SITAL Technology

MIL-STD-1553 & PP194 or H009 Bus Tester & Analyzer Hardware Software Interface & User's Manual

MultiComBox

Bus Tester & Error injections
Multiple RT &
Word Monitor
4 ports of RS422/485

29 July 2015 – Injected data now for simulated RTs only
20 Jan 2016 – added Sital Smart Wiring block
8 Nov 2016 – updated to version 9.0 with Cyber Attack Emulation – see register 4B
25 Jan 2017 – Added RS485 Tx tail measure for detection of missing termination
March 2018 – Read and write of time tag counter

www.sitaltech.com

The information provided in this User's Guide is believed to be accurate; however, no responsibility is assumed by Sital Technology for its use, and no license or rights are granted by implication or otherwise in connection therewith. Specifications are subject to change without notice.

Table of Contents

Introduction

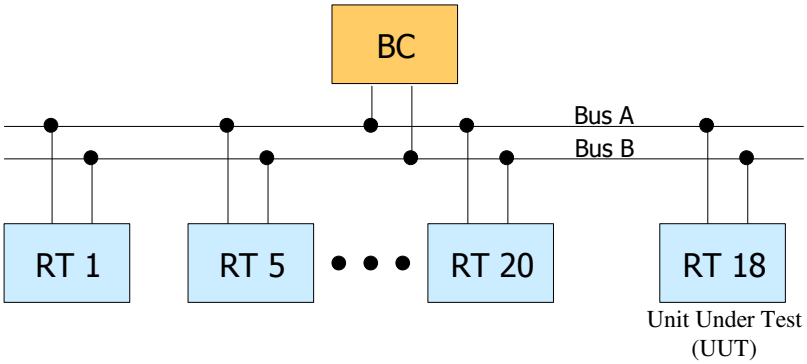
The TestersChoice bus tester is a compact design that is intended for MIL-STD-1553B (1553) and 16PP194 (194) bus testing of Line Replacement Units (LRUs) that communicate over standard 194 and 1553 busses.

The TestersChoice incorporates a Bus Controller (BC) in addition to simulating Multiple Remote Terminals (Multi RT) or Simulate Multiple RTs as a stand-alone mode.

Multi RT is required for testers to simulate those RTs that exist in the simulated system. The Multi RT can simulate 0 to 31 RTs based on user programming.

A unique Live-RT mode changes all RTs that are not simulated to Live RTs. Live RTs reply to the TestersChoice transmit commands with data payload that changes over time. All receive commands are replied with status words according to the standard.

A typical operational system would look like:



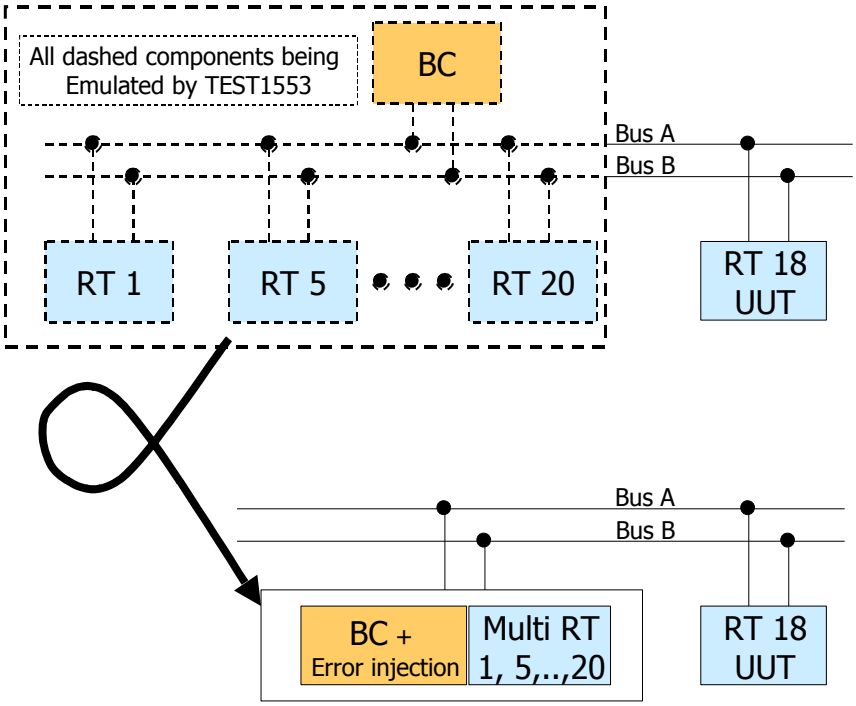
To-Be-Simulated system block Diagram

Single Bus Controller with multiple Remote Terminals

Block Diagram

When building a simulation environment for the Unit Under Test (UUT), and assuming that the Interconnect Control Document (ICD) includes communications between the RTs themselves, a tester would need to simulate both the BC and all of the RTs that communicate with the UUT RT.

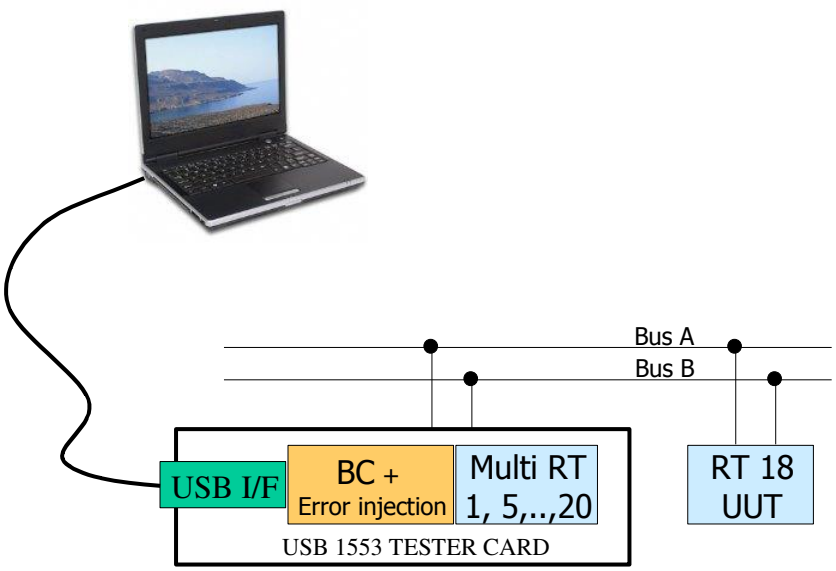
The same system in simulation would thus look like in the following diagram:



The TestersChoice design can be incorporated in a tester card. The tester card is being controlled by software in a host computer. PCI, USB among other communication means can serve as the connection between the tester card and the host PC. Typically, the card is supplied with SW drivers. The SW driver relieves the programmer from the need to master the memory and register mapping that the BC and Multi RT need for their continuous operation.

The TestersChoice design also incorporates a word monitor. This word monitor is like a 1553 scope, allowing the host a good view of what words ran on the bus. The word Monitor Monitors everything including words sent by TestersChoice.

A block diagram that includes a sample USB interface and the host computer can be found in the following block diagram.



USB card example

16PP194 extension

The PP194 standard is a weapon bus used on F16s. The PP194 standard uses the same physical level electric characteristics as the 1553 standard. The protocol is different. The PP194 bus is built of twice the number of wires as the 1553 bus. There is the Tx bus, which output from the CIU (BC of PP194) and the Rx bus, which is input to CIU and output from the RIUs (RTs). The RIUs reply on the Rx bus and listen to the Tx bus. There is full redundancy with a backup bus.

In latest F16s, MIL-STD-1553, or better MIL-STD-1760 (Weapons version of 1553), has been added on the 194 Rx bus. This was done to allow the usage of advanced weapons needing more data than provided by the older PP194 standard.
This creates a unique situation that two different communication standards share the same physical media and do not disturb each other. It also challenges the ability of the Stores system to manage a bus with two concurrent message formats in the same frame.

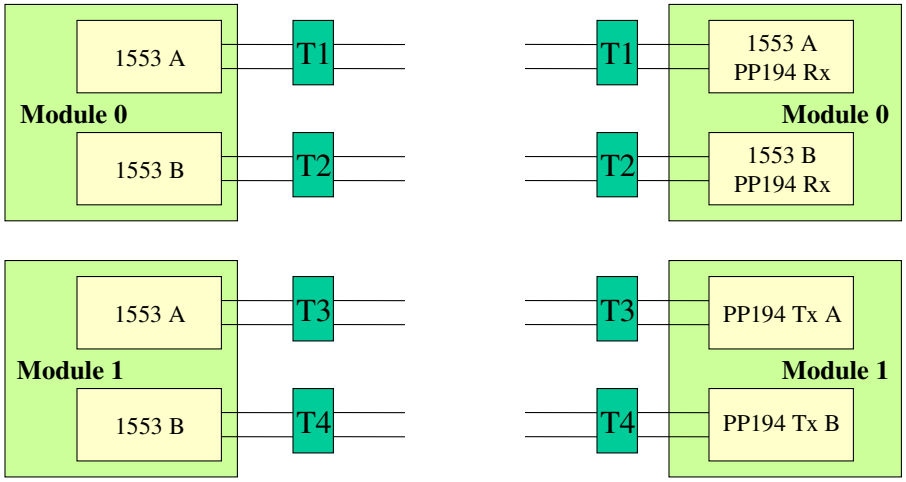
TestersChoice was designed to operate as a BC with each message being tagged as PP194 or 1553. The word monitor also supports 1553 and PP194.

PP194 MultiRIU stand-alone mode

In MultiRIU mode (User code 2) the MultiComBox can only specify data time 4 status word on the Rx bus.
The data transmitted at that time is the same data that was sent on the Tx bus at Data time 3.

Physical Bus assignments

TestersChoice supports two modules each with a single full duplex 1553 bus (4 twisted pair lines). These two 1553 modules support a single 194 bus. Module 0 is the Rx bus and its backup, and Module 1 accommodates the Tx bus.



1553 Configuration **1553/194 Configuration**

TestersChoice in the 1553/194 configuration would work as in 1553 mode, i.e., two busses in two modules. If one of the frame messages is defined as a 194 message (see Message Control bit 2) it will use both modules. The Data or Go/NoGo command will be transmitted on module 1 (T3 and T4), and the data echo or status will be transmitted on module 0. All 1553 messages in the frame would be transmitted on module 0 (T1 and T2).

Memory space

The full 64Kx16 bit memory is arranged in blocks of 8K by 16 bits blocks.

Size	range	Usage
8K	0x0000 to 0x1FFF	Module 0 1553 / PP194 / H009 bus tester or MultiRT.
8K	0x2000 to 0x3FFF	Module 0 1553 / PP194 word monitor. In card from version 6: the memory is external and 256Kx16.
8K	0x4000 to 0x5FFF	Module 1 1553 / PP194 / H009 bus tester or MultiRT.
8K	0x6000 to 0x7FFF	Module 1 1553 / PP194 word monitor. In card from version 6: the memory is external and 256Kx16.
256	0x8000 to 0x80FF	Module 0 - Sital Smart Wiring block - see appendix F
8K	0x8100 to 0x9FFF	Future use.
1K	0xA000 to 0xA3FF	Module 0 RS422 line 0 transmit
1K	0xA800 to 0xABFF	Module 0 RS422 line 0 receive
1K	0xB000 to 0xB3FF	Module 0 RS422 line 1 transmit
1K	0xB800 to 0xBBFF	Module 0 RS422 line 1 receive
8k	0xC000 to 0xDFFF	Future use.
1K	0xE000 to 0xE3FF	Module 1 RS422 line 0 transmit
1K	0xE800 to 0xEBFF	Module 1 RS422 line 0 receive
1K	0xF000 to 0xF3FF	Module 1 RS422 line 1 transmit
1K	0xF800 to 0xFBFF	Module 1 RS422 line 1 receive

BC tester interface

The Host interfaces with the BC by updating a shared memory and a set of registers. The TestersChoice works with the clock frequency that is used by the host thus the update is being done with a HW synchronous method eliminating the chance of partial update of memory or registers. The memory and register structure and contents is defined in the following sections.

MultiRT Modes

TestersChoice has two modes of operation for the MultiRT. One is BC\MultiRT, and the other is MultiRT standalone, or simply MultiRT. For the MultiRT standalone, please refer to applicable section. Following is a description for the BC\MultiRT where the BC state engine can also simulate RTs that are not physically attached to the bus.

BC\Multi RT mode

The only definition required for the Multi RT is a 31 bit long register that enables the relevant RTs. If the Host sets bit X of this register high, TestersChoice will simulate this RT. If this bit is set low, Multi RT will not respond to commands sent to that RT. Valid values for X is 0 to 30.
For simulated RTs transmitting data, the BC message block should be loaded with the data words and status words that the simulated RT will transmit, as explained in the following sections.
For simulated RTs receiving a message, the message block will be loaded with the data and status words that the simulated RT received.
RT to RT (RT2RT) transactions are supported by the TestersChoice. Both the transmitting and-or the receiving RTs, can be simulated.
Bit (RT) 31 represents the broadcast address in 1553 standard. If a broadcast command issued by the BC, the Multi RT will not perform any special action since Broadcast does not require any RT response. Therefore, when bit 31 (BCST address) of that register is set high, Live RT mode is enabled. No real RTs should be connected to the tester in this mode. All NON-Simulated RTs become Live RTs. Live RTs' transmitted payload increments over time and is used to verify proper system operation when operating in stand alone without real RTs connected.

For PP194 there are up to 15 RIUs, and a reset RIU (0). A vector of simulated RIUs should be similarly supplied for the RIU simulation.

MultiRT standalone mode

This mode is used when a BC exists on the bus outside TestersChoice, and TestersChoice is required to reply to its commands for several RTs.
This mode of operation is designed to act with as much similarity in SW control as the BC\MultiRT mode.
To enter this mode, it is required to set the User Code option to 2 (bit 1 set high, other bits don't care). After mode 2 is set, the HW responds differently to SW commands, especially BC Go command will not initiate transmission, but rather enable the MultiRT standalone state machine to start replying to bus commands.
For this mode, the SW loads a stack of commands and a 31-bit register that defines which RTs are to be simulated.

When a command is detected on the bus, the MultiRT state machine checks if the RT referred by the command is enabled for simulation. If it is not, the state machine will continue to monitor the message but will not do anything else.
If it is to be simulated, the MultiRT state machine scans all of the "BC" stack entries for the length defined by stack length, for a machine command. If found, the MultiRT state machine starts replying to the message with the status words and data words pointed by the stack entry. This allows the SW to setup the data and status contents that the simulated RT is required to reply.
If the scanning for the command fails, but the RT has to be simulated, the MultiRT will reply to the command with clear status (only RT address set) and with data words with value picked up from address 0 to 0x20 of the shared memory (preset to all zeros).

In real time systems, the SW would want to replace the data on every frame. This capability exists in TestersChoice. Each time a successful scan yielded a stack entry, the time tag is updated for the time the message was detected. The SW reads the stack and data blocks when the frame is completed. If it detected that the Time changed from the previous frame read, it should assume the message was executed (data sent ort received).

The BC\MultiRT SW control algorithm initiates a BC Go command, and monitors the activity of the BC state machine. After the active part of the frame finishes (bus list transmission completed) a passive state is entered. During the passive state, the SW normally exchanges data with the HW to read the received data and to load the stack and data for next frame. In the MultiRT standalone mode, the BC Go does not initiate the active part, but rather enables the MultiRT state machine. MultiRT will not reply to any message until instructed by BC Go. Since the BC Go isn't synchronized with the external BC, the HW automatically guesses frame passive time after both busses are inactive (passive) for a period of 1 ms. It is assumed that a external BC does not embed 1 ms message gap before it completes its bus list transmission. Once the MultiRT detected passive time, and the number of frames to Go is finished, the MultiRT state machine will go offline until a new BC Go command even if there are additional messages on the bus. As with the BC\MultiRT mode, once in the passive time, and the SW initiates a BC Go, the HW emulates Active time for the SW until the first bus activity, and from this time on, the actual activity monitor will report bus status to the SW.

If User Code 0022 is issued, the MultiRT will reply with illegal message (rather than not reply at all as in User Code 0002) for commands sent to an enabled RT but the command could not be located in the message list of the frame. This ME bit in status word will be replied when the MultiRT state machine is running!

If User Code bit 7 is set in MRT mode, i.e., 00A2 or 0082, then the Word Count field is ignored when searching for messages.

Word Monitor interface

The word monitor is a tool embedded inside the TestersChoice that monitors both 1553 lines and records all 1553 words that exist on them.
Each 1553 word is tagged by a tag word that provides additional data on the 16-bit data word. The tag word information includes the sync type, any error, the gap from the previous word and channel.
Notice the word monitor is not a message monitor, so message assembly has to be done by software if needed.

There are 4K words (8K bytes) mapped for the Word monitor (*see upgrade of card type B). These 4K words are mapped after the 4K words that serve the TestersChoice BC + MultiRT circuits.
The Word Monitor provides a single address pointer, that point to the next location that a word will be written to. The first address is 0x2000. The highest address is 0x3FFF. The address counter will cyclically return to 0x2000 after 0x3FFF.
The software should record the address pointer, wait for the transactions, then read the new address pointer, and finally read all words between the two pointers.

For pp194 each 24 bit word is stored in two consecutive entries, first the header 8 bits as LSBs with its descriptor, and then the 16 payload bits with its descriptor.

For cards of type B, there is an external memory added to save longer recording periods that the host PC is busy.
Since version 6.1 of the HW, the address pointer value is also between 0x2000 and 0x3FFF (0x6000 and 0x7FFF for module 1) but the 12 LSBs represent the number of 32 word blocks of data. In case the SW reads a value of 0x2001, it should assume that at least 32 words have been recorded and ready for download via the USB. In other words, the actual word address is derived from the 12 LSBs of the HW pointer times 32.
In case the incoming data stopped before completing a 32 word block, the HW would wait for 1 ms before injecting 32 words of time stamp in to the buffer.

For SW simplicity, the above HW pointer functionality would be the same for Card Type A and Card Type B.

Data Payload source

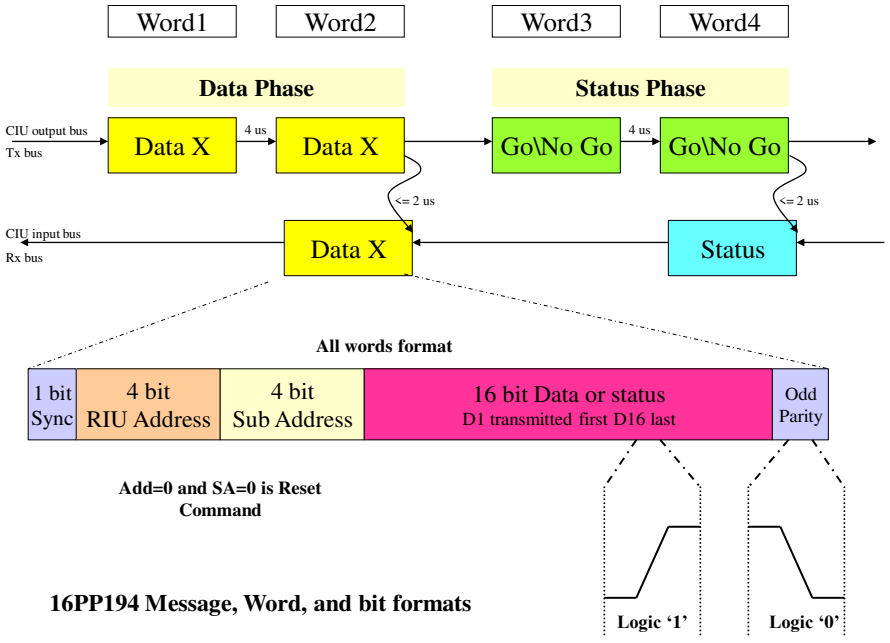
As of Version 7.2 of the tester the data automatic hardware incremental method was changed.

Bit 31 (MSB of the 32 bit vector that defines which RT is simulated) define the data source for the messages.
If this bit is '0', then the data transmitted in the message is loaded from the software. If this bit is '1' the data that is sent on the lines is sourced from a 16-bit register that constantly increments for each new data word.
Injecting incremental data from the hardware allows for hardware autonomous operation and data generation for BER test. It can increment with no PC load in 100% bus activity frames.

PP194 message format

The PP194 message is a very simple 6 word message performed in two phases.
The first phase is the data phase.
The second phase is the Go/NoGo phase, or status phase.

In each phase the CIU issues a 26 bit long word, and the RIU replies to that word after exactly 2 us, in parallel to the CIU transmitting that first word again. The CIU decides if to perform the second Go\NoGo phase pending a success in the first phase.
In any case if something goes wrong, the CIU can send a reset command.



16PP194 Message, Word, and bit formats

In the Data phase, the CIU sends a 26 bits data. The RIU digest the data and echoes the same data word back after 2 us.
If this phase is successful, when a Go\NoGo word of 26 bits is received by the RIU, it replies after 2 us with the status word in parallel to the CIU sending the Go\NoGo again.

H009 message format

The H009 messages include BC to RT and RT to BC without status, as well as mode commands with one data word.
The standard calls for clock and data signals for each bus A or bus B. Total of 4 lanes per bus compared with 2 lanes per bus in 1553.

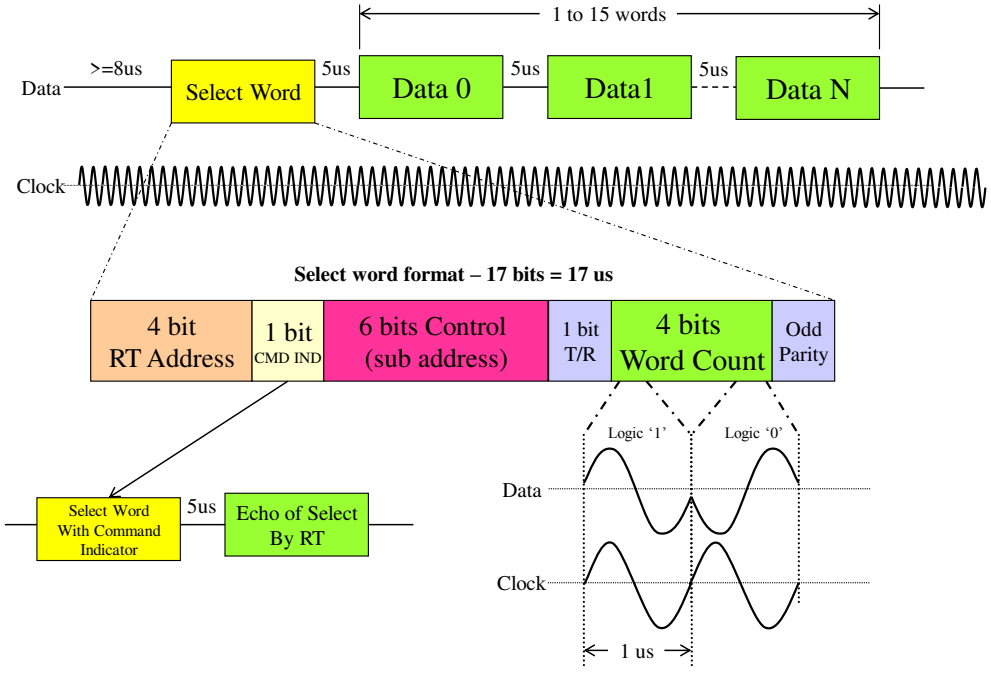
The command word is composed of 17 bits, and has at least 8 us of bus idle preceding it. Each data word is preceded by 5 us bus idle.

There is no full handshake with status respond.

The command word bits are:

Unit Addr 4	CMD IND 1	Control 6	T/R	#Words 4	Parity 1

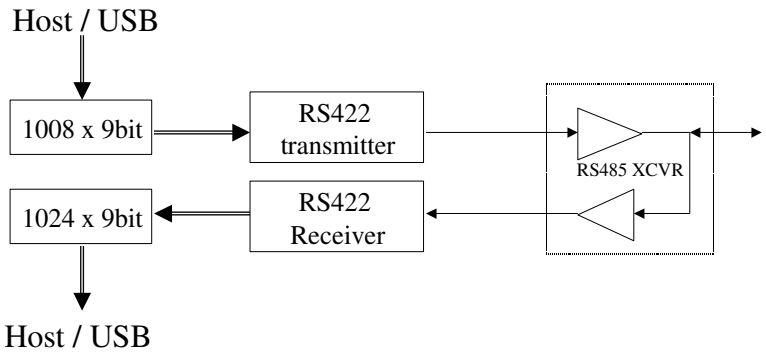
Unit Address is a 4 bit field that indicates which unit is targeted for this message.
Command indication is a single bit. When high it indicates it's a mode command with one word.
Control is like the sub-address in 1553 and covers 64 different values using 6 bits.
T/R is high for a unit transmit message and low for a unit receive message.
Number of Words has 4 bits to specify 0 to 15 words. 0 is not used as far as we know.
Odd Parity of 1 bit terminates each H009 word.

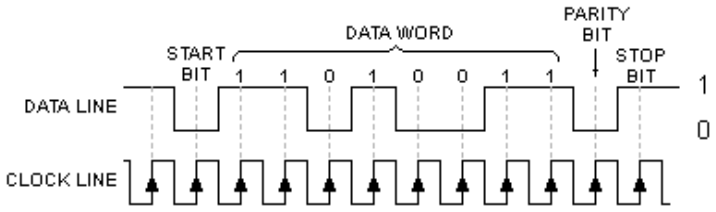


H009 Message, Word, and bit formats each bus A or B

RS422/485 Uart

The MultiComBox incorporates 2 transceivers of RS422/485 for each of the modules.
There are a total of 4 bi-directional RS422/485 lines.
Each transceiver includes a transmitter and a receiver.
The receiver also receives every word transmitted.
There is a 1008 word buffer for each one of the transmit buffers, and 1024 words for the receive buffers, total of 4 buffers for transmit, and 4 buffers for receive.
Each one of the words can be 5 to 9 bits long and also include a parity bit if needed.
The bit rate or baud rate can be set for each line. In addition, the parity type and stop bits can be set separately for each one of the lines.
The last 16 words of each transmit buffer is dedicated for control registers.





tib ytiraP htiw drow tib 8

The reception of all transmitted words allows for wrap around testing as well as working in both full duplex and half duplex protocols.
For full duplex, it is possible to connect line 0 as a transmit bus, and line 1 as a receive bus.
For half duplex it is possible to transmit a number of words to the remote unit, and that remote unit replies on the same lines. The receive buffer will contain the transmitted words followed by the received words.
For these modes, an additional setup has to be defined. Line mode can be transmit, or transmit\receive.
When Line mode is set to transmit, the RS422 hardware buffer will always drive the line, even when not transmitting.
When Line mode is set to transmit\receive, the buffer is driven only when transmitting.
Please refer to the appropriate registers description and Appendix E for Software control of the channels.

Sital Smart Wiring Block

The Passive TDR block for MIL-STD-1553 is designed to monitor the bus constantly and provide TDR information on all active UUTs on the bus, including the UUT in which it is located.
The block works throughout the mission, and picks up wiring faults, and latches the relevant information required to determine that there is a fault, and where on the bus.

Refer to "pTDR1553 Hardware Software Manual.docx" for additional information.

Asynchronous message mode

As of March 20th 2018 a new Async message sending mode was added to the transmission capabilities of MCX BC.
This new mode of operation allows the controller to inject a new message instantaneously to the transmissions on the bus.

1. If the MCX is running and transmitting a bus list, say message #3, and the Async is initiated, then it will be transmitted ONCE after #3 ends, but before message #4.
2. If MCX is running, but it is in a passive phase, i.e., between two frames, the message would go out instantly with no delay, and the first message on the next frame might be delayed until the Async message has been transmitted.
3. If the MCX is in idle mode, i.e. no bus lists are running, the Async message would be instantly transmitted ONCE and MCX returns to idle mode.
4. If the MCX is in idle mode, i.e. no bus lists are running, the Async message would be instantly transmitted ONCE and if during transmission, bus-list transmission is engaged, MCX would start the bus-list transmission back-to-back with the Async message completion.

Some avionic systems make use of asynchronous messages, and the above method facilitates this mode to MCX.
The usage of Async message on IDLE simplifies the procedure for sending messages, and avoids using bus lists. The controller can transmit any message one after the other, and each can be different from the previous one. This mode of operation might be useful for some application that want the controller intimately managing the bus list. Please note that if message results are tested, there would be no bus activity periods between two messages, depending on this result analysis takes.

An Async message is defined by a standard message block format, but one which resides between address 0 and 7. Writing word 7 initiates the transmission.
A second Async 2 message is provided in addresses 8..F. Using these two async messages, one can work in pipe line mode, and achieve very high bus utilization even if using USB interface.

It is recommended that the data blocks and state blocks be located after the last block. Assuming there are 64 blocks supported, block location 65 and 66 should be used.

RT response times

As of April 2018, MCX can control the RT response time in mRT mode (and later in BC + mRT) to a programmed time.
Before, the response time was set to 8 us. This is the default time.

The control is by specifying a value between 1 and 511. Each step, i.e. the LSB is a clock cycle of the main clock. The main clock can be one of 30-Mhz (MCX), 32-Mhz (GRIP2) and 33-Mhz (PCI), and potentially others. The value of the

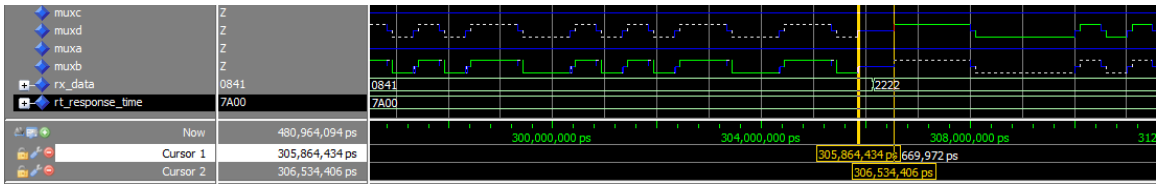
Register address 0x18 serves as the programming port, and reporting port for the global setting of RT response time.

Please note that the counter starts counting from the end of the command so that a value of 1 can create a continuous, no gap mode. Some monitors might see that as a fault.
The standard allows 4 us to 14 us, and these are between the zero crossings of the parity to the zero crossing of the sync. There is still ½ us after that parity sync, and 1½ us sync before zero crossing, so there are actually 2 us to 12 us bus "dead" time expected if the command has no leftovers on the bus.

The RT Response time counter starts counting from the end of the parity, and once reached the programmed value, would start the sync. So it actually counts the bus "dead" time. Setting it to 2 us, would make a 4 us response, and 2 us bus "dead" time.

The mRT is designed initially to avoid response if the bus is not quite ("dead" bus) when it is time to transmit the response. If the response time is set to very fast, it might be needed to respond before the bus does quite down. In order to persuade the mRT to respond on a busy bus, an additional bit should be set to '1' in the register.

It was measured that the bus dead time is about 660 ns if counter is set to 0.



Kindly refer to register 0x18 description in the following sections.

Programming and Setup

Overall Structure

There are number of configuration and setup registers that are mapped on the memory space. These registers are written both into the memory and registers, and read from the registers if these registers are being updated by the core, or from the memory when they are constant for the HW.
The memory and registers are arranged in groups. The belonging of a particular data to each group depends on the owner of the data. If the Host PC writes to a particular register of memory, then it is the owner of the data. If the TestersChoice core writes / updates it then the core is the owner of that data.
All registers owned by the same owner are mapped consecutively inside a group, thus the Host PC can read or write them with burst operations rather than accessing particular registers and bits and performing single word read or write operations.
This technology increases the communication efficiency between the Host PC and TestersChoice.

The memory mapping was designed to use pointers to data structures. A careful assignment of these pointers by the SW allows for configuring large chunks of data to the same groups yielding bigger bursts of read and write cycles.

Operation Methodology

The basic operation methodology is for the Host PC to setup a list of message entries in a stack structure. Each stack entry defines a particular message that would be transmitted on the bus. The Host defines an initial stack pointer and an initial stack length and issues a start command. Once started, the core starts transmitting the messages one after the other until all messages have been transmitted. When finished, the core will wait for the frame time counter to zero. If the mode of operation is auto-repeat mode, a new frame would begin at this point, else the core will remain idle.

Several frames can be arranged in the memory structure with the initial stack pointer and initial stack length set to point to the relevant frame before the start command.

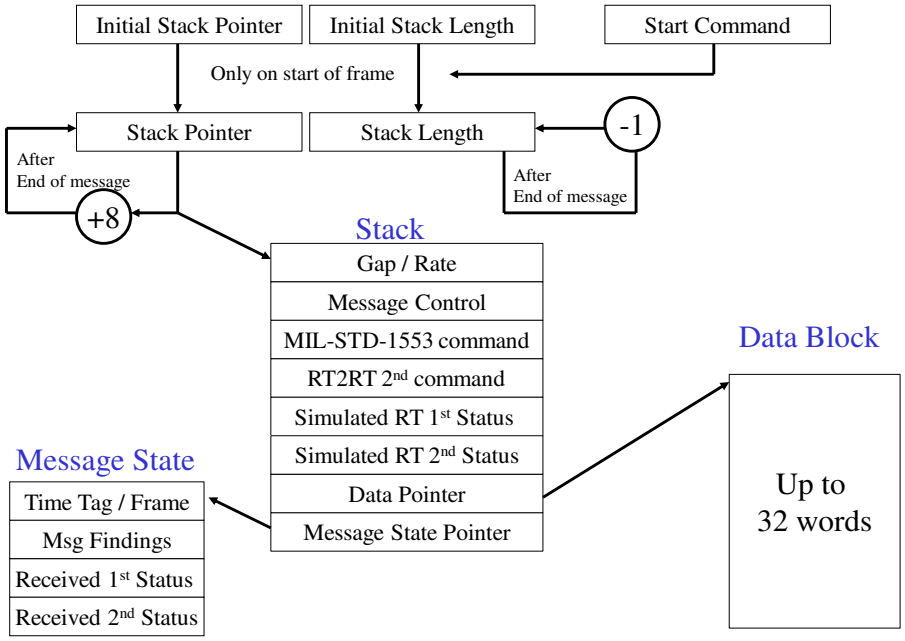
Notation

If a reference is made to a particular register and a specific bit the notation is R01B12 for Register 01 bit 12.
If a reference is made to a particular memory location the notation is M0100 for memory address 0x100.

Memory Structure

The host should prepare the frames of messages in the memory stack prior to initiating the start command.
Stack entries point to memory data blocks and Message state block described in the following sections.

Please refer to the following memory structure:



In PP194 mode, the message is mapped on the same memory structure. The Hardware will activate the command in PP194 mode automatically.
The following PP194 highlights are:

1. Where not mentioned, stay the same as in 1553.
2. Message Control – a single bit #2, if set enables a PP194 message.
3. MIL-STD-1553 command – 4 bits of RT address, and 4 bits of Sub address will be used. All other bits are ignored.
4. RT2RT 2nd command – ignored.
5. Simulated RT 1st Status – If the RIU (RT) is simulated, the transmitted status word will be taken from this entry.
6. Simulated RT 2nd Status – Not used in PP194 mode.
7. Data pointer – Points to the data block. The first data in the block is the PP194 data word contents used in data phase (word 1 and word 2). The second word is the Go or NoGo code used in status phase (word 3 and word 4).
8. Msg findings – used. Reports errors in message.
9. Received 1st Status – contains the status return in status phase.
10. Received 2nd Status – not use in PP194.

Registers 0x30 to 0x34

Initial Stack Pointer (Fix address 0x0030) – The Host PC loads this address. The Address points to the first stack entry message of the frame. Any address with 3 LSBs being '0' and higher or equal to 0x0048 can be used. Notice that several frames of messages can be stored in the memory structure prior to the start command, and just before the start command the Host PC loads this pointer to the frame it wants.
The value of this register is copied to the running stack pointer and is not used by the core until start of next frame. The host PC can update this parameter 1 us after issuing the start command for the next frame.

Initial Stack Length (Fix address 0x0031) – The Host PC loads this address. This parameter indicates the number of messages that would be processed in the next frame. The value of this register is copied to the running stack length and is not used by the core until start of next frame.
If 0 is loaded to this parameter, no messages would be transmitted after start command. The host PC can update this parameter 1 us after issuing the start command for the next frame.

Stack Pointer (Fix address 0x0032) – The core loads this address. The Address points to base of the current stack entry message being processed. The value is incremented by 8 following the end of each message processed.
The Host PC should not write to this register AFTER start command and BEFORE the frame is complete.

Stack Length (Fix address 0x0033) – The core updates this address. The value of this parameter indicates the number of stack entries that are still going to be processed. This parameter is decremented by 1 after each message. When zeroed, frame transmission stops. The Host PC can write to this register but the core will overwrite the value after each message completion.

Start Command (Fix address 0x0034) – By writing to the start command address, the TestersChoice machines start processing the messages. The value written to the start address is the number of frames will be executed. The value read from this register is the value written to it. The core does not change its value. A frame number up counter can be read from register 0x22.
Writing to this register should be the last write before frame begins.

Writing the value 65535 will transmit 65535 frames.
:
Writing the value 1 will transmit one frame.
Writing the value 0 will transmit endlessly.

Stack

The stack is a set of message entries, 8 words deep each. Each stack entry is composed of 8 words. The stack is arranged with indirect pointing database. The stack is intended to be written only once during setup by the host, then to be used by the core during operation.

A stack pointer points to one of the messages. The initial stack pointer is loaded by the host, and could point to any non reserved location in the 64Kx16 memory space, however, initial stack pointer should point to a word address with all of three LSBs being zero (Word address 0x0050 should be the base address for stack).

Gap / Rate (stack offset 0) - This parameter works in either one of two modes of operations: the message Rate mode, or the message Gap mode. The mode is controlled globally for all stack entries by a configuration register.

Gap mode
In Message Gap mode, this register defines the amount of microseconds from the beginning of this message to the beginning of the next message. A value could be in the range of 0 us to 64K us =~ 65 ms.
A value shorter than the message actual length on the bus results with a default inter-message gap of about 10 us.
In cases where not all messages are needed in every frame, but rather some are transmitted every other frame, or every 4th frame… for example, it is popular to define minor and major frames. With the usage of these delays, the host can program minor and major frames.

The HW core however, supports a more compact and efficient method for sequencing of such low rate messages. The Rate mode is a much more appropriate method for such systems.

Rate mode
For each message define its rate. Possible rates are
0 – skip this message.
1/1 – send every frame.
1/2 – send every second frame.
1/4 – send every forth frame.
1/8 – send every 8th frame.
:
1/N – Where N is a power of 2 and N_{max} = 2^14, i.e., once every 16,384 frames.
15 – send only once. Core will change to 0 after transmission.

If two stack entries point to the same message, than the resulting rate of that message would be higher. The resulting rate would be the sum of both rates. For example 1/4 + 1/16 would be 5/16, which is almost 1/3 of the frame rate.

The HW core sequences the messages at lower rates, such that for each frame, only the 1/1 rate messages are transmitted along side only ONE slower rate messages. Such that 1/N (N>1) messages and 1/M (M>1, N<>M) messages are not transmitted in the same frame.

The HW core provides a register that indicates the frame number. After start command this frame counter starts counting up until the operation is stopped. The host can determine which message is transmitted on which frame using a simple equation. This will allow the host to update the transmitted data buffers on time.

It might happen that the transmission length in time of all messages of rate 1/1 and messages of lower rate in a specific frame sum up to a total length which is too long for the frame length. If one of the following frames is not crowded, the HW core supports message

skew definition. A specific stack entry message can be skewed forward 1 to 15 frames ahead from its designated frame.

Please refer to Appendix A for a more in-depth theoretical description of Gap and Rate modes.

Message Control (stack offset 1) - This parameter defines the various modes of operations for the specific message. Each bit is explained below.
The "Write Verify" mode allows the SW to instruct the HW to perform a Verify before write operation on the received data words. If the RT message words are known expected data "Write Verify" mode can be enabled. In this mode words written into the memory are first verified against the memory location contents where they are written, and if the data does not match, the "good data block received" bit 3 in the message state register is reset.

MIL-STD-1553 command (stack offset 2) - This parameter defines the 1553 command that will be transmitted for this stack entry.

RT2RT 2nd command (stack offset 3) – In case the Message control define this message as an RT to RT format, This parameter defines the 2nd 1553 (Transmit) command that will be transmitted for this stack entry.

Simulated RT 1st status (stack offset 4) – In case the command approaches an RT that is defined as a simulated RT, This status word would be transmitted by TestersChoice as if an RT transmitted it. If it's a receive or transmit command, TestersChoice will use this status word. If this command is an RT2RT command format, then this status word would be the status of the first simulated RT. If both RTs are simulated, then this will be the transmitter's status. If only the receiver RT is simulated, this status would be its status word.
For simplicity of programming, if only one RT is simulated, fill both this entry and the next with the same "returned" status.

Simulated RT 2nd status (stack offset 5) – only in RT2RT command format, with BOTH RTs simulated, this parameter would be the receiver RT status word.

Data Pointer (stack offset 6) – This word can point to any 64Kx16 memory location. This pointer points to the data block in memory. Data block should be 1 to 32 words deep. It would be efficient to arrange all transmitted data blocks consecutively in memory to allow their update with a single burst write. If misfortune happens and this parameter is un-initialized or is zero, data will be stored in the first 32 locations which is reserved just for such misfortunes…
Many data pointers can point to the same location (if these messages' data is irrelevant for the Host PC).

Message State Pointer (stack offset 7) – This word can point to any 64Kx16 memory location. This pointer points to the Message State Block, which collects message information during its transmission.

Bit description of stack contents:
Notice first word (offset 0) has two options depending on configuration.

Offset	Purpose	Bits	Description
0	Inter-message gap (Gap mode)	15..0	Number of microseconds between the start of this message and start of next message.
0	Spacing (Rate mode)	15..8	Spacing gap: Number of microseconds between **end** of message and **start** of next message. Could be left zero.
	Rate (Rate mode)	7..4	Case N (decimal value of bits) is 0: Skip this message. 1: transmit this message every frame. 2..14: transmit this message every 2^(N-1) frames. Will be transmitted in frames who's frame N-1 LSBs = 2^(N-2) (unless skewed). 15: Transmit this message once. The HW core resets this value to 0 after message has been transmitted once.
	Skew (Rate mode)	3..0	Frame skew: Defines a number M between 0 and 15. Will skew the message M frames away from its rate planned location (defined above). M is derived from max(N-1,4) bits of this 4 bit field. Example: If N is 3 then only bits 1..0 are added to "10" (2) and compared to 3 LSBs of frame number.
1	Message Control Deafult: xFA80	15	Transmit internal time tag data in Synchronize with data mode command: '1' - Use time tag counter as data. '0' - Use data word in message block. In PP194 – conditional Go/NoGo – '1' will transmit the Go/NoGo command if data phase OK. '0' will transmit Go/NoGo phase regardless of data phase.
		14	'1' – Mask Message Error (ME) bit in status word. '0' – No masking. If ME is set, this will cause "Status Bit Set" condition.
		13	'1' – Mask Service Request (SR) bit in status word. '0' – No masking. If SR is set, this will cause "Status Bit Set" condition.
		12	'1' – Mask busy bit in status word. '0' – No masking. If busy is set, this will cause "Status Bit Set" condition.
		11	'1' – Mask Sub-System Flag (SSF) bit in status word. '0' – No masking. If SSF is set, this will cause "Status Bit Set" condition.
		10	'1' – Mask Terminal Flag (TF) bit in status word. '0' – No masking. If TF is set, this will cause "Status Bit Set" condition.
		9	'1' – Mask Reserve bits (RSRV) in status word.

			'0' – No masking. If RSRV is set, this will cause "Status Bit Set" condition.
		8	'1' – Retry enable. Perform retry if message fails. The number of retries and the retry bus is defined in configuration register in address 0x004A.
		7	'1' – Transmit on bus A. '0' – Transmit on bus B.
		6	'1' – Perform internal loop back test. Message does not go out to 1553 bus but rather loop backed just like the normal echo of the transceiver. The TestersChoice core simulates the RTs, thus this internal loop back can be completed with out any error.
		5	If R08B11='1', i.e., Mask mode for BCST bit then this bit is the mask bit: '1' - will mask checking BCST bit '0' - will verify BCST is not '1', otherwise "Status Set" condition found. ELSE in compare mode: This bit should be equal to BCST bit, else "Status Set" condition found.
		4	'1' – End OF Message Interrupt Enable (if not masked by interrupt mask register).
		3	'1' – "Write Verify". Before Writing received data to memory, verify it matches the memory preloaded to the data block. Only for transmit commands. Check Message state bit 4 to see if verify passed. '0' – do not verity. (Receive commands are not verified).
		2	'0' – 1553 message format. '1' – PP194 message format.
		1	Ignored. Should be '0' for future compatibility.
		0	'1' – The message is an RT2RT command format.
2	1553 command	15..0	The 16 bits of the command sent. The core detects BCST and Mode formats automatically. PP194 – Use 4 bits of Address and Subaddress only.
3	2nd command	15..0	In RT2RT mode only: this is the transmit command. The RT2RT mode set in the message control word bit 0. PP194: not used.
4	1st Simulated RT Status	15..0	This entry defines the first RT simulated status response. This data will be transmitted by TestersChoice. PP194: simulated status word.
5	2nd Simulated RT Status	15..0	This entry defines the second RT simulated status response in RT2RT format. This data will be transmitted by TestersChoice. Not used in PP194.
6	Data Block Pointer	15..0	Points to the data block where the message data is either stored or read from.
7	Message State Pointer	15..0	Points to the starting memory location of the Message State block.

Message State

The message state block incorporates 4 words that are collected from the bus while the message is being processed.

Time Tag / Frame (block offset 0) – This parameter works in either one of two modes of operations: the message Rate mode, or the message Gap mode. The mode is controlled globally for all stack entries by a configuration register.
In Gap mode - The 16 LSBs of the 32 bit time tag counter are stored here when the message was launched by the core.
In Rate mode – A frame counter is incremented by 1 at EOF. This frame counter value is stored in this entry when the message is transmitted.

Message Findings (block Offset 1) – This register is collects a number of parameters during the message processing. Bits 13 downto 0 are updated at EOM after change.

Received RT 1st status (block offset 2) – The first status that was received from a real non-simulated RT is stored here.

Simulated RT 2nd status (stack offset 3) – only in RT2RT command format, the second status (Rx Status) that was received from a real non-simulated RT is stored here.

A detailed description of the Message State block:

	Name	Bit	Description
0	Time Tag Word 16 LSBs. (Gap mode)	15..0	16 LSBs of the real time counter. Written by core when the message started.
0	Frame Number (Rate mode)	15..0	Frame number when this message was transmitted. Frame number is incremented every EOF. It is recommended to init this value to 0xFFFF before run.
1	Message findings	15	End Of Message – Set to '1' by the core when the message has been complete.
		14	Start Of Message - Set to '1' by the core when the message has been started. In most cases, this bit is stuck at '1' after end of message if there is a 1553 bus-coupling problem.
		13	'0' – Was sent on Bus A. '1' – Was sent on Bus B.
		12	'1' – Error was found in the message. Bits 10, 9, 8, 3, 2, 1, 0 indicate cause of error.
		11	Status Set. One of the status bits (excluding BCST bit) of the status return was '1'. Masking ignored. BCST bit works in either mask mode or compare mode. In mask mode it works like other mask bits on the BCST bit. In compare mode, Status set occurs if BCST bit is different from bit 5 of BC control word.
		10	Format Error. The returned echo from the RT contained 1553 violations. See bits 3, 2, 1, 0 for a more accurate guess of the source of the problem.

		9	Response timeout. The RT responded too late or didn't respond at all. In PP194 – The RIU did not respond properly.
		8	Loop back failed. The nature of 1553 bus is that every word transmitted, is also echoed back. The core verifies that the echo is correct and equal to the transmitted word. If not, this bit is set to '1'. Also set in messages with error injected. Tip: The source of this type of error could be transceiver fault, or bus coupling problem. In PP194 –Loop back Failed.
		7	Unmasked Status bit set. This bit will be set to '1' if one of the status bits are set high and its appropriate mask bit in the BC control word is unmasked ('0'). BCST bit influences only in mask mode. See registers section for description of BCST bit.
		6..5	Number of retries done for this message. "11" is 3, "10" is 2…
		4	Good data block received by TestersChoice, waiting in Data Block. '1' – after an RT-BC, RT2RT, and Transmit Mode code with data commands if the message ended OK. '0' – after other message types, or if the above type of message was invalid. '0' – for received words that did not match the expected values if "Write Verify" mode is enabled for the message. Loop back test failure does not cripple this bit result. ~~In PP194 – Both phases completed successfully and a real RIU sent its status and saved to memory.~~
		3	'1' indicates the RT responded with wrong RT address. In PP194 – RIU status respond with wrong RIU address.
		2	'1' indicates that the RT transmitted a wrong number of words. In PP194 – RIU Data phase error.
		1	'1' – Incorrect sync type response by RT. In PP194 – RIU Status phase error.
		0	'1' – Invalid word. Indicates that the RT responded with a word containing 1553 errors. In PP194 – The RIU responded with Manchester / parity error.
2	Received 1st status	15..0	First status received from un-simulated RT. In PP194 – Status bits of status word.
3	Received 2nd status	15..0	Second status received from un-simulated RT.

Data Block

The data block is the location for either storing data words received from the bus as a response for transmit commands, or as a location for the Host PC to write data word payloads that need be transmitted to the bus.
Two different messages can point to the same data block. New data will override the former.

Memory Setup Example

Typical Memory structure for a two-message design:

Hex Address	Description	Comments
0000 to 0007	Async 1 Message block	Asynchronous message insertion
0008 to 000F	Async 2 Message block	Asynchronous message insertion
0010 to 0013	Real Time Counter	Read and write real time timer
0018 to 001F	Reserved Memory	Intended for future usage.
0020 to 002F	Card State Registers	Initialized by host.
0030	Initial Stack Pointer	Initialized by host.
0031	Initial Stack Length	Initialized by host.
0032	Current Stack	Updated by HW
0033	Stack Remaining	Updated by HW
0034	START command	Initialized by host.
0040 to 004F	Configuration Registers	Initialized by host, updated by core.
0050	Gap / Rate	Initialized by host.
0051	Message Control	Initialized by host.
0052	1553 command	Initialized by host.
0053	RT2RT 2nd command	Initialized by host.
0054	Simulated RT 1st status	Initialized by host for simulated RT.
0055	Simulated RT 2nd status	Initialized by host for simulated RT.
0056	data pointer	Initialized by host.
0057	State pointer	Initialized by host.
0058	Gap / Rate	Initialized by host.
0059	Message Control	Initialized by host.
005A	1553 command	Initialized by host.
005B	RT2RT 2nd command	Initialized by host.
005C	Simulated RT 1st status	Initialized by host for simulated RT.
005D	Simulated RT 2nd status	Initialized by host for simulated RT.
005E	data pointer	Initialized by host.
005F	State pointer	Initialized by host.
0100	Frame number / Time Tag 16 LSBs	Last transmitted frame / Last transmitted time
0101	Message Findings	HW reports message state & errors
0102	Received 1st status	Received for non-simulated RT.
0103	Received 2nd status	Received for non-simulated RT.
0104	Frame number / Time Tag 16 LSBs	Last transmitted frame / Last transmitted time
0105	Message Findings	HW reports message state & errors
0106	Received 1st status	Received for non-simulated RT.
0107	Received 2nd status	Received for non-simulated RT.
0108 – 0127	1st message block	
0128 – 0147	2nd message block	

Memory Setup before start

- the HW. Load 0 for no-op.
- Setup all stack entries. A set of 8 words each, one after the other.
- Setup a message state block for each stack entry.
- Setup data blocks for all messages. For messages that their data is received from the RTs, simply reserve the space. For simulated RTs, load the simulated data into the data block.
- Load the initial stack pointer (0x0030) to point to the first message in the stack.
- Load the initial stack length (0x0031) to the number of messages that are processed by
- Start TestersChoice frame engine by writing the number of frames to the start (0x0034).

Note: When writing to address 0x30 and 0x31 and 0x34, you may also write over 0x32 and 0x33. This supports burst write operation and optimizes the CPU access.

Stack Pointers and Length

TestersChoice can either perform a single frame of messages, or work in **auto repeat** mode, which will run, frames one after the other.
In any case, the host PC should load the **Initial Stack pointer** (M0030) and the **Initial Frame Length** (M0031). After the HW core is triggered for operation, the HW core copies these two words to the two running addresses **Stack pointer** (M0032) and **Stack Length** (M0033). The values in these two addresses are updated by the HW core and could be used by the host to track the state of the frame.
The stack length counter is decreased by one after each message. When this counter reaches zero, the frame is finished. If auto repeat mode is enabled, the next frame will start automatically. When the next frame is started, the initial values from memory locations M0030 and M0031 are again loaded to the running memory locations M0032 and M0033.

Registers

There are several registers mapped to the memory space. No special register access is available.
The following section describes the registers and the operational bits of these registers.

If a reference is made to a particular register and a specific bit the notation is R01B12 for Register 01 bit 12.
If a reference is made to a particular memory location the notation is M0100 for memory address 0x100.

Hex Address	Description	Comments
0..7, 8..F	Async 1 & 2	2 asynchronous message hard location.
0010	Time Tag 15..0	When read, other 32 bits are latched for additional reads
0011	Time Tag 31..16	
0012	Time Tag 47..32	When Written, latched bits 47..32 & 31..0 are written
0013	Time Tag 63..48	Place holder for future support of 64 bit time
		"On-the-fly read" registers
0020	General Status	Interrupt conditions & state machine state.
0021	Frame Time Remaining	Number of 100us steps to End Of Frame (EOF).
0022	Frame Number	Frame Counter value.
0023	Status bits	Status bits of last returned status word
0024	Time Tag LSB	16 LSBs of time tag 32 bit counter. Cyclic every ~4.1 sec
0025	Time Tag MSB	16 MSBs of time tag 32 bit counter. Cyclic every 76 hours
0026	Bus Load	0..4K out of 4K busy on A OR B
0028	Word Monitor Address	Address pointer to next location of monitor write.
002D	RS485 0 activity	
002E	RS485 1 activity	
002F	Version Register	Version of Hardware
		Frame control registers
0030	Initial Stack Pointer	Written by Host.
0031	Initial Stack Length	Written by Host.
0032	Stack Pointer	Written by core. Updated after start command.
0033	Stack Length	Written by core. Updated after start command.
0034	Start Command	Written by Host.
		"Write-once" per "start" registers
40-43	RS485 setting	
44-46	Error Injection	
0047	Multi RIU 0 to 15	Enable simulated RIUs 0 to 15. 0 is used for Live RT on.
0049	Interrupt enable	Interrupt masking
004A	Configuration	Modes of operation for TestersChoice
004C	Reset & Stop	SW reset of HW and graceful stop requests
004D	Frame Time	Number of 100 us steps for frame duration
004E	Multi RT 0 to 15	Enable simulated RTs 0 to 15
004F	Multi RT 16 to 30	Enable simulated RTs 16 to 31

Please find a detailed description of the bits for every register in the following sections.

Async 1 & 2 Message Block

A message entry of 8 words per each asynchronous message transmission. Kindly refer to the stack description section for details on the 8 words that make up each message.
Writing to register address 7 flags Async 1 message as ready for transmission.
Writing to register address F flags Async 2 message as ready for transmission.
If both are pending for transmission, Async 1 would be transmitted first.
It would ask the transmission engine to allow it to be transmitted instantly, or as soon as the on-going message has finished.

Read only registers

48 bit Time Tag Registers Address 0x0010...0x0012

The 48 time tag counter which is tagged to each monitor entry may be read or written to in order to align it with the rest of the system.
Accessing this timer has been added in March 2018.
The resolution of this time tag counter is ½ us. The amount of time before cycling condition is reached is 2^47 microseconds which is almost 4.5 years!

RT Response time Register 0x18

Bits	Purpose	Description
15..10	Clock Rate	6 bits that describe the Mhz clock rate of the counting engine. Could be 30 (MCX), 32 (Grip2), 33 (PCI)
9	Anyway	'1' - Respond to command even before bus quiets down.
8..0	RT Response	Number of clocks for RT response. Default value is 8*Mhz.

Default value: Clock rate as it is. "Anyway" set to '0' and RT response set to 8 us.

To set a new RT Response time please follow the following steps:
1. First read register 0x18, and see the clock rate value in bits 15..10, and set it to F.
2. Calculate the requested bus dead time as multiple of 1/F => N
3. Write N to bits 8...0.
4. Consider setting bit 9 to '1' if the required response is below 2 us.
5. Read register 0x18 again to verify it was set correctly.

General Status Register Address 0x0020

This register indicates the cause of an interrupt and the card status.

Interrupt status register will set its bit flags high if condition occur even if it is masked by the interrupt mask register. An interrupt, however, will be generated only if both a bit is set and is not masked.

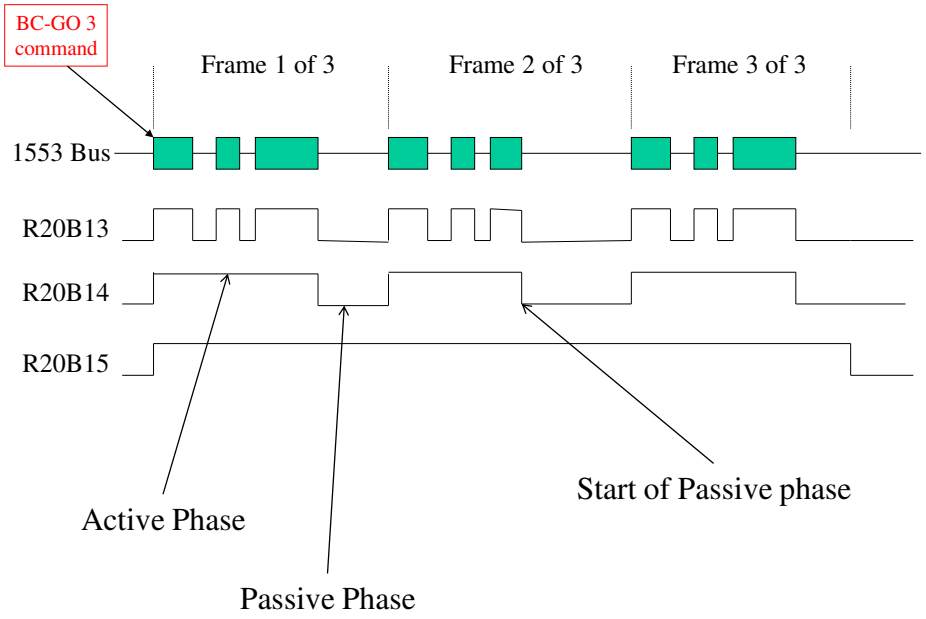
To reset this register either read it or write '1' to bit 2 of stop & reset register.

Bit number	Read/ Write/ default	Description
15	Status Read/'0'	'1' – Tester Busy. Tester engine has been started, and has not finished its frames. This signal will stay high until frames ended or halted by host or error. '0' – Tester engine is waiting for BC-GO command.
14	Status Read/'0'	'1' – Frame active - from start of first message in frame till end of last message in frame. '0' – Tester is not accessing the memory or registers.
13	Status Read/'0'	'1' - Message active - from start till end of message. '0' – during inter-message gap.
12	Status Read/'0'	'1' – Valid PP194 words detected. '0' – no PP194 words detected.
11	Read/'0'	This bit is an OR of all interrupts bits 0 to 10.
10	Read/'0'	'1' - Transmitter fail safe timeout.
9	Read/'0'	'1' – Data Pointer pointed to a protected area where data blocks should not be present. Data was not written and memory was not corrupted. (Handshake Error in DDC devices)
8	Read/'0'	'1' - retry has occurred. Each message has a bit in status word in stack that indicates if retry was done.
7	Read/'0'	'1' – Word Monitor Data Overrun – The SW did not read the Word Monitor fast enough and the buffer overran the previous data.
6	Read/'0'	'1' - Time Tag Rollover from 0xFFFFFFFF to 0x0.
4	Read/'0'	'1' – Message was sent. This interrupt can be enabled or not for each message with Message control word bit 4.
3	Read/'0'	'1' – TestersChoice has reached end of frame.
2	Read/'0'	'1' – RT has not replied OR replied with error OR loop-back error of transmitted words OR "Write Verify" mode failed. Access the Message State registers for error report on a message basis.
1	Read/'0'	'1' – Wrong RT address response or one of the (non BCST) status word bits were set high. BCST bit sets this bit high if configurations register 4 bit number 11 is set low (compare mode) AND the BCST bit is different from the BC control word of the message bit #5.
0	Read/'0'	'1' – End of message has occurred.

Default Value: 0x0000

Please refer to the following diagram to understand the level of the three-engine status of the tester:

R20B13 – Register 0x20, bit #13 – Message busy.
R20B14 – Register 0x20, bit #14 – Frame busy.
R20B15 – Register 0x20, bit #15 – Tester busy.



Frame Time Remaining Register Address 0x0021

16 bits that define the number of 100us left until end of frame, or zero.
This is a count down register that is read only.

Frame Number Register Address 0x0022

~~The frame number register resets to zero upon start command and is incremented every END of frame. If the tester engine stopped for any reason this counter will maintain its last value until next start command, after which it will reset to 0 again.~~

~~The frame number register resets to zero upon start command and if the frames to go register is 0. Otherwise,~~ every end of frame active part of a frame this register will increment by one.
This change was done to support multiple BC Go 1 needed for the BRT mode.
In MRT it is more common to do a BC Go 0, and this is when the frame number will reset and only increment from then on.

~~If it is required to reset the frame number register, a Stop EOF command should be issued.~~
Reset register (0x4C) bit 7 set high will reset the frame number register.

Time Tag LSB Register Address 0x0024
Time Tag LSB Register Address 0x0025

The time tag register counter can be loaded to any value by writing it to these two registers. Register 0x0024 holds the LSBs of the Time Tag counter and register 0x0025 holds the MSBs.
After register 0x0024 is loaded, register 0x0025 should be loaded within 64us in order to assure data consistency.

The time tag counter counts in clock steps of 64 us.
The counter will rollover every 64us * 2^32 =~ 76 hours.
The value of this counter is also reset by the hardware raw_reset line, or the software reset done by writing '1' to bit 0 or 3 of the stop / reset register.
In Gap mode, the 16 LSBs of the time tag counter is placed in the message state block when the message transmission begins.

When performing a "synchronize with data" command, the time tag data sent on the bus is the 16 LSBs of this counter.

For best system synchronization it is recommended to perform synchronize with data in broadcast command every couple of seconds. In this case the 16 LSBs of the time tag counter is loaded simultaneously to all RT's internal time tag counter.

Programmer warning: When reading the time tag as two registers one after the other, there might be a potential hazard. During the gap time between readings of the LSBs and the MSBs, the counter might advance by 1. To overcome this problem, please check the value of the 16 LSBs, if it is 0, please read the 32 bits time tag again. On the second read, the value is correct even if the 16 LSBs are still 0.

Address	Bit number	Read/ Write/ Default	
0x0024	15..0	Read/Write/0	Read or write 16 LSBs of Time Tag Counter.
0x0025	15..0	Read/Write/0	Read or write 16 MSBs of Time Tag Counter.

0x0024 Default Value: 0x0 - incremented every 64 us.
0x0025 Default Value: 0x0 - incremented every ~ 4.1 seconds.

Bus Load Register Address 0x0026

The busload register provides a figure that helps calculate the actual time the bus was active compared to the total time.
The load is measured for bus A and B in the current module. The bus is considered busy in the inter-message gap, as well as average of 8us after the message is complete.
The measure is based on a 16us one shot. If in this 16us an activity is found, the counter is incremented, else its not.

Busload is measured periodically every 0.65 seconds.
The 12-bit number provided in this register is in the range of 0 to 0x0FFF. 0x0FFF indicates 100% bus utilization, while 0x0 indicates 0% usage.
0x0800 indicates 50% bus utilization.

Word Monitor Address 0x0028

This register holds the next location in memory where the word monitor will write its findings.
This registers starts with 0x2000, goes all the way to 0x3FFF, and cycles back to 0x2000.
Hardware or software reset, resets this register to the 0x2000 value. Otherwise the register is only readable.
As of version 6.1 (date code xx5A or later) the HW pointer 12 LSBs indicate the number of 32 word blocks instead of the words address directly.
Card type B has a maximum of 256Kx16 word buffer, while Card type A has only 8Kx16 internal memory. The functionality of this register is the same for both card types for SW simplicity.

Temperature Sensor address 0x0029

This register holds the temperature of the MCX card in case such a sensor is installed.
The value is in the 13 LSBs, it is 2's compliment, and the LSB is 0.0625C
0x190 is 25C.

RS485 tail length 0x002A

Bits 5..0 indicate the delay between transmit and receive for RS485 line 0.
Bits 13..8 indicate the delay between transmit and receive for RS485 line 1.
These delays would be different if there is a termination resistor on the other side or not.
Using the recorded number, the tester can warn that the bus is not connected to the load.
Resolution is in nanoseconds

RS422 Line-in 0 status address 0x002D
RS422 Line-in 1 status address 0x002E

This register is cleared when read.

Bit number	Read/ Default	
15	R/0	'1' - There was a Parity error in one of the RS422 receive words. *
14	R/0	'1' – Incorrect number of bits in RS422 received words. *
13	R/0	'1' – Format error, either start or stop bits error. *
12	R/0	'1' – Receiver is busy receiving words. Still more words are expected.
11	R/0	'1' – Transmitter is busy transmitting words. Wait until '0' before transmitting a new message.
9..0	R/0	10 bits that point to a memory location in the 1K block where the next RS422 word received will be saved.

* Please check baud rate settings!

Version Register Address 0x002F

This register returns the hardware version of the core.
SW application can validate if it's using the correct version with this register.
Date code is a 4 nibble code. The first 2 left most nibbles are in the range of 1 to 31 and represent the day in month, the next nibble represents the month and in the range of 1 to C, and the right most nibble represents the year (as offset from year 2010) in the range of 0 to F (2010 to 2015).
Examples:
Jan 1st 2000 : 0x0110.
May 10th 2010: 0x105A
Dec 31st 2016: 0x31C6

Write only registers

RS422 Line 0 configuration Register Address 0x0040
RS422 Line 1 configuration Register Address 0x0042

Bit number	Read/ Write/ Default	
7..6	R/W/0	Number of stop bits: "00" – 1 stop bit. "01" – 1.5 stop bits. "10", "11" – 2 stop bits.
5..4	R/W/0	Parity bit added to each word: "00", "01" – no parity bit. "10" – Even parity. "11" – Odd parity.
3	R/W0	'0' – Line mode transmit only. '1' – Line mode transmit and then receive, or receive only.
2..0	R/W/0	Number of bits in each word: "000" – 5 bits. "001" – 6 bits. "010" – 7 bits. "011" – 8 bits. "100" – 9 bits.

RS422 Line 0 frequency divider register Address 0x0041
RS422 Line 1 frequency divider register Address 0x0043

This register holds the value of the main clock divisor to get the proper baud rate.
Only the 14 LSBs are used. The divisor legal range is 2 to 2^14-1 (16383).

Example: for a 30Mhz system clock, and a 312.5KHz data baud rate, set this signal to 96.

Error Definition 0x0044

Bit number	Read/ Write/ Default	
15..12	R/W/0	Error type: 0x0 – No Error – Register 0x46 used as Sync Pattern. 0x1 – Parity Error 0x2 – Word Length 0x3 – Bi Phase 0x4 – Sync Error 0x5 – Message Length 0x6 – Contiguous Data 0x7 – RT to RT Timeout/H009 drop clock 0x8 – Zero Crossing 0x9 – Bus Switching 0xA – Superseding message 0xB – Minimum Time to next message 0xF – Noise
9..8	R/W/0	Message number to insert the error. 0 to 3.
7	R/W/0	Auto Increment to next error injection location in case this register is not re written by the Host.
5..0	R/W/0	Word number to insert the error. 0 to 35.

Error Params 1 0x0045

Bit number	Read/ Write/ Default	
15..8	R/W/0	Error Value field: Unsigned if "signed" is not mentioned.
		Parity Error : This field is ignored.
		Word Length: 0x0 – Decrease 2 bits from specified word. 0x1 – Decrease 1 bit from specified word. 0x2 – Increase 1 bit to specified word. 0x3 – Increase 2 bit to specified word. 0x4 – Increase 3 bit to specified word.
		Bi Phase: 0x0 for bit 0 0x1 for bit 1 : 0xF for bit 15 0x10 for parity bit. In case Auto increment is selected, go from parity of Word N to first bit of Word N+1.
		Sync: 0x0 - 111100, 0x1 - 110000,

		0x2 - 111001, 0x3 - 011000, 0x4 - 000011, 0x5 - 001111, 0x6 - 000110, 0x7 - 100111, 0xF - Inverted of what is expected. 000111 ⇔ 111000
		Message Length: 6 bits, signed number in range of 31 down to -32. Value indicates the number of data words with respect to normal data words.
		Contiguous Data: Not required, see word-data in register 0x44.
		RT to RT Timeout: 6 bits. Effective range 4 to 63. Indicates how many half micro second gaps to insert between the 2nd command and the 1st status word. In H009 mode – After command and first 2 bits of received data, stop the clock.
		Zero Crossing: 0 for 1st half of bit 0 1 for 2nd half of for bit 0 2 for 1st half of bit 1 3 for 2nd half of for bit 1 : 38 for 1st half of bit 19 (parity bit) 39 for 2nd half of for bit 19 Zero Xing is inserted in one of the ½ bits of each word. This change would either expedite the arrival of the next zero Xing, or skew it away by the amount defined in bits 7..4 below. Note that Zero Xing in some of the bits might skew the zero Xing of the next bit. This will always happen during the sync in bit 0 and 2, and in 2nd half of bits that are followed by an opposite bit value.
		Bus Switching: <<Need to finish…>>
		Superseding message: <<Need to finish…>>
		Minimum Time to next message: <<Need to finish…>>
7..4	R/W/0	Zero Xing: Signed field in the Range of -8 to +7. -8 : skew next Xing by 8 * 1000/30 nano seconds = +266 ns. -7 : skew next Xing by 7 * 1000/30 nano seconds = +233 ns. -6 : skew next Xing by 6 * 1000/30 nano seconds = +200 ns. : 6 : expedite next Xing by 6 * 1000/30 nano seconds = -200 ns. 7 : expedite next Xing by 7 * 1000/30 nano seconds = -233 ns.
3..0	R/W/0	The expected response of the Unit under test for this message 0x0 – (CS) Clear status: Proper response, Status and datas. 0x1 – (NR) No response: The UUT should not respond. 0x2 – (NR) No response: The RT2RT Rx UUT should not respond. 0x3 – (MR) Message Error: A status with ME bit set. No datas. 0x4 – (MR) Message Error: A status with ME bit set. With datas.

		0x8 – CS or NR 0x9 – NR and when CS, stop incrementing. 0xA – CS and when NR, stop incrementing.

Sync Pattern 0x0046

The value of this register is compared with any command sync word either echoed back or received, and if they match, a rising edge pulse is set. On any following end of word that do not match this pulse is reset. Typically for the length for command word matching is 32 us pulse, with the rising of it accurately set.
If User Code 15 and 14 are both '1', the RS485 pins output these pulses.
Ch1 - Sync Pulse of Module 0 bus A.
Ch2 - Sync Pulse of Module 0 bus B.
Ch3 - Sync Pulse of Module 1 bus A.
Ch4 - Sync Pulse of Module 1 bus B.

Multi RIU Simulation enable Register Address 0x0047

This is a read and write register.
Any bit in register 0x0047 that is '1', enables one of the simulated of RIUs 1 to 15.
RIU 0 is not used in PP194, and is used as a reset mode.
Bit 0 of register 0x0047 enables Live RIU mode. When high, all RIUs that are ~~not~~ simulated, i.e. ~~'0'~~'1', become Live RIUs (instead of real RIUs in the regular mode). The Hardware has a single 16 bit counter for all Live RIUs. Each status word transmitted by any Live RIU gets a new incremented counter value and is entered into the message's status. The counter is reset each time Bit 0 of 0x0047 is reset to Zero.
If bit 0 is '0', then the status response is the second data word, which makes it identical to the first RIU response.

For example:
if bit 1 of register 0x0047 is '1' then RIU address 1 is enabled.
if bit 12 of register 0x0047 is '1' then RIU address 12 is enabled.
if bit 0 of register 0x0047 is '1' then Live RIU mode is enabled (real RIUs should be disconnected from the bus).

User Port 0x0048

This register allows the ability to set a register of 16 bits from the command line of the user software.
The 16 bit value of the user port will be sent by the SW to the HW.
It is advised that command line option –usercode XXXX be added to applications such as MuxSim. XXXX are 4 Hexadecimal characters.

Bit number	Read/ Write/ Default	What event triggers the interrupt when enabled by '1'.
15..14	R/W/0	"11" Use RS485 pins for Sync Pulses, see register 46 for details. "10" Use RS485 pins for debug of channel A. "01" Use RS485 pins for debug of channel A. "00" Use RS485 pins for 485 module.
8	R/W/0	MRT mode – Use the frame Time (0x4D) parameter as the bus idle time required to achieve Passive state. Usable if messages in the same frame are separated by more than 1 ms. Prevents Frame number from incrementing in the middle of a frame.
7	R/W/0	MRT mode – Ignore Word Count field when searching command in frame. Will not reply with ME even if WC do not match. This bit also exists in the Configuration Register 2 bit 7.
6	R/W/0	Frame Length Compensate – reduce frame length following a peak in delay of BC Go 1 when Windows lacks behind.
5	R/W/0	MRT mode – reply with ME for messages not in frame.
4	R/W/0	Read Time Monitor – User 48 bit time in monitor instead of 24 bits. Set as default by SW.
3	R/W/0	Unused
2	R/W/0	SWAP Bus B – used for SPI loader when 1553 signals are inverted.
1	R/W/0	MRT mode – Change module mode of operation Multi RT rather than BRT
0	R/W/0	H009 Mode - Change module mode of operation to H009

Interrupt enable Register Address 0x0049

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.
If a bit is not enabled, an interrupt will not occur, however, the General Status Register would set the appropriate bit to '1'.
Reading this register clears all bits.

Bit number	Read/ Write/ Default	What event triggers the interrupt when enabled by '1'.
10	R/W/0	Transmitter timeout occurred
9	R/W/0	Memory protection fault **
8	R/W/0	BC Retry – following incorrect RT response has been initiated.
6	R/W/0	Time tag counter rollover
4	R/W/0	BC End Of Message.
3	R/W/0	BC End Of Frame.
2	R/W/0	1553 message error occurred.
1	R/W/0	Wrong RT status or unexpected status bits set in status respond.
0	R/W/0	End Of Message.

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

** When Data pointer or Message State Pointer points to a reserved area (M0020 to M004F) due to wrong memory setup by the Host PC, the core sets bit 9 of the interrupt status register to '1'. The core's memory protection will also prevent the illegal write, thus memory integrity is maintained.
If this bit is not masked than an interrupt will occur.

Configuration Register Address 0x004A

Bit number	Read/Write/ Default	
15	Write/Read/'0'	'0' – generates a 500ns low pulse on the INTn signal. '1' – Level mode. INTn stays low until the host reads interrupt status register.
14	Write/Read/'0'	'1' – Loop back transmission in the FGPA, no bus transmission. '0' – Normal operation.
13	Write/Read/'0'	Mask / Compare the BCST bit in returned status bit. ** '1' – Mask. The relevant bit in the BC control word operates as mask for the BCST bit of the received status. '0' – Compare. The relevant bit in the BC control word is compared with the BCST bit of the received status.
12	Write/Read/'0'	'1' – Stop at end of Message error. If optional retry succeeded => messages continue!
11	Write/Read/'0'	'1' – Stop at end of frame if error. If optional retry succeeded => frames continue!
10	Write/Read/'0'	'1' – Stop messages if unexpected, non-masked status bits are set. If optional retry succeeded => messages continue!
9	'0'	'0' is preset for this bit.
8	'0'	'0' is preset for this bit.
7	'0'	'0' is preset for this bit.
6	Write/Read/'0'	'1' - PP194 enabled - only module 0 monitor is functional. PP194 Tx bus is mapped to module 1 busses. '0' – only 1553, both monitor modules are functional.
5	Write/Read/'0'	'1' - Message gap mode enable. Gap is defined in 1st word in stack. If enabled will start next message after gap*1us. If message gap is smaller than message length, will use default 10 microseconds inter-message gap. '0' – Rate mode. 1st word in each stack entry defines the rate of the message.
4..3	Write/Read /"00"	"00" – Global retry off. Do not retry. "01" – Retry once. "10" – Retry twice. "11" – Retry three times.
2	Write/Read/'0'	'0' – First & third retry on same bus if message failed. '1' – First & third retry on opposite bus if message failed.
1	Write/Read/'0'	'0' – Second retry on same bus as original failed message. '1' – Second retry on opposite bus of original message.
0	Write/Read/'0'	'1' – Retry a message if retry enabled and one of the unmasked status bits are set high in the returned RT status word. BCST bit pass/fail has a special treatment as seen above in bit 11 setup. '0' – No retry if status bits that are not masked are set.

Default Value: 0x0000.

** Note that the BCST bit in the status return is only set by the RT in the proceeding message's status word after a broadcast message FOR a transmit status or transmit command mode commands. Otherwise the BCST bit should be '0'.

Configuration Register 2 Address 0x004B

Bit number	Read/Write/ Default	
11..8	Write/Read/0	Cyber-attack mode: 0 – no Cyber-attack mode 1 – Time delayed attack mode with 65 ms steps of delay 2 – Time delayed attack mode with 100 us steps of delay 3 – Trigger Message delay mode
7	Write/Read/'0'	MRT mode – '1' - when searching if command exists, ignore Word Count field, bits 0 to 4. Also covered by USER CODE bit 7.

Cyber-attack mode 1 –

Frame counter loaded to delay BC start.
Delay is programmable to 0 to 2^16 steps of 65 ms
Once started, perform as BC frame (frame length not used, only message to message GAPS)

Cyber-attack mode 2 –

Same as mode 1 but with programmable steps of 100 us.

Cyber-attack mode 3 –

Frame counter loaded to delay BC start.
Frame counter incremented each time the bus command matches register 46 value.
Once found all occurrences, start frame normally from second message.

Reset & Stop Register Address 0x004C

This is a write only register. Reading back will read the value last written but this data should be ignored.

Bit number	Read/ Write/ Default	
7	Write	Writing '1' Clears the frame number register.
6	Write	Writing '1' Stops operation at end of message.
5	Write	Writing '1' Stops operation at end of frame.
4	Write	Writing '1' will insert Time Symbol into Monitor data stack.
3	Write	Writing '1' resets the Time Tag counter.
2	Write	Writing '1' resets the interrupt. Resets the FF that latches the interrupt condition. If the condition for interrupt persists, the relevant FF would be set again until the condition for causing this interrupt is cleared. Reset of Interrupt Register #1.
0	Write	'1' Reset core. Reset all registers, FFs in core. Memories are not reset, and should be reset by CPU.

Frame Time Register Address 0x004D

These 16 bits define the number of 100 us that is the frame length.
The frame duration can be set from 100 us (0x0001) to 6.55 seconds (0xFFFF).
The running value can be read from register address 0x0021.

Multi RT 0 to 15 enable Register Address 0x004E
Multi RT 16 to 30 enable Register Address 0x004F

This is a read and write register.
Any bit in register 0x004E that is '1', enables one of the simulated of RTs 0 to 15.
Any bit in register 0x004F that is '1', enables one of the simulated of RTs 16 to 30.
Bit 15 of register 0x004F enables Live RT mode. When high, all RTs that are ~~not~~ simulated, i.e. ~~'0'~~'1', become Live RTs (instead of real RTs in the regular mode). The Hardware has a single 16 bit counter for all Live RTs. Each word transmitted by any Live RT gets a new incremented counter value and is entered into the message's data-from-UUT as with real RTs. The counter is reset each time Bit 15 of 0x004F is reset to Zero.

For example:
if bit 0 of register 0x004E is '1' then RT address 0 is enabled.
if bit 1 of register 0x004E is '1' then RT address 1 is enabled.
if bit 15 of register 0x004E is '1' then RT address 15 is enabled.
if bit 0 of register 0x004F is '1' then RT address 16 is enabled.
if bit 14 of register 0x004F is '1' then RT address 30 is enabled.

if bit 15 of register 0x004F is '1' then Live RT mode is enabled, real RTs should be disconnected.

Supported Message Formats

The host PC should load the stack with the relevant words for the message.

The structure of the stack varies according to the type of message. The following table defines the stack contents types. Each entry is a 16bit word.

GR - Gap or Rate value depending on mode.
CONT – Message control word.
CMD – 1553 Command word.
CMD2 – RT2RT 2nd 1553 Command word.
STSrx – Status of simulated receiving data RT.
STStx – Status of simulated transmitting data RT.
DP – Data pointer
MP – Message pointer
BCST – Broadcast

Message Findings Entries:
TT – Time Tag or frame number
MF – Message findings
STSrx – status of ***non*** simulated RTs picked up from the bus.
STStx – Second status received from ***non*** simulated RTs picked up from the bus.

Color-coding:
Blue – Simulated RT (SimRT).
Red – Non Simulated RT (RealRT).
~~Strikethrough~~ – this word is not used in this message format.

Type	Data Stack contents		Message Findings
BC => RealRT	GR CONT CMD ~~CMD2~~ ~~STS1~~ ~~STS2~~ DP MP		TT MF STSrx ~~STS2~~
BC => SimRT	GR CONT CMD ~~CMD2~~ STSrx ~~STS2~~ DP MP		TT MF ~~STS1~~ ~~STS2~~
RealRT => BC	GR CONT CMD ~~CMD2~~ ~~STS1~~ ~~STS2~~ DP MP		TT MF STStx ~~STS2~~
SimRT => BC	GR CONT CMD ~~CMD2~~ STStx ~~STS2~~ DP MP		TT MF ~~STS1~~ ~~STS2~~
RealRT => RealRT	GR CONT CMD CMD2 ~~STS1~~ ~~STS2~~ DP MP		TT MF STStx STSrx
RealRT => SimRT	GR CONT CMD CMD2 STSrx ~~STS2~~ DP MP		TT MF STStx ~~STS2~~
SimRT => RealRT	GR CONT CMD CMD2 STSxt ~~STS2~~ DP MP		TT MF STSrx ~~STS2~~
SimRT => SimRT	GR CONT CMD CMD2 STStx STSrx DP MP		TT MF ~~STS1~~ ~~STS2~~
BC => RealRT Mode	GR CONT CMD ~~CMD2~~ ~~STS1~~ ~~STS2~~ DP MP		TT MF STSrx ~~STS2~~
BC => SimRT Mode	GR CONT CMD ~~CMD2~~ STSrx ~~STS2~~ DP MP		TT MF ~~STS1~~ ~~STS2~~
RealRT => BCST	GR CONT CMD CMD2 ~~STS1~~ ~~STS2~~ DP MP		TT MF STStx ~~STS2~~
SimRT => BCST	GR CONT CMD CMD2 STStx ~~STS2~~ DP MP		TT MF ~~STS1~~ ~~STS2~~
BC => BCST	GR CONT CMD ~~CMD2~~ ~~STS1~~ ~~STS2~~ DP MP		TT MF ~~STS1~~ ~~STS2~~
BC => BCST Mode	GR CONT CMD ~~CMD2~~ ~~STS1~~ ~~STS2~~ DP MP		TT MF ~~STS1~~ ~~STS2~~

Word Monitor Memory Contents

Each word captured on the bus occupies 2 words in memory. The 16 bits of the MIL-STD-1553 word is stored in the even address of the two, while a descriptor word follows on the odd address.

Hex Address	Description	Comments
2000	1553 word N	16 bits of 1553 word
2001	Word N descriptor	Further describing word N.
2002	1553 word N+1	16 bits of 1553 word
2003	Word N+1 descriptor	Further describing word N+1.
:		
2FFE	1553 word N+2047	16 bits of 1553 word
2FFF	Word N+2047 descriptor	Further describing word N+2047.

Each pair of data word and descriptor is called Symbols.
In addition to the data Symbols, there are real time Symbols.

The Data Symbol

The data symbol is composed of the 16 bits from decoded from the bus, followed by 16 bits of descriptor that describe the first 16 data bits.

The descriptor word contains the following fields:

Bit number	Name	Description
15..8	Gap Size	Number of ½ us from the previous word. 40 (0x28) for contagious data. Less than 40 when switching from one bus to another, 60 for a 10us gap, and up to 127.5 us (0xFF) for gaps of 127.5 us and up.
7	'1'	Always '1'.
6 *	'1'	'1' - for gaps up to 127.5 us. '0' - for time symbols, see next section.
5	X	Unassigned.
4	Error	'1' – if any word error encountered. (Sync, Manchester, Parity, Wrong H009 Gaps (5us or >= 8us))
3	Sync type	'1' – Command or Status sync. '0' – Data sync.
2	Channel	'1' – Word received on channel B. '0' – Word received on channel A.
1	Gap Flag	'1' – No gap (less than ½ us) before this word. Ignore Gap Size. '0' – A gap was found. See Gap size for actual gap from previous symbol. '1' – H009 mode: H009 Clock Ok. '0' – H009 mode: H009 Clock not Ok.
0	PP194_n1553	'0' – MIL-STD-1553 or H009 standard detected. '1' – PP194 Standard detected.

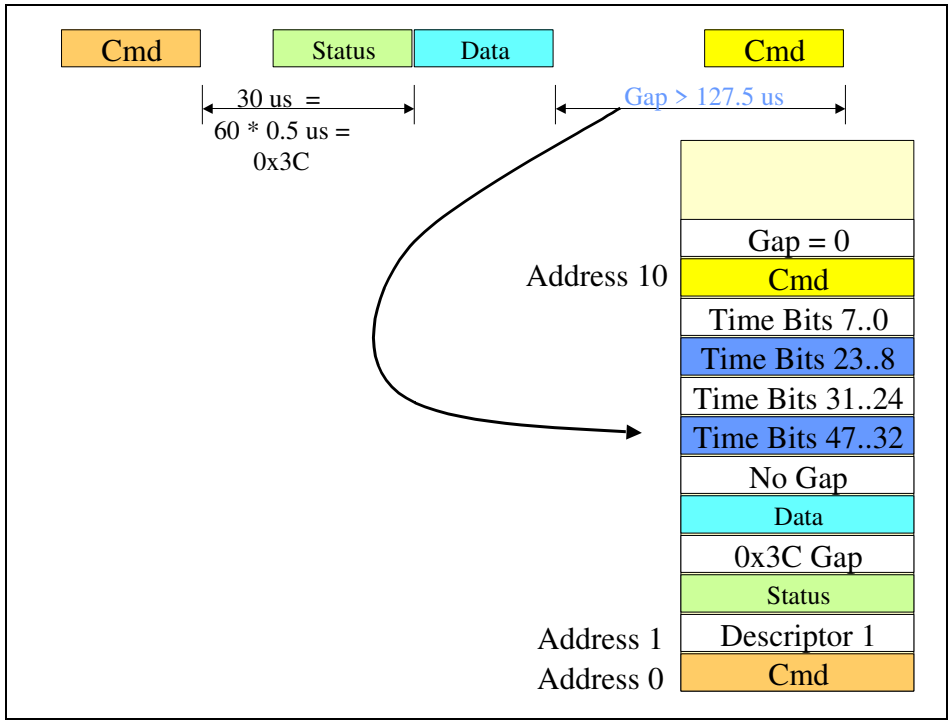
The Time Symbol

There is a 48-bit counter inside the Word Monitor. The counter is incremented every ½ us. The counter would roll over after 4 years…
The counter is reset with the hardware reset or with the software reset.
There is one counter for both word monitors, and its reset with the SW reset on either modules.
If there is a gap bigger than 127.5 us, the word monitor inserts the time symbols upon reception of a new data symbol. So a data symbol arriving more than 127.5 us from a previous data symbol would result in writing two time symbols followed by this gapped data symbol.

There are two time symbols possible. The MSBs time Symbol and the LSBs time symbol. The word monitor engine inserts the MSBs time symbol first and then the LSBs time symbol. If only one of these symbols exist, it's an error. The Software should ignore a time symbol if it does not come in pair.

The 48 bits spread on both time symbols are real time since the last reset and are not relative as in the 8 MSBs of the data symbol descriptor.

The following example describes the operation of the word monitor in case a gap is larger than 127.5 us between two data symbols.

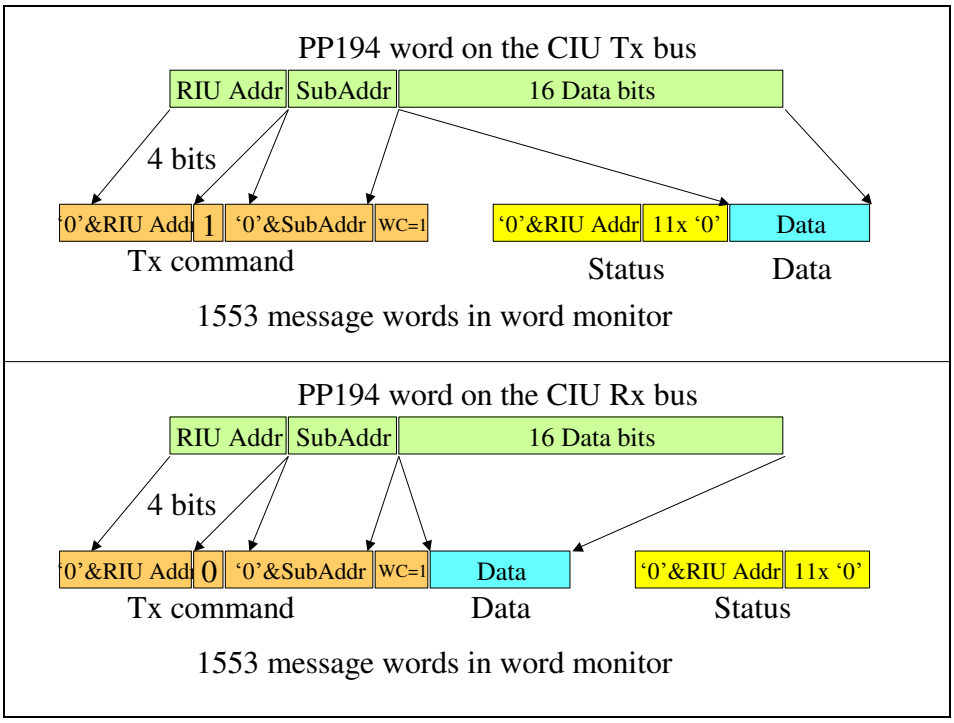


A 48 bit time mark in stack is composed of the two symbols:

Bit number	Name	Description
15..0	MSBs	Bits 47..32 of the 48 bit counter that counts the number of ½ us since the last reset.
Bit number	Name	Description
15..8	Gap Size	Bits 31...24 of the 48 bit counter that counts the number of ½ us since the last reset.
7	'1'	Always '1'.
6 *	'0'	'1' - for gaps up to 127.5 us. '0' - for time symbol.
5..4	Time Symbol Type "10"	"11" – Reserved for future use "10" – 24 MSBs of 48 bit time symbol "01" – 24 LSBs of 48 bit time symbol "00" – Larger than 127.5 gap time symbol (legacy mode).
Bit number	Name	Description
15..0	MSBs	Bits 23...8 of the 48 bit counter that counts the number of ½ us since the last reset.
Bit number	Name	Description
15..8	Gap Size	Bits 7...0 of the 48 bit counter that counts the number of ½ us since the last reset.
7	'1'	Always '1'.
6 *	'0'	'1' – Data Symbol with gap up to 127.5 us. '0' – Time symbols..
5..4	Time Symbol Type "01"	"11" – Reserved for future use "10" – 24 MSBs of 48 bit time symbol "01" – 24 LSBs of 48 bit time symbol "00" – Larger than 127.5 gap time symbol (legacy mode).

PP194 words organization in Word Monitor

Each PP194 word is composed of 24 bits + sync + parity. The 24 bits are mapped to a legal MIL-STD-1553 message in order to fit the 16 bit memory size and ease on the Message Reassemble.
The mapping is described in the following diagram.



A PP194 word on the Tx bus will be mapped to a three word 1553 message describing a transmit command with 1 data word.

A PP194 word on the Rx bus will be mapped to a three word 1553 message describing a receive command with 1 data word.

Note that the only non-1553 similarity is the gap time between the 3 1553 word messages representing each PP194 word. The command word entry is the first entry and its time tag is equal to the time from the previous word. The other data and status words have a gap time of 0. Message reassemble software should be aware of the difference.

All PP194 word's descriptors have their LSB set to '1', and in all 1553 words its set to '0'.

H009 words organization in Word Monitor

H009 protocol supports three types of messages: BC to RT, RT to BC and BC to RT mode code.
The H009 words are saved in the word monitor as they flow in, without any modification (as in PP194).
The H009 command word format is as follows:

Bit #	Field	Purpose
15..12	RT address	16 RTs are supported
11	Command Indicator	1 bit. If '1' this is a H009 mode command, with one word following command.
10..5	Sub Address	6 bits of Sub address supporting 64 sub addresses.
4	Transmit	'1' – its an RT transmit command, '0' – an RT receive command.
3..0	Word Count	0 to 15 words. 0 should not exist, but supported.

H009 data word is a 16 bit data.

In the descriptor, the 8 MSBs define the time since the last word, i.e., gap time. Its resolution is 1/2us.

For 1553 the data words are separated by 20 us, thus the value for data words is typically 40 or 39 (0x28 or 0x27), but for H009, valid data words are those that have 5 us gap from previous word. Since each word is 17 us long, the GAP is expected to be 22 us, thus in ½ us resolution it would be 44 (0x2A).

The Gap value should be used for time axis tracking and not for data type validity, the HW would tag a data word as valid if it had 5 us gap, else would tag this word as error. Command sync would be set for words with more than 8 us gap before them.

Since the gap defines the sync type, the specific Gap Flag (bit 1 of the descriptor) is used for other indication. It indicates if the H009 accompany clock was working Ok when the specific data was word was saved. A logic '0' here indicates an error in that clock, a '1' indicates the clock was Ok.

Appendix A: Gap/Rate mode

Definition.

Typically MIL-STD-1553 ("MuxBus") has a frame time definition.
The frame is a period of time, typically 10 or 20 milliseconds long.
Several messages are transmitted every frame. These messages manage the system.

In more complex MuxBus systems, not all messages are transmitted every frame. For example, the direction that a Radar Antenna is pointing at, should be transmitted every frame, i.e. 50 times a second, for display units to display target position. On the other hand, button position on one of the panels can be transmitted twice a second, since its not practical that the pilot would press that button faster than that.

The Operational Flight Program (OFP) programmer tailors the frames based on the Interface Control Document (ICD) that defines all message types required. In the ICD, each message is tagged by its usage rate.

Existing sequencing mechanisms.

For example, let's define a system with a rate of 50 frames per second (50 Hz).
Message A50 and B50 are transmitted every frame.
Message E25 is transmitted every 2nd frame.
Message G125 is transmitted every 4th frame.

A possible order of the system would be:

Frame #1	Frame #2	Frame #3	Frame #4
A50 B50 E25	A50 B50 G125	A50 B50 E25	A50 B50

Frame #5 repeats frame #1 and so on…

As exampled, message G125 is transmitted in the frame that does not serve 25 Hz messages. This is done for load balancing as explained below.

Existing Bus Controllers (BC) use a stack of message entries to control their message sequencing. The Host CPU updates the stack, and initiates the bus controller to execute the messages automatically and autonomic.
Each Stack entry points to a location in its memory where the actual message command and words are stored.
Existing BCs define minor and major frames. In the above example, there is one Major Frame that is 4x20ms => 80ms, and 4 minor frames, each one take 20ms.

The stack would look like this: A50 B50 E25 A50 B50 G125 A50 B50 E25 A50 B50.
This list would be transmitted every 80ms, BUT there need to be a tool to force a gap between the end of E25 of frame #1 and A50 of frame #2 in order to make frame #1 20ms

in length. So, existing BCs also hold in their stack a 'message-length' parameter. So the stack would now look like (message-length in parenthesis):
A50 (1ms) B50 (1ms) E25 (18ms) A50 (1ms) B50 (1ms) G125 (18ms) …
Message-length parameter in stack is a way of composing a minor frame of 20ms.

This technique starts falling apart when lower rate messages have to be sequenced. If the slowest message is 50Hz/64 => 0.78 Hz then a complete list of more than a second has to be stacked. Most of the stack entries point to the very same message, it's simply a very big stack.

Sital Technology's enhanced BC Sequencing.

Message Sequencing
The minor-major frame mechanism described above is not a systolic design. Systolic design is such that maintains its size as complexity rises. Following is a systolic solution for the same requirement.

Instead of the message-length stack entry for each message, Sital's BC sequencing requires a stack entry for each message that specifies the rate of that message. For practicle reasons, the rate would be expressed as ½, ¼,… (negative power of two) of the fastest rate. If the frame rate is 50Hz, then ½ of it would be 25Hz, ¼ of it would be 12.5Hz, and so on.

Message sequencing would be automatically assigned as seen in the following table:

Frame Rate	Frame Hertz	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	50																		
½	25																		
¼	12.5																		
1/8	6.25																		
1/16	3.13																		

In practice, message of rate 1/N where, N is a power of 2, would initially be transmitted in frame #N/2-1, then in N+N/2-1, then in 2N+N/2-1…
As seen, a message with rate 1/8, would be transmitted in frame #3, #11, and cyclic onwards.

Message Skewing.
As seen above, each frame accommodates only the 50 Hz rated messages, and one, and only one, of the other rates, but not more than two rate types. This characteristic is very useful, and spreads the bus load evenly. Well, not always!
There could be many messages in the ICD that are running at 25Hz. There could be so many and long 25Hz messages that along with all the 50Hz messages, produce a frame longer than 20ms (frame #0, #2…). On the other hand, the following frame, #1, serves the 12.5Hz messages that could be very short. So you would want to be able to skew some of the 25Hz

messages from frame #0 to frame #1. These messages would also be transmitted every two frames, but in the even frames rather than the odd frames in the above example.
For lower rate frames, the frame skew can be 1, 2, 3 or even more frames. Sital's enhanced BC sequencing allows for up to 15 frames skew.

Non power-of-2-messages rate.

If the ICD defines a message with 40Hz rate, and the frame rate is 50Hz, this rate can be achieved. Simply add the message to the stack a number of times. Once with rate 25Hz, once with rate 12.5Hz, one with rate 3.13Hz. All together the message would be transmitted at the above, pink, green and the blue marks which would be close to 40.63 Hz.

Summary

The above rate definition and positioning along with the optional frame skewing delivers an exact, predefined message list for each frame, by specifying two simple parameters that are derived from the ICD definition.

Appendix B: Sample memory setup

Example 1 – used in VHDL simulation testing

This appendix examples a memory setup for a small system with 2048x16 memory space.
This memory example assumes that there are up to 40 messages that the system is supposed to support. Please replace 40 with actual number in bigger systems.

The memory blocks are:
All addresses in Hex.

Start Addr	End Addr	Contents
0	1F	Reserved
20	2F	State Registers – findings
30		Initial Stack Pointer
31		Initial Stack Length
32		Current Stack
33		Stack Remaining
34		START command
40	4F	Register setups – configurations
50	18F	Stack- 40 entries of 8 words each
190	22F	40 Message State blocks
230	72F	Data Blocks both directions, 0x230 + 32*i (i::=0..39)

Color-coding:
Blue – fields updated by the Host PC
Green – fields updated by the core – read by the PC.
Black – not used could be updated by Host before Run command.

Example 2 – This is the memory setup used by SW ver 1.3

This second example is for a memory setup of a system with 2048x16 memory space.
This memory example assumes that there are up to 32 messages that the system is supposed to support. Please replace 32 with actual number in bigger systems.

The memory blocks are:
All addresses in Hex.

Start Addr	End Addr	Contents
0	1F	Reserved
20	2F	State Registers – findings
30		Initial Stack Pointer
31		Initial Stack Length
32		Current Stack
33		Stack Remaining
34		START command
40	4F	Register setups – configurations
50	14F	Stack- 32 entries of 8 words each
150	54F	Data to UUT – 32 blocks of 32 words (Assume half transmitted)
550	5CF	32 Message State blocks
5D0	9CF	Data received from UUT – 32 blocks of 32 words (Assume half are received)

Color-coding:
Blue – fields updated by the Host PC
Green – fields updated by the core – read by the PC.
Black – not used could be updated by Host before Run command.

Example 3 – 64 messages memory usage

This example is for a memory setup of a system with 4096x16 memory space.

The memory blocks are:
All addresses in Hex.

Start Addr	End Addr	Contents
0	1F	Reserved
20	2F	State Registers – findings
30		Initial Stack Pointer
31		Initial Stack Length
32		Current Stack
33		Stack Remaining
34		START command
40	4F	Register setups – configurations
50	24F	Stack- 64 entries of 8 words each
250	A4F	Data– 64 blocks of 32 words
A50	B4F	64 Message State blocks of 4 words each

Color-coding:
Blue – fields updated by the Host PC
Green – fields updated by the core – read by the PC.
Black – not used could be updated by Host before Run command.

Example 4 – Maximum memory usage

This example is for a memory setup of a system with 4096x16 memory space.

The memory blocks are:
All addresses in Hex.

Start Addr	End Addr	Contents
0	1F	Reserved
20	2F	State Registers – findings
30		Initial Stack Pointer
31		Initial Stack Length
32		Current Stack
33		Stack Remaining
34		START command
40	4F	Register setups – configurations
50	320	Stack- 90 entries of 8 words each
320	E60	Data– 90 blocks of 32 words
E60	FC8	90 Message State blocks of 4 words each

Color-coding:
Blue – fields updated by the Host PC
Green – fields updated by the core – read by the PC.
Black – not used could be updated by Host before Run command.

Appendix C: MultiComBox MultiRT standalone

Version 2 - April 20th 2009

Target

The purpose of this paper is to allow MultiComBox users use the MultiRT Standalone mode, referred to as MRT mode in this document.
It is assumed that the reader is familiar with MultiComBox' API programming or MuxSim setting, and BC/MultiRT simulation. The BC/MultiRT mode is referred to as BRT mode.
MRT was developed to allow real time emulation of multiple RTs for complex bus simulations by commonly used computing power through USB 2.0 channel.

Fundamentals

MRT is used when a remote bus controller (BC) manages the bus and MultiComBox emulates the RTs on the bus.
It is assumed that the remote BC runs in frames, and that each frame has got at least 4ms of dead bus time.
MRT mode was developed as an additional piece of Hardware (HW) inside MultiComBox with the intension that the same Software (SW) written for BRT is used for MRT. What distinguishes between BRT and MRT is the introduction of MuxSim's UserCode (*SetUserPort in API*) parameter before operation starts.

Enabling MRT

In order to enable MRT mode, please invoke MuxSim with –UserCode=0002 or –UserCode=0022 (or *SetUserPort* in API*)*. User code 0002 activates MRT and disables BRT for both modules. User code 0022 enables MRT and disables BRT for both modules and also treats messages that are not in the active frame as illegal message, i.e., ME bit set in status return and no data transmitted.
Using TesterDLL.dll, each module is controlled by *SetUserPort* API function separately; hence it is possible to run one module in BRT and the second in MRT.

MRT in HW

MRT HW is a compact block very similar to Sital Technology's space qualified RT1553FE validated RT that was upgraded to support any number of RTs in the same time, and to allow RT to RT messages when both RTs, one of the RTs, or non of the RTs is enabled in the MRT. It is expected that if one RT is enabled, the MRT passes RT validation.
The MRT backend block that communicates with the USB interface is designed to the same look-and-feel as the backend of the BRT, so that the SW built for BRT is identically used for MRT although it's built of a completely different HW.

MRT and BRT differences

MRT should be run forever as opposed to BRT, which is typically run once, updated and rerun – repeated for as long as required.
The BRT transmits the messages in one continuous "active" time and performs its data exchange with the USB port at the "passive" time. The BRT HW obviously actively manages these two time periods.
The external bus controller (BC) defines the active and passive times in the MRT mode and the MRT syncs with that BC by detecting bus activity. When the bus is active, the MRT signals the USB that it is in the "active" time, and after 1 ms of dead bus (A and B) it signals the "passive

time" until the first message starts the next "active" time. In MRT the frame period is thus less important, and is derived from the activity on the bus.

Typical MRT SW API

SW developed for MRT using MultiComBox API typically follows the following stages:

1. Setup one or more messages.
2. Characterize each message.
3. Setup one or more frames of messages.
4. Characterize each frame.
5. Enable the RTs that need to be simulated.
6. Loop the following steps for as long as needed:
    a. Activate one of the frames to go (BC_GO 0) go forever!
    b. Read the results of the messages of the previous frame.
    c. Calculate and update the messages' data for next frame.
    d. Update the frame.
    e. Wait for the frame to complete.
    f. Repeat the loop.

SW stages for MRT using MultiComBox API are almost the same as for BRT. This paragraph describes how the MRT does each one of the steps as in the BRT.

In steps 1 through 3 – do the same. Setup all of the expected messages (up to 64 messages are supported) to a frame (No need to specify bus A or B, MRT answers both). Expected messages can be any valid 1553 message including BC2RT, RT2BC, RT2RT, Mode2RT, RT2Mode, Broadcast2RTs and Broadcast mode commands.
No need to specify the status words' contents.
Step 4 – It is not important to *SetFrameTime* because an external BC defines the frame size and rate, it is advised to set it to the actual bus period for future use.
Step 5 – Enable RTs for simulation. It is essential to enable the RTs that the MRT is emulating. The MRT replies to messages who's RTs are enabled.
Step 6a – BC_Go command is required even though it does not trigger transmission. BC_Go command enables MRT to listen to the buses then searches for bus dead time (1 ms of no activity), and after dead time has been detected, MRT waits for the first message. When this message arrives, it is assumed that a new frame starts.
Before BC_Go instruction, the MRT does not reply to any message!
In MRT mode it is strongly advised to BC_Go 0 (forever) and not BC_Go once as in BRT.
Step 6b – relevant for the second loop. Read the results received in the previous frame.
Step 6c,d – Calculate and load the data for the messages for the next frame.
Step 6e – In BRT mode, when the frame is finished, the bus is said to enter the inactive state, or passive time. This passive time is reported by the HW and detected by the SW, and allows the SW to read the results of the previous run and load fresh data for next frame. In MRT mode, the HW detects passive time by sensing bus inactivity of at least 1 millisecond. After a 1ms of inactivity passive time state is reported to the SW, and the same data exchange is performed as in BRT.
Step 6f – If MRT is used for dynamic bus emulation, i.e., change the message data on every frame, BC Go 0 (forever) should be used. The SW should be highly responsive because if a new frame is started by the remote BC, and BC Go has not triggered, the MRT will NOT reply to any of the messages.

Non pure "Real Time Operating Systems" (RTOS) such as Windows XP, might not be able to assure this for by 100%. For non-RTOS SW it might be better to set BC_Go 0 (infinite) to assure reply by the emulated RTs in all frames, and the SW updates the frames as much as it can.

Appendix A: Emulation options

1. If an RT or several RTs are enabled for simulation but a message on the bus are not part of the frame –
    a. If UserCode=0002 is used, the message is responded correctly by the MRT, and if it's a transmit message, the data payload is set to all zero, or incremented by 1 for each word if live RT is set. Receive command's payload is dumped.
    b. If UserCode=0022 is set, any message received for an enabled RT that is not on the frame list will be treated as illegal message. Illegal transmit messages will be replied with no data and ME bit set in status return (bit 10). Illegal receive commands' data is not saved, and the status reply is ME.
2. MuxSim setup possibilities:
    a. The simple setup would be to enable the desired RTs for emulation, set the expected frame time length, set "run forever", and press run. No messages are set in the frame (use the default Broadcast to BC) and UserCode=0002 is used. Messages will be replied with no error, with zero data (or increments if Live RT is set).
    b. Simple Setup as above but with the desired messages in the frame:
        i. MuxSim will report message error (no end of message bit set) as long as the message has not been on the bus.
        ii. Once the message occurred for the first time, error count is stopped, and the number of messages is reported.
3. RT to RT messages:
    a. Once an RT2RT message is received, the MRT HW searches this message, both in the first command and in the second command and replies for the transmitting RT or the receiving RT depending on the RT simulation setup.

Appendix D: User Code modes

It is possible to instruct the Hardware to do some activities different than the hardware described in this document with the use of User Code register.
An API function to set User Port access the User Code register and loads it with value.
A user Code register exists for each channel.

The following values define the special working mode:

For list of User Codes, please refer to User Code register.

For example:

If module 1 has to do RS485 instead of 1553 then set the UserCode=0008
If module 0 has to swap bus B for SPI programming set UserCode=0004
If module 0 has to return on all messages of simulated RTs set UserCode=0002
If module 0 has to return ME for messages not in the frame set UserCode=0022
If the windows XP takes the control from our application, and a new BC Go command is issued AFTER the last frame ended, the next frames would be 100 us shorter until they catch up with the rate if there was delay.

User Code bits 15 downto 12 define the operation of the 4 RS422 outputs. When these bits are all zero, the RS422 outputs are normal.

Module 0 MIL-STD-1553 Bus A on 4 RS422 lines:
UserCode=1xxx –Tx on RS422 channel 0.
TxN on RS422 channel 1.
Rx on RS422 channel 2.
RxN on RS422 channel 3.
In this mode of operation, the Rx and RxN have to be connected to a live RS422 driver, otherwise if left unconnected would probably generate "loopback error".

User Codes:
User(0) => H009 instead of 1553
User(1) => MultiRT
User(2) => Swap Bus B
User(3) => --UART RS485 - not from v5.2
User(4) => Real time in monitor
User(5) => MultiRT ME for Msg not in list
User(6) => Compensate Frame length for slow peaks
User(7) => Ignore WC in MRT scanning of new command
User(15..12) => RS485 pins usage : See list below.

User Code specific for RS422 pins:

```
IF USER0(15 downto 12)="1100" THEN ---Sync and debug mode
    DI_CH(1) <= M0_SyncPulseA;
    DI_CH(2) <= M0_SyncPulseB;
    DI_CH(3) <= M1_SyncPulseA;
    DI_CH(4) <= M1_SyncPulseB;
    DE_CH <= x"F";
ELSIF USER0(15 downto 12)="1010" THEN ---Error insertion sync pulse
    DI_CH(1) <= Scope_Error_pulse_M0;
    DI_CH(2) <= M0_SyncPulseA;
    DI_CH(3) <= Scope_Error_pulse_M1;
    DI_CH(4) <= M1_SyncPulseA;
    DE_CH <= x"F";
ELSIF USER0(15 downto 12)="1000" THEN ---Bus A debug mode
    DI_CH(1) <= to_std(not H15530_Record_A.CS);
    DI_CH(2) <= RxA;
    DI_CH(3) <= RxAN;
    DI_CH(4) <= to_std(H15530_Record_A.recieving);
    DE_CH <= x"F";
ELSIF USER0(15 downto 12)="0110" THEN ---Sync and debug mode
    DI_CH(4) <= WEN;
    DI_CH(3) <= CMD_DATA;
    DI_CH(2) <= REN;
    DI_CH(1) <= CMD_DATA OR REN;
    DE_CH <= x"F";
ELSIF USER0(15 downto 12)="0100" THEN ---Bus B debug mode
    DI_CH(1) <= to_std(not H15530_Record_B.CS);
    DI_CH(2) <= RxB;
    DI_CH(3) <= RxBN;
    DI_CH(4) <= to_std(H15530_Record_B.recieving);
    DE_CH <= x"F";
ELSIF USER0(15 downto 12)="0010" THEN ---USB debug
    DI_CH(1) <= WEN;
    DI_CH(2) <= CMD_DATA;
    DI_CH(3) <= REN;
    DI_CH(4) <= CMD_DATA OR REN;
    DE_CH <= x"F";
ELSIF USER0(15 downto 12)="0001" THEN ---1553 on RS422 bus
    DI_CH(1) <= Tx;
    DI_CH(2) <= TxN;
    DI_CH(3) <= '0'; --Rx line
    DI_CH(4) <= '0'; --RxN line
    DE_CH <= x"3";
    r0  <= RO_CH(1); --echo of Tx
    r1  <= RO_CH(2); --Echo of TxN
    r2  <= RO_CH(3); --Rx
    r3  <= RO_CH(4); --RxN
ELSE --UART mode
    DI_CH(1) <= t0;
    DI_CH(2) <= t1;
    DI_CH(3) <= t2;
    DI_CH(4) <= t3;

    DE_CH(1) <= t_en0;
    DE_CH(2) <= t_en1;
    DE_CH(3) <= t_en2;
    DE_CH(4) <= t_en3;

    r0  <= RO_CH(1);
    r1  <= RO_CH(2);
    r2  <= RO_CH(3);
    r3  <= RO_CH(4);
END IF;
```

Appendix E: RS422/485 working flow

For the API command of RS422 setup:
Please load configuration registers 0x40 and 0x41, 0x42 or 0x43 according to the module and line number. This is done per module.

For the API send RS422 command:

1. Check register 0x2D for line 0 or 0x2E for line 1 bit 11 (0x800) if it is high.
    a. If it is it means that the line is still transmitting words. Reply with a busy error to the user.
    b. If its 0, go ahead and continue.
2. Check if the length is less than or equal 1008 words.
    a. If bigger reply with error that maximum length is 1008.
3. Write words to buffer.
4. Write number of words to offset address 1023 (0x03FF) of the buffer. Base address is different for all 4 buffers. See memory mapping.

From that point on the hardware will transmit the words to the bus.
While transmitting, bit 11 of 0x2D or 0x2E is set to logic high.

For the API Number of Rx Words Pending RS422 command:

1. Check register 0x2D for line 0 or 0x2E for line 1, bits 0 to 9 for the pointer of next available place.
2. Compare to the SW pointer.
3. The difference (mind the cyclic condition) is the number of words that are in the HW but not read yet.

For the API Get Words RS422 command:

1. Check register 0x2D for line 0 or 0x2E for line 1 bit 0 to 9 for the pointer of next available place.
2. Calculate the number of available words.
    a. If the number of words requested in the command is bigger than the available words return error – not enough words for reading.
3. Read the buffer from the hardware and send to customer.

Appendix F: EBR1553 mode

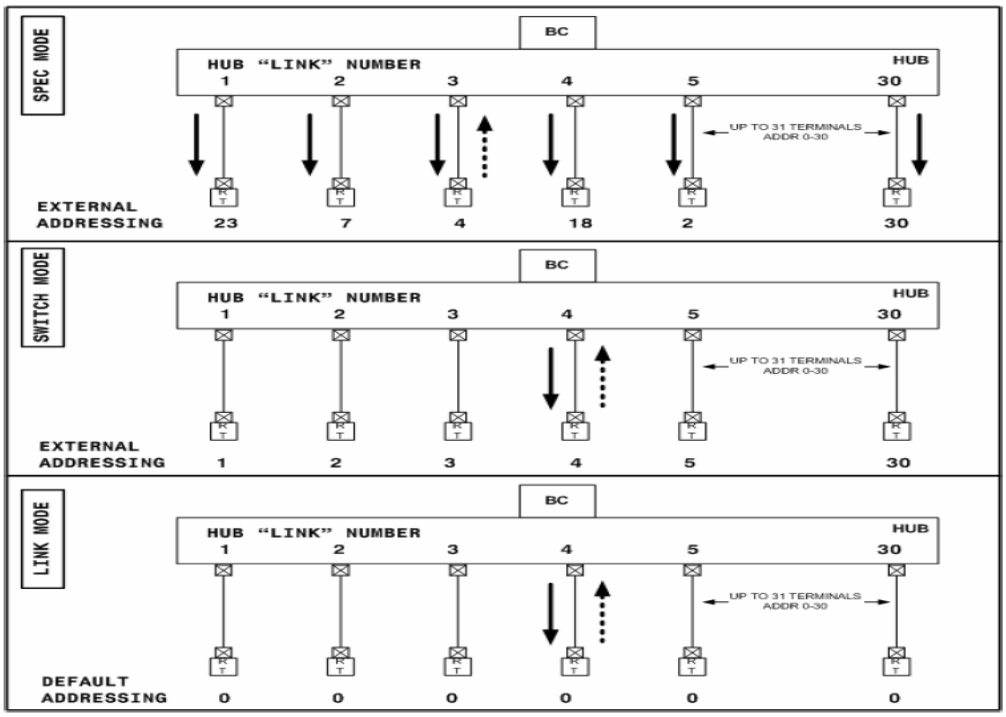
What is EBR1553?
Enhanced Bus Rate (EBR) is a special 10Mhz 1553 link. http://www.sitaltech.com/EBR1553D.asp

What is different with MultiComBox (MCX) in EBR1553 mode?
In EBR1553 mode MCX loads a different firmware to communicate through the RS485 lines instead of through the standard 1553 transceivers. No standard transceiver signaling activity is possible. No standard RS485 serial communication activity is possible.
MCX supports a HUB of 4 lines, and can act as a BC+MultiRT, or MultiRT stand-alone mode, just like the regular 1553 mode.
Error injection is also supported.

How do I set the HUB mode?
The HUB modes are set by the USER code bits 13 and 12 as follows:

"00" SPEC mode – BC transmits on all lines, MultiRT answers on the line based it's RT address.
"01" SWITCH mode – BC transmits on the Line x to the RT x. MultiRT replies on line x.
"1X" Link – BC transmits a message for RT x on line x, and change command to RT0. MultiRT changes RT0 to RT x based on the line it received the command, and answers on x with RT0.

Example:
For MultiRT link mode write 0x2002 to User code. (bit 1 is MultiRT selection).
For BC+MultiRT switch mode write 0x1000 to user code.
For BC+MultiRT spec mode write 0x0000 to user code.



Appendix F: pTDR1553 memory

The Smart Wiring Passive TDR block allows the detection of 1553 wiring faults during the operation of the bus.
The list includes tails of messages detected on bus A and On Bus B.
In transmit messages, the BC tail signature is captured, while in receive messages, or mode commands without data, the RT's tails are listed.
The list includes 28 entries for RTs on both bus A and B.
The base address for the pTDR1553 in GRIP2 is from address 0x8000. The memory contents offset from this address is described in the following table:

Address (Hex)	Contents	
0	Version number	Date code
1	7..0 – Distance threshold	Smaller than that – no bus fault, higher is a bus fault
2	number of messages since first interrupt	0 to 64K-1.
3	Time of last write	Counts # of 0.1sec from reset
6	Bit 0 – '0' is SCAN mode – collect tails data '1' – track mode – verify no TDR, else interrupt Bit 8 – '1' Flash enable mode, else no flash.	Writing to this register 0 will prevent interrupts.
7	When reading: xFFFF – No fault detected from reset x"BAD2" Massive bus fault, probably missing 78 Ohm x"BAD1" Active bus fault Bus fault, x"BAD0" Bus Ok, fault detected previously.	Write - SW reset by writing x"9876" – acquire new values.
Fix Addresses: x8 to xB is the entry of the BC on bus A		
Fix Addresses: xC to xF is the entry of the BC on bus B.		
x10 to x7C	Bits 15..11 RT address Bit 10 – '1' for Bus A, '0' for B (x400) Bit 9..0: other bits of status response	Mod 0 Address X"FFFF" indicates empty slot
x11 to x7D	Status word: Bit 12 – '0' Last write was Mod2, '1' was Mod3 Bit 11..4 – Number of writes of Mod3 tail, up to 255. Bit 3..0 – Number of writes of Mod2 tail, up to 15.	Mod 1 Address
x12 to x7E	Bit 15..0 – Number of nanoseconds of length of 1553 word If clocks (combined) are 128Mhz equivalent, then ideal signal is 20x1024=> 0x5000	Mod 2 Address "Ok Tail" Value
x13 to x7F	Bit 15..0 – Number of nanoseconds of length of 1553 word For non 128Mhz: (Mhz x 8) x 20	Mod 3 Address "Bad Tail" Value

Appendix G: Cyber Attack Mode

Cyber-attack mode is enabled by writing a non-zero value to bits 8...11 of configuration register 2 address 0x4B.
In this mode the transmission is delayed until the predefined scenario is met, and when it starts, it repeats based on the setting.
The frame time value and the message gap value set for each frame run is analyzed differently for each attack.
See more details in "MCX Cyber Attack function.pdf" document.

Attack type 1: time delayed attack. Frame length counter now counts up to 64K of 64K us (~65ms) allowing a delayed attack of up to 1 hour and 10 minutes. Once attack starts, the frame runs as in BC mode with the exception that after the last message, the machine does not wait for the frame counter, but rather starts the frame again (if thus defined for the frame). To achieve delay between one frame and another, the message gap counter can be used to postpone the end of the frame.

Attack type 2: Message trigger attack. Same as above with the exception that the frame counter counts the number of times the first message's command is detected. Once matched the counter, the frame starts transmitting, but from the second message.