

A MUX BUS UNIT

FIELD OF INVENTION

5

The invention broadly relates to a bus unit for connection to a mux bus and to a method of executing debugging, instrumentation, testing and verification functions, on a bus unit for connection to a mux bus.

10 BACKGROUND

15

A mux bus is a network where a plurality of units on the bus are interconnected and are able to communicate with each other. The units may then collectively be able to perform cohesive controlling, actuation and monitoring functions.

20

An example of a mux bus standard used in avionics is MIL-STD-1553, which is a computer and control system interconnect standard for units like sensors and computing elements. Bus units connected to a MIL-STD-1553 mux bus typically function as either bus controllers (BC), remote terminals (RT) or monitors. These bus units can take on one or more functions across one or more buses at any time.

25

A Line Replaceable Unit (LRU) connected to a mux bus can comprise one or more bus units; one or more processing elements (e.g. General Purpose Processor); and one or more Input/Output (I/O) Elements (e.g. Video boards, Audio boards, Ethernet ports, Serial ports).

30

Figure 1 illustrates an example of a prior art mux bus setup 100 that comprises various interconnected bus units 102 / 104a – d / 110 and an external bus unit 108 on the mux bus. Bus unit 102 has been configured as a BC. The BC initiates and controls communications traffic between all the other bus units by means of commands addressed to them. Additionally, BCs can send/receive data to/from RTs. There may be more than one bus unit that is configured as a BC in a single mux bus, however, only one BC can be active at any time. The additional BC usually acts as a backup BC. The backup BCs can either be configured as a RT or monitor, or disabled while the main BCs are in control. In other words, only one bus can initiate and control communications traffic between all the other bus units at any one time.

35

The other bus units 104a – d are configured as RTs, each having a specific address. RTs may either be actual sensor, power or motor control remote units or simulate traffic from said actual sensor, power or motor control remote units. Additionally, RTs can provide responses to commands from the BC and send/receive data to/from a BC. There are typically one or more RTs per mux bus. Bus unit 110 is configured as a monitor and is able to passively monitor mux bus traffic on the bus, e.g. during maintenance or trouble-shooting. Each mux bus can either have no monitor or any number of monitors. These monitoring devices may not transmit messages over the data bus. Their typical role is to monitor and record bus traffic, without interfering with the operation of the BCs or RTs. External bus unit 108 is used for maintenance services like debugging, instrumentation, testing and verification of the mux bus 100. It can interchangeably function as a BC, RT or monitor depending on the maintenance service to be rendered. The bus units in the mux bus setup 100 are interconnected via mux bus couplers 106a – 106d.

15

A typical method of debugging, instrumentation, testing and verification of a mux bus involves the connection of external bus units e.g. 108 to the mux bus. These external bus units typically reside in an external host computer such as a notebook or workstation.

20

In the BC role, the external bus unit replaces an existing BC on the bus. Low level testing and verification can be performed by sending / receiving data to / from the various interconnected bus units via a Graphical User Interface (GUI) on the external host computer. Receiving systems can be observed to see whether an expected response is produced; such as a change in the user interface display of the system, activation of an actuator, or electronic function. Transmitting systems can be observed for correct data sent to the BC, in response to the activation of a system function.

30

In the RT role, the external bus unit replaces an existing RT on the bus. Data can be exchanged between the external bus unit and the BC or other RTs. Receiving and transmitting systems can be tested in the same manner as in the BC role.

35

In the monitor role, the external bus unit listens to all traffic on the bus and displays it on the GUI of the external host computer. Testing and verification can be performed by activating a particular system functionality and observing the expected data on the GUI.

However, the limitations of this approach of debugging, instrumentation, testing and verification include having a need for an additional external bus unit. These bus units are typically expensive proprietary devices running dedicated
5 software and consume mux bus coupler resource as they require a direct connection to a mux bus coupler. Furthermore, a user is unable to test, verify, debug and monitor interactions not transacted on the actual mux bus.

A need therefore exists to provide a bus unit for connection to a mux bus and
10 a method of executing debugging, instrumentation, testing and verification functions on the bus unit for connection to the mux bus that seeks to address at least one of the abovementioned problems.

SUMMARY

15 According to a first aspect of the present invention, there is provided a bus unit for connection to a mux bus, comprising a mux bus transceiver, a bus driver with an integrated shell and a connection port for connecting an external computer to the bus unit, wherein the integrated shell is capable of responding to commands entered
20 via the external computer connected directly to the bus unit via the connection port for executing debugging, instrumentation, testing and verification functions.

The connection port may comprise a network (Ethernet) connection port or a
25 serial port.

The integrated shell may comprise a command parser for interpreting the commands entered via the external computer and is configured for being invoked by the external computer using a Telnet application.

30 In alternate embodiments, the integrated shell may be configured to listen on the connection port and to accept an incoming connection request from the external computer.

The debugging, instrumentation, testing and verification functions may
35 comprise one or more of a group consisting of performing memory dumps, reading/writing of message data, reading/writing of internal state of the bus driver, and reading/writing of bus unit hardware registers and memory.

In other embodiments, the debugging, instrumentation, testing and verification functions may comprise one or more of a group consisting of modifying the behaviour of the bus driver with the integrated shell, independent of an application layer; modifying data transactions to or from the bus driver with the integrated shell; injecting data into the application layer of other systems; identifying sources of incorrect behaviour or data on the mux bus; the ability to test, verify, instrument and debug the internal state of one or more bus drivers having the integrated shell; and the ability to test, verify, instrument and debug other systems on a mux bus having the bus driver with the integrated shell.

The mux bus may comprise a MIL-STD-1553 standard.

According to a second aspect of the present invention, there is provided a method of executing debugging, instrumentation, testing and verification functions on a bus unit for connection to a mux bus, the method comprising the steps of providing a bus driver with an integrated shell on the bus unit, connecting an external computer to the bus unit via a connection port of the bus unit and entering commands via the external computer, wherein the integrated shell is capable of responding to the commands for executing the debugging, instrumentation, testing and verification functions.

BRIEF DESCRIPTION OF THE DRAWINGS

Example embodiments of the invention will be better understood and readily apparent to one of ordinary skill in the art from the following written description, by way of example only, and in conjunction with the drawings, in which:

Figure 1 shows a schematic drawing of an example of a prior art mux bus debugging, instrumentation, testing and verification setup.

Figure 2 shows a schematic drawing of a mux bus setup embodying the present invention.

Figure 3 shows a flow chart illustrating a method of executing debugging, instrumentation, testing and verification functions on a bus unit for connection to a mux bus embodying the present invention.

Figure 4 is a schematic diagram illustrating a system architecture embodying the present invention.

Figure 5 shows an example screen capture showing frame and message instrumentation from a bus unit configured as a BC, embodying the present invention.

Figure 6 shows an example shell prompt embodying the present invention.

Figure 7 shows an example screen capture when reading message data from a bus unit configured as a BC, embodying the present invention.

Figure 8 shows an example screen capture when performing a memory dump operation from a bus unit configured as a BC, embodying the present invention.

Figure 9 illustrates the instruction flow in a bus unit having a bus driver with an integrated shell, which is operating in BC mode, embodying the present invention.

DETAILED DESCRIPTION

Figure 2 illustrates a mux bus setup 200 embodying the present invention, comprising a computer 214 connected directly, i.e. not via a mux bus coupler, to one of the bus units 202. A number of interconnected bus units 202 and 204a – f are present on the mux bus, in this example a MIL-STD-1553 mux bus. In this embodiment, bus unit 202 has been configured as a bus controller (BC). The BC 202 may initiate and control communications traffic between all the other bus units by means of commands addressed to them. The other bus units 204a – f are configured as RTs, each having a specific address. RTs may either be actual sensor, power or motor control remote units or simulate traffic from said actual sensor, power or motor control remote units. RTs can provide responses to commands from the BC and send/receive data to/from a BC. The bus units in the mux bus setup 200 are interconnected via mux bus couplers 206a – 206d.

The computer 214 is connected via a connection 212 to the BC 202. The connection is preferably network (Ethernet) or serial. In this embodiment, instead of incorporating an external bus unit 108 that is connected directly to the mux bus (compare Figure 1) to facilitate debugging, instrumentation, testing and verification of the mux bus 200, the same is performed using the computer 214 that is connected to the BC.

It will be appreciated by a person skilled in the art that a BC can only control one bus at a time. In other words, only one BC can initiate and control

communications traffic between all the other bus units on the mux bus at any one time. Therefore, a BC can only be connected to one mux bus at any one time. Further, within each mux bus, a bus unit may be connected to both Channel A and Channel B. If a Line Replaceable Unit (LRU) needs to control multiple mux buses, it will need to have multiple bus units, which are configured to operate in BC mode.

Figure 3 shows a flow chart 300 illustrating a method of executing debugging, instrumentation, testing and verification functions on a bus unit for connection to a mux bus embodying the present invention. At step 302, a bus driver with an integrated shell is provided on the bus unit. The integrated shell may be integrated into the bus driver of the bus unit's central processing unit. At step 304, an external computer is connected to the bus unit via a connection port of the bus unit. At step 306, commands are entered via the external computer, wherein the bus driver with an integrated shell is capable of responding to the commands for executing debugging, instrumentation, testing and verification functions.

Figure 4 illustrates one embodiment of the system architecture, generally configured as reference numeral 400, of the present invention. This system architecture 400 can be implemented in a Line Replaceable Unit (LRU) that is desired for connection to a mux bus. A bus driver with integrated shell 404 allows application software 402 and an operating system 406 to interact with the hardware 408 of the bus unit. The bus driver may be able to execute functions such as the initiation and shutting down of the bus unit; enabling and disabling messaging; modifying messages; resuming a frame and/or execution command; and the reading and writing of messages. The integrated shell, which is a piece of software that provides an interface for a user, is embedded within the bus driver and may be able to execute functions such as enabling the automatic resumption of the next frame; enabling and disabling of frame instrumentation; enabling and disabling message instrumentation; displaying frames or messages; halting any operations; and exiting the shell.

It will be appreciated by a person skilled in the art that under the control of the application software 402 and in conjunction with the bus driver with integrated shell 404 and the operating system 406, the bus unit may be configured to function as one or more of a BC, a RT, and a monitor on one or more mux buses. In Figure 4, functional units 414 to 416 represent such different configurations, connected to one or more mux buses via one or more mux bus transceivers 418.

One or more connection ports, for example, a serial interface 410 and Ethernet network interface 412 are also provided as hardware components of the system architecture 400. The serial interface 410 and Ethernet network interface 412 allow an external computer (not shown) with a suitable serial or Ethernet port to be connected to the bus unit so that the user may execute debugging, instrumentation, testing and verification functions.

The external computer may be connected to the integrated shell by having the integrated shell listen on a specific network port and accept incoming connection requests. Alternatively, the operating system 406 may provide a Telnet service from which the integrated shell may be invoked remotely from the connected external computer using a Telnet application.

Once connection to the integrated shell has been established by the external computer, the user may invoke various functions by entering commands into a command parser of the integrated shell. The command parser of the integrated shell allows the commands to be interpreted so that the desired functions may be executed. Using the command parser of the integrated shell, engineers can interactively enter commands into the shell to perform various operations across all modes of operations, e.g. BC Mode, Monitor Mode, and RT Mode.

Examples of commands that the user may enter and their corresponding descriptions are listed in Table 1 below. These commands include the initiation of operations, executing instrumentation functions and modifying the behaviour of the bus unit.

Table 1

Commands in any mode	Description
<DevId>	Select specified device
<DevId> B	Switch specified device to BC (Bus Controller) Mode
<DevId> M	Switch specified device to MT (Monitor) Mode
<DevId> R <RTAddr>	Switch specified device to RT (Remote Terminal) Mode at specified RT Address
<DevId> z	Shutdown specified device

BC Mode Commands	Description
h	Halt Operation
e <MsgId> <State>	Enable specified message id if State=1 Disable specified message id if State=0
s <DisplayMask>	Bit 17-16: Display Transacted Messages order by Bus then Command Word Bit 15: Display Device Memory Information Bit 14: Sort display by handle if 0 Sort display by memory location if 1

	Bit 13-8: Display specified frame if >0, else Display all frames Bit 7: Display Messages within Frames Bit 6: Display Instructions within Frames Bit 5: Display Instructions (Asynchronous) Bit 4: Display Instructions Bit 3: Display Message Bit 2: Display Message Bit 1: Display Data Block Bit 0: Display Memory Block
j	Toggle automatic resume of next frame
r <MsgId>	Read and Display specified message, if id >=0, Read and Display all messages, if id=-1
w <MsgId> [<data00> <data31>]	Write and Display data for specified message id. Up to 32 words may be specified
m <MsgId> <Mask> <Retry> <Ch> <CW1> <CW2>	Modifies specified message. Mask Bit 7: Modify Message Retry Mask Bit 6: Modify Message Channel Mask Bit 5: Modify CW1 RT Mask Bit 4: Modify CW1 SA Mask Bit 3: Modify CW1 WC Mask Bit 2: Modify CW2 RT Mask Bit 1: Modify CW2 SA Mask Bit 0: Modify CW2 WC Retry: Retries enabled if 1 Retries disabled if 0 Ch: BUS_A if 0 BUS_B if 1 CW1: Mil-Std-1553 BC Command Word 1 CW2: Mil-Std-1553 BC Command Word 2
1	Enable Free Running Frame and Message Instrumentation
0	Disable Free Running Frame and Message Monitoring Instrumentation
x <FrameId>	Execute specified frame
x	Resume execution

5

Monitor Mode Commands	Description
r <CW1> <CW2>	Read and Display Message specified by a pair of Command Words
1	Enable free running message instrumentation
0	Disable free running message instrumentation
s	Show message summary of all messages that that were processed

RT Mode Commands	Description
r <T> <SA>	Read and Display Message specified by direction and sub address
w <T> <SA> [<data00> <data31>]	Write and Display Message specified by direction and sub address. Up to 32 words may be specified
1	Enable free running message instrumentation
0	Disable free running message instrumentation

The described example embodiment advantageously enables a user with an external computer, comprising a suitable serial or Ethernet port, to execute debugging, instrumentation, testing and verification functions via the integrated shell's user interface, instead of using an external bus unit connected to the mux bus.

5

By entering appropriate commands, some of which are listed in Table 1, into the command parser of the integrated shell, the user may be able to:

- (a) Verify the correctness of bus driver operations across all modes of operation by triggering specific bus driver functionality using an appropriate syntax
- 10 (b) Verify the correctness of an application via the "r" command to view transmitted data from the application
- (c) Verify correctness of other systems via the "r" command to view received data from an application
- (d) Verify correctness of an application via the "w" command to modify received
- 15 data from other systems
- (e) Verify correctness of other systems via the "w" command to modify transmitted data from an application
- (f) Perform Register / Memory Dump via "s" command
- (g) Enable / Disable free running Frame / Message Instrumentation

20

Figure 5 shows an example screen capture 500 illustrating frame and message instrumentation from a bus unit configured as a BC. Screen capture 500 shows information such as Frame Boundaries; Frame Number; Message Interrupts; Interrupt Status Register Values and Interrupt Queue Indices that is not transacted

25 on the mux bus and can not be picked up by existing external bus units for testing, verifying, instrumenting and debugging.

The integrated shell advantageously converts raw binary data from the mux bus traffic into a raw hexadecimal format for the user, which is displayed on a

30 computer screen through the integrated shell's user interface. The shell's user interface may comprise a shell prompt 600 as seen in Figure 6. The integrated shell's prompt 600 may contain information about various registers and driver state information. The shell prompt 600 shows information regarding the Device Id 602, Mode of Operation 604, Time since start of system & Time-tag Register 606, BC

35 Frame Active Flag & BC Running Flag 608, Frame Number 610, Instruction Pointer 612, Frame Time Elapsed 614, Hardware Exception Counter 616, IRQ Queue Length & Queue Metrics 618, Number of interrupts serviced 620, Number of Frame

Interrupts serviced 622 and Number of Message interrupts serviced 624. The information above is displayed in a cooked engineering format. Further, the information is not transacted on the mux bus and can not be picked up by existing external bus units for testing, verifying, instrumenting and debugging.

5

Figure 7 shows an example screen capture 700 when reading message data from a bus unit configured as a BC. In addition to data transacted on the bus, internal bus driver information such as Time tag of Message; Internal Control Words and Status Words; Transaction Counters, Failure Counters, Retry Counters and Interrupt

10 Counters that is not transacted on the mux bus is shown in screen capture 700, and can not be picked up by existing external bus units for testing, verifying, instrumenting and debugging.

Other example operations include reading and writing of an internal state of

15 the bus driver, including instrumentation and debug information; reading and writing of bus unit hardware registers and memory; and performing memory dumps. Figure 8 shows an example screen capture 800 for a memory dump operation from a bus unit configured as a BC. The memory dump operation shown on screen capture 800 displays message definitions and instructions in the BC's memory.

20

Figure 9 illustrates the instruction flow 900 in a bus unit having a bus driver with an integrated shell, which is operating in BC mode, embodying the present invention. For a bus unit configured as a BC, the BC may be controlled to perform operations such as retrieving and/or transacting messages and updating transaction

25 statuses 902. In such an example, when the BC identifies a transaction with an interrupt request (IRQ) flag, an interrupt service routine (ISR) is triggered 906. The ISR is a callback subroutine in the bus driver whose execution is triggered by the reception of the IRQ. The ISR may then determine the type of IRQ and process the interrupt. If the user enables frame and/or message instrumentation through the

30 shell's user interface, frame and/or message instrumentation will be generated and displayed. Within the integrated shell, when a Telnet session is initiated by the user, the integrated shell preferably accepts the Telnet session and may then read and parse commands such as enabling and disabling messages to be modified; enabling the next frame to be automatically resumed; enabling and disabling of frame

35 instrumentation; enabling and disabling message instrumentation; displaying frames or messages; halting any operations; and exiting the shell 904.

Embodiments of the present invention can advantageously allow the user to execute debugging, instrumentation, testing and verification functions via a serial or Ethernet network connection on a bus unit, so that external dedicated bus units are not required when performing field investigations. Mux bus coupler resources
5 otherwise required for connection of the external dedicated bus units are freed up. Furthermore, a user may be able to test, verify, instrument and debug the internal state of one or more bus drivers having the integrated shell, in any mode of operation. e.g. BC Mode, Monitor Mode, and RT Mode. In addition, the user may be able to test, verify, instrument and debug other systems on a mux bus having a bus
10 driver with an integrated shell. The user may also be able to modify the behaviour of the bus driver with the integrated shell, independent of an application layer; modify data transactions to / from the bus driver with the integrated shell, independent of the application layer; inject data into the application layer of other systems; and identify sources of incorrect behaviour or data on the mux bus.

15

It will be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the embodiments without departing from a spirit or scope of the invention as broadly described. The embodiments are, therefore, to be considered in all respects to be
20 illustrative and not restrictive.

CLAIMS

1. A bus unit for connection to a mux bus, comprising:
a mux bus transceiver;
5 a bus driver with an integrated shell; and
a connection port for connecting an external computer to the bus unit;
wherein the integrated shell is capable of responding to commands entered
via the external computer connected directly to the bus unit via the connection port
for executing debugging, instrumentation, testing and verification functions.

10

2. The bus unit according to claim 1, wherein the connection port comprises a
network connection port or a serial port.

15

3. The bus unit as claimed in claim 2, wherein the network port comprises an
Ethernet port.

20

4. The bus unit according to any one of the preceding claims, wherein the
integrated shell comprises a command parser for interpreting the commands entered
via the external computer.

5. The bus unit according to any one of the preceding claims, wherein the
integrated shell is configured for being invoked by the external computer using a
Telnet application.

25

6. The bus unit according to any one of claims 1 to 4, wherein the integrated
shell is configured to listen on the connection port and to accept an incoming
connection request from the external computer.

30

7. The bus unit according to any one of the preceding claims, wherein the
debugging, instrumentation, testing and verification functions comprise one or more
of a group consisting of performing memory dumps, reading/writing of message data,
reading/writing of internal state of the bus driver, and reading/writing of bus unit
hardware registers and memory.

35

8. The bus unit according to any one of the preceding claims, wherein the
debugging, instrumentation, testing and verification functions comprise one or more
of a group consisting of modifying the behaviour of the bus driver with the integrated

shell, independent of an application layer; modifying data transactions to or from the bus driver with the integrated shell; injecting data into the application layer of other systems; identifying sources of incorrect behaviour or data on the mux bus; the ability to test, verify, instrument and debug the internal state of one or more bus drivers having the integrated shell; and the ability to test, verify, instrument and debug other systems on a mux bus having the bus driver with the integrated shell.

9. The bus unit according to any one of the preceding claims, wherein the mux bus comprises a MIL-STD-1553 standard.

10

10. A method of executing debugging, instrumentation, testing and verification functions on a bus unit for connection to a mux bus, the method comprising the steps of:

providing a bus driver with an integrated shell on the bus unit;
connecting an external computer to the bus unit via a connection port of the bus unit; and
entering commands via the external computer;
wherein the integrated shell is capable of responding to the commands for executing the debugging, instrumentation, testing and verification functions.

20

11. The method according to claim 10, wherein the connection port comprises a network connection port or a serial port.

12. The method according to claim 11, wherein the network port comprises an Ethernet port.

25

13. The method according to any one of claims 10 to 12, further comprising using a command parser of the integrated shell for interpreting the commands entered via the external computer.

30

14. The method according to any one of claims 10 to 13, further comprising using a Telnet application to invoke the integrated shell by the external computer.

15. The method according to any one of claims 10 to 13, further comprising the integrated shell listening on the connection port and to accept an incoming connection request from the external computer.

35

16. The method according to any one of claims 10 to 15, wherein the debugging, instrumentation, testing and verification functions comprise one or more of a group consisting of performing memory dumps, reading/writing of message data, reading/writing of internal state of the bus driver, and reading/writing of bus unit hardware registers and memory.

17. The method according to any one of claims 11 to 16, wherein the debugging, instrumentation, testing and verification functions comprise one or more of a group consisting of modifying the behaviour of the bus driver with the integrated shell, independent of an application layer; modifying data transactions to or from the bus driver with the integrated shell; injecting data into the application layer of other systems; identifying sources of incorrect behaviour or data on the mux bus; the ability to test, verify, instrument and debug the internal state of one or more bus drivers having the integrated shell; and the ability to test, verify, instrument and debug other systems on a mux bus having the bus driver with the integrated shell.

18. The method according to any one of claims 10 to 17, wherein the mux bus comprises a MIL-STD-1553 standard.