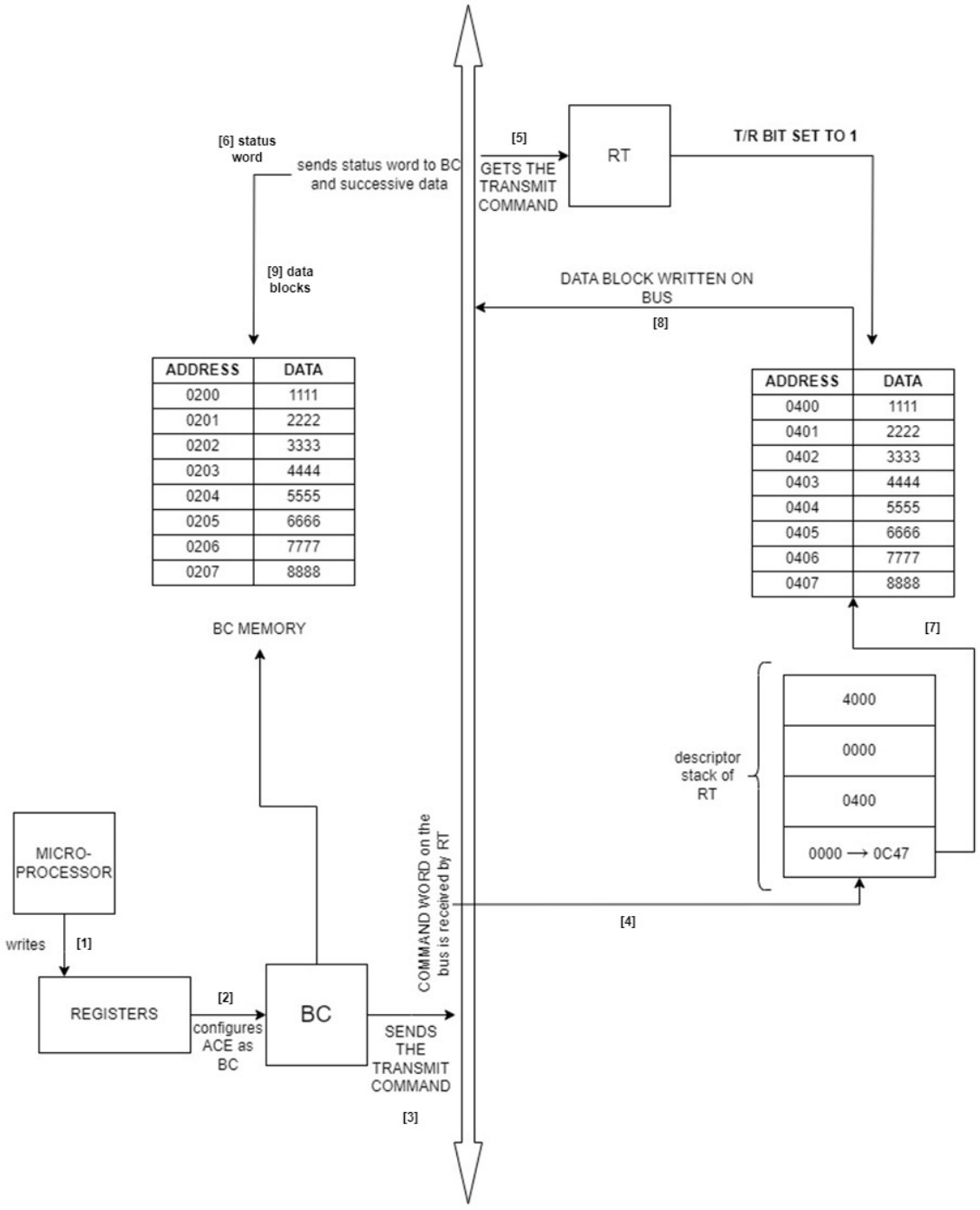



**Fig 10.2: Illustration of Test Case 2 (RT-TO-BC)**

**Terminal output:**

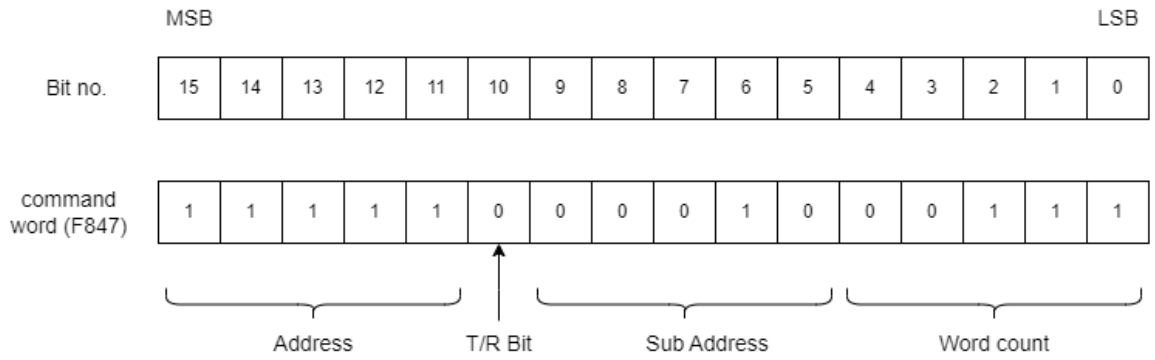
```
Begin sending message ##########################################
Command Word Recieved from bus---> C47
Status word Written to bus ---> 4000
RT to BC transfer ---> Receiving Data word 1111 on message block address 200 Time Taken 20us
RT to BC transfer ---> Receiving Data word 2222 on message block address 201 Time Taken 40us
RT to BC transfer ---> Receiving Data word 3333 on message block address 202 Time Taken 60us
RT to BC transfer ---> Receiving Data word 4444 on message block address 203 Time Taken 80us
RT to BC transfer ---> Receiving Data word 5555 on message block address 204 Time Taken 100us
RT to BC transfer ---> Receiving Data word 6666 on message block address 205 Time Taken 120us
RT to BC transfer ---> Receiving Data word 7777 on message block address 206 Time Taken 140us
RT to BC transfer ---> Receiving Data word 8888 on message block address 207 Time Taken 160us
End of message ##########################################
Time Taken for msg 1: 180us
```

### 10.1.3 Test Case 3: Single message BROADCAST Transfer

Consider a BC, with the following configurations:

| ADDRESS | DATA | |
|---|---|---|
| 0004 | 4000 | Message Block Descriptor |
| 0005 | 0000 | |
| 0006 | E000 | |
| 0007 | 0200 | |
| : | : | |
| 0200 | 0847 | Command Word |
| 0201 | AAAA | Message Block |
| 0202 | BBBB | |
| 0203 | CCCC | |
| 0204 | DDDD | |
| 0205 | EEEE | |
| 0206 | FFFF | |
| 0207 | 1234 | |

When the BC is in INIT mode, the command word is analyzed and then it is written to the bus. The command word (F847) contains the following information:



From the above command word, we can say that, address field contains 32, which is used to indicate broadcast message format, the T/R bit is set to 0 i.e. receive, the corresponding RT sub address is 2 and the number of data words to be sent is 7. The test case can be illustrated as follows:

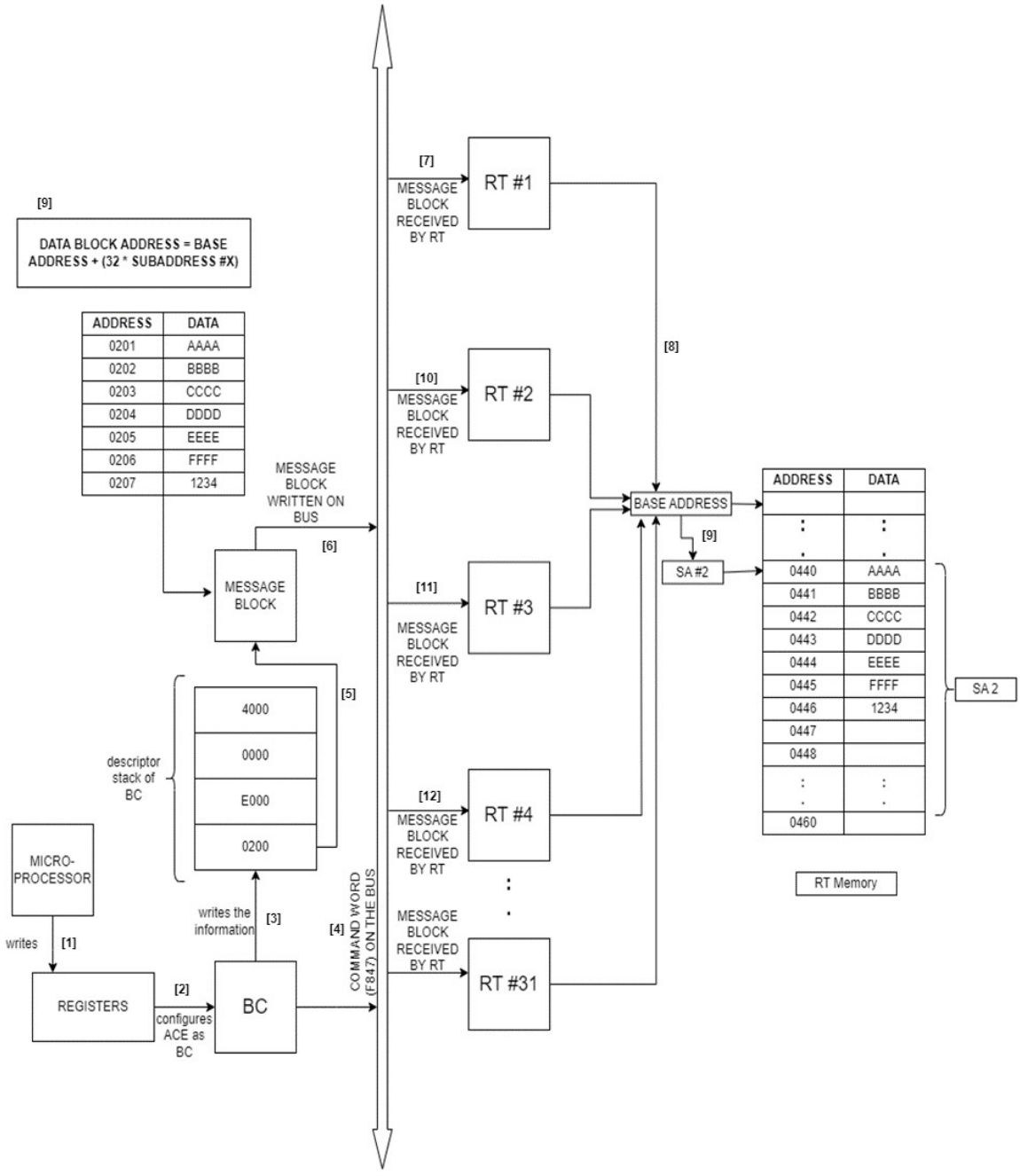



**Fig 10.3: Illustration of Test Case 3 (BROADCAST)**

**Terminal Output:**

```
Begin sending message ##########################################
Writing to BUS------> Command Word-----> F847
Writing to BUS------> Address-----> 1F
Writing to BUS------> Word Count-----> 7
Writing to BUS------> T R bit-----> 0
Writing to BUS------> Sub Address-----> 2

BC to RT transfer ---> Sending Data word 'AAAA' on receive buffer address 440 Time taken 20us
BC to RT transfer ---> Sending Data word 'BBBB' on receive buffer address 441 Time taken 40us
BC to RT transfer ---> Sending Data word 'CCCC' on receive buffer address 442 Time taken 60us
BC to RT transfer ---> Sending Data word 'DDDD' on receive buffer address 443 Time taken 80us
BC to RT transfer ---> Sending Data word 'EEEE' on receive buffer address 444 Time taken 100us
BC to RT transfer ---> Sending Data word 'FFFF' on receive buffer address 445 Time taken 120us
BC to RT transfer ---> Sending Data word '1234' on receive buffer address 446 Time taken 140us
End of message ##########################################
Time Taken for msg 1: 140us

RT 1 Recieved Data Words from BC
RT 2 Recieved Data Words from BC
RT 3 Recieved Data Words from BC
```

### 10.1.4 Test case 4: Multiple messages BC-TO-RT transfer:

Similar to test case 1, only difference is the number of messages sent is 3.

**Terminal Output:**

```
Begin sending message ##########################################
Writing to BUS------> Command Word-----> 847
Writing to BUS------> Address-----> 1
Writing to BUS------> Word Count-----> 7
Writing to BUS------> T R bit-----> 0
Writing to BUS------> Sub Address-----> 2

BC to RT transfer ---> Sending Data word 'AAAA' on receive buffer address 440 Time taken 20us
BC to RT transfer ---> Sending Data word 'BBBB' on receive buffer address 441 Time taken 40us
BC to RT transfer ---> Sending Data word 'CCCC' on receive buffer address 442 Time taken 60us
BC to RT transfer ---> Sending Data word 'DDDD' on receive buffer address 443 Time taken 80us
BC to RT transfer ---> Sending Data word 'EEEE' on receive buffer address 444 Time taken 100us
BC to RT transfer ---> Sending Data word 'FFFF' on receive buffer address 445 Time taken 120us
BC to RT transfer ---> Sending Data word '1234' on receive buffer address 446 Time taken 140us
End of message ##########################################
Block status word recieved---> 8000
Time Taken for msg 1: 140us
```

```
Begin sending message ##########################################
Writing to BUS------> Command Word-----> 867
Writing to BUS------> Address-----> 1
Writing to BUS------> Word Count-----> 7
Writing to BUS------> T R bit-----> 0
Writing to BUS------> Sub Address-----> 3

BC to RT transfer ---> Sending Data word '8888' on receive buffer address 460 Time taken 180us
BC to RT transfer ---> Sending Data word '3333' on receive buffer address 461 Time taken 200us
BC to RT transfer ---> Sending Data word '1234' on receive buffer address 462 Time taken 220us
BC to RT transfer ---> Sending Data word '1234' on receive buffer address 463 Time taken 240us
BC to RT transfer ---> Sending Data word '1234' on receive buffer address 464 Time taken 260us
BC to RT transfer ---> Sending Data word '9999' on receive buffer address 465 Time taken 280us
BC to RT transfer ---> Sending Data word '8888' on receive buffer address 466 Time taken 300us
End of message ##########################################
Block status word recieved---> 8000
Time Taken for msg 2: 300us
```

```
Begin sending message ##########################################
Writing to BUS------> Command Word-----> 887
Writing to BUS------> Address-----> 1
Writing to BUS------> Word Count-----> 7
Writing to BUS------> T R bit-----> 0
Writing to BUS------> Sub Address-----> 4

BC to RT transfer ---> Sending Data word '2222' on receive buffer address 480 Time taken 340us
BC to RT transfer ---> Sending Data word '3333' on receive buffer address 481 Time taken 360us
BC to RT transfer ---> Sending Data word '4444' on receive buffer address 482 Time taken 380us
BC to RT transfer ---> Sending Data word '5555' on receive buffer address 483 Time taken 400us
BC to RT transfer ---> Sending Data word '6666' on receive buffer address 484 Time taken 420us
BC to RT transfer ---> Sending Data word '7777' on receive buffer address 485 Time taken 440us
BC to RT transfer ---> Sending Data word '8888' on receive buffer address 486 Time taken 460us
End of message ##########################################
Block status word recieved---> 8000
Time Taken for msg 3: 460us
```

### 10.1.5 Test case 5: Two different message formats (BC-TO-RT & RT-TO-BC)

Similar to test case 1 and test case 2

**Terminal Output:**

```
Begin sending message ##########################################
Command Word Recieved from bus---> C47
Status word Written to bus ---> 4000
RT to BC transfer ---> Receiving Data word 1111 on message block address 200 Time Taken 20us
RT to BC transfer ---> Receiving Data word 2222 on message block address 201 Time Taken 40us
RT to BC transfer ---> Receiving Data word 3333 on message block address 202 Time Taken 60us
RT to BC transfer ---> Receiving Data word 4444 on message block address 203 Time Taken 80us
RT to BC transfer ---> Receiving Data word 5555 on message block address 204 Time Taken 100us
RT to BC transfer ---> Receiving Data word 6666 on message block address 205 Time Taken 120us
RT to BC transfer ---> Receiving Data word 7777 on message block address 206 Time Taken 140us
RT to BC transfer ---> Receiving Data word 8888 on message block address 207 Time Taken 160us
End of message ##########################################
Time Taken for msg 1: 180us

Begin sending message ##########################################
Writing to BUS------> Command Word-----> 847
Writing to BUS------> Address-----> 1
Writing to BUS------> Word Count-----> 7
Writing to BUS------> T R bit-----> 0
Writing to BUS------> Sub Address-----> 2

BC to RT transfer ---> Sending Data word '8888' on receive buffer address 440 Time taken 200us
BC to RT transfer ---> Sending Data word '3333' on receive buffer address 441 Time taken 220us
BC to RT transfer ---> Sending Data word '1234' on receive buffer address 442 Time taken 240us
BC to RT transfer ---> Sending Data word '1234' on receive buffer address 443 Time taken 260us
BC to RT transfer ---> Sending Data word '1234' on receive buffer address 444 Time taken 280us
BC to RT transfer ---> Sending Data word '9999' on receive buffer address 445 Time taken 300us
BC to RT transfer ---> Sending Data word '8888' on receive buffer address 446 Time taken 320us
End of message ##########################################
Block status word recieved---> 8000
Time Taken for msg 1: 320us


PS C:\Users\SHozabROG\Desktop\ISRO Project\Final Code>
```

# CHAPTER 11

# Conclusion

## 11.1 Conclusion:

This report is developed as part of the B.E Final Year Project Work in Vikram Sarabhai Space Centre, Thiruvananthapuram, Kerala. During the period of Dec 2023 to Jan 2024.

While we have simulated the behaviour of Indigenous MIL-STD-1553B protocol controller ASIC, the project does not accurately represent the internal working of ASIC. We have taken physical layer components completely for granted and we have ignored many of the internal registers and memory areas to employ the data transfer method in the MIL-STD-1553B protocol.

We have considered only the bus controller and remote terminal as the protocol components and have only employed BC to RT, RT to BC, and Broadcast message format.

# CHAPTER 12

# Bibliography

## 12.1 Bibliography:

[1] National Aeronautics and Space Administration, USA. (1978, September 21). MILITARY STANDARD AIRCRAFT INTERNAL TIME DIVISION COMMAND/RESPONSE MULTIPLEX DATA BUS. NASA. https://nepp.nasa.gov/docuploads/43745C0A-323E-4346-A434F4342178CD0E/MIL-STD-1553.pdf

[2] Wikipedia contributors. (2024, April 25). *MIL-STD-1553*. Wikipedia. https://en.wikipedia.org/wiki/MIL-STD-1553

[3] Data Device Corporation, USA. (2012, January 12). MIL-STD-1553. DDC. https://www.milstd1553.com/wp-content/uploads/2012/12/MIL-STD-1553B.pdf

[4] Mil-STD-1553. (n.d.). https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Onboard_Computers_and_Data_Handling/Mil-STD-1553

[5] Data Device Corporation, USA. (n.d.). ACE Complete Integrated BC-RT-MT Terminal. https://www.ddc-web.com/en/connectivity/databus/milstd1553-1/components-1/bu-65170-bu-65171-bu-61580-bu-61581-bu-61585-bu-61586?partNumber=BU-65170,%20BU-65171,%20BU-61580,%20BU-61581,%20BU-61585,%20BU-61586

[6] K. Sushma, C. D. Naidu, Y. P. Sai and K. S. Chandra, "Design and Implementation of High Performance MIL-STD-1553B Bus Controller," 2017 IEEE 7th International Advance Computing Conference (IACC), Hyderabad, India, 2017, pp. 266-269, doi: 10.1109/IACC.2017.0065. keywords: {Military standards;Aerospace electronics;Military aircraft;Monitoring;Hardware design languages;Field programmable gate arrays;MIL-STD-1553B;Bus Controller (BC);Verilog;ModelSim;Design Compiler;FPGA}