Advances in Intelligent Systems Research, volume 136
4th International Conference on Sensors, Mechatronics and Automation (ICSMA 2016)

Design and Research on Universal 1553B Bus Controller Communication Module

YongShan Liu[1,a], Ping Wu[1,b] and Chengxiao Zou[1,c]

[1] School of Aerospace Engineering Beijing Institute of Technology, Beijing 100081 , China.

[a]liuysh@bit.edu.cn,[b]146639161@qq.com, [c]1690457712@qq.com

Keywords: 1553B, Bus Controller, Universal.

Abstract. The software compatibility problems of 1553B Bus Controller system caused by the application of different manufacturers, making it inconvenience to users and reducing efficiency of development of new products. To solve the problem, the paper designs and develops universal communication module for Bus Controller. The module provides unified functional interfaces, and solves different board’s functional implementation via virtual function technology. The design idea of bus controller communication module is explained in this paper. At last, the communication module is tested and verified, the result proved that the design scheme is feasible.

Introduction

In the 1970s, aiming for solving complex confusing data transfer among subsystems in integrated avionics systems, the United States raised and released 1553B bus ,which stands for “Aircraft Interior Time Division Command / response multiplex data bus”. It regards specification about electrical characteristics and data transfer protocol [1]. It was brought into China in 1987, and China launched GJB289A-97 (Digital Time Division Command / Response Multiplex Data Bus) [2] in 1997. With the features of real-time response, fault tolerance, centralized control and so on, 1553B Bus holds high reliability. Therefore, it has been developed into one of the most important data bus in avionics systems, and employed in many areas, such as ships, tanks, satellites and transportation.

Generally, 1553B bus system uses multiple manufacturers’ boards. Different manufacturers may supply greatly different software applications, the users have to take much time to understand and debug various boards. The use of various boards causes inconvenience to users and low efficiency of developing new application products. Based on typical terminal task requirements, a universal communication module for bus controller has been put forward against the question, supporting unified functional interfaces to adapt different boards’ software applications. Finally, the paper takes the prevalent 1553B boards of Excalibur and GE as an example to explain the design idea of the module, and verifies the feasibility.

The Design of Communication Module for Bus Controller

1553B bus system mainly consists of three kinds of terminals: bus controllers (BC), remote terminals (RT) and bus monitors (BM), as is shown in Fig. 1.

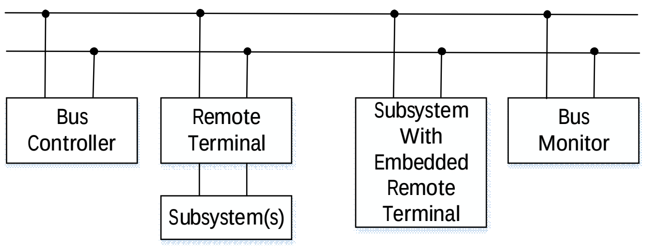


Fig. 1 Typical 1553B Bus System Architecture

Bus monitor does not participate in the bus data transmission, but just monitors and records total or selective bus data. Remote terminal responds the data transmission passively. The mounts of remote terminals can be up to 31 in a 1553B bus system. Bus controller sponsors and controls the bus data transmission. Based on message transmission, 1553B bus system exchanges the data. Bus controller arranges message sequence and controls the flow of bus data directly, so that bus controller is the core of the 1553B bus system. The paper focuses on universal communication module taking an example of bus controller.

According to the typical terminal application task requirements, following some principles of GJB1188A-97 and GJB289A-97 standard (Requirements of interface for aircraft/store electrical interconnection system) [3], which provide the function of the unified interface function, and combining with the characteristics of 1553b message structure, this article redefines the necessary data structures. There are a total of ten types of message formats, in which the two types of receive message (BC-RT) and transmit message (RT-BC) are applied in the communication module, in addition to transmit vector word of mode code with data.

Data structures. In the design, bus messages are assigned into a tertiary structure, nesting level-by-level, which is taken as core in the module, as is shown in Fig. 2.

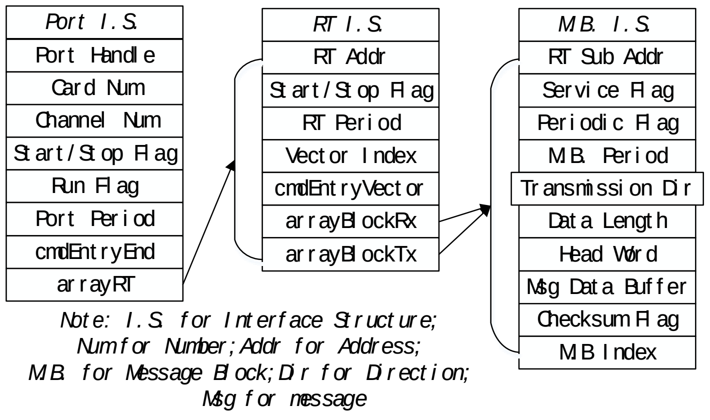


Fig. 2 Tertiary Structure of Bus Messages

The tertiary structure is port interface structure, RT interface structure and message block interface structure. They are marked with port handle, RT address and RT sub-address, respectively.

Message block interface structure is the bottom-level structure. As shown in Fig. 1, remote terminal communicates interactively with subsystems, where each subsystem usually monopolizes a RT sub-address. In general, it takes some sub-addresses for specific functions in a 1553B bus system. RT sub-address maintains practical engineering significance as identify of bottom-level structure.

Port interface structure is the top-level structure. While opening successfully a specified port on 1553B board, a unique value (called “port handle” in the paper) associated with the port would be return, and subsequent operations can be put through the port handle.

It nests in form of CArray pointer, level-by-level for tertiary structure. It should be noted that the RT Interface structure nests two CArray pointers of message block structure, for receive message(s) and transmit message(s), respectively.

Working status flags are defined to manipulate bus messages in all structures. The top-level structure defines a start/stop flag and a running flag; the middle-level structure (referred to “RT interface structure”) defines a start/stop flag and the bottom-level structure defines a service flag and a checksum flag.

In addition, period variables are also defined in all structure.

Tertiary structure will be created dynamically when needed.

In addition to above mentioned structures, two structures are also defined: user message block structure and status information structure. The first one is provided to user for setting single message block, including four categories of information: identifies of port handle, RT address and RT sub-address; data words of data length, head word, message data buffer and checksum flag; message format of transmission direction; periodic information of message block period and periodic flag. Each variable is mapped to message block structure, excluding port handle and RT address. The periodic information is only valid for receive message. The last one is used for transmit message block and defines three statuses: normal (0x00), head word fault (0x11) and checksum fault (0x12).

Design on Class Library of Communication Module of Bus Controller

Member Variables. Member variables are the core of design of class library. There are four member variables in the class library.

m_arrayPort: CArray variable of port interface structure with storing any messages related to 1553B bus data. It is initialized to NULL.

m_iPeriodSysInt: variable of interrupt cycle. It is initialized to 25ms, and can be reset by calling Set1553SysINTPeriod(). It employs periodic clock tick to process interrupt event in the module.

m_iTimerCount: variable of interrupt count, which records times of interrupt.

m_infoBlockTx: variable of status information structure. It updates continuously and records the latest processed status information of transmit message on data bus.

Functional Interface Functions. Based on typical terminal application task requirements, the module provides unified interface functions for users. The Interface functions can be summarized into five sub-modules in function, as is shown in Fig. 3. They are port module, information management module, period setting module, running control module and event handle module.

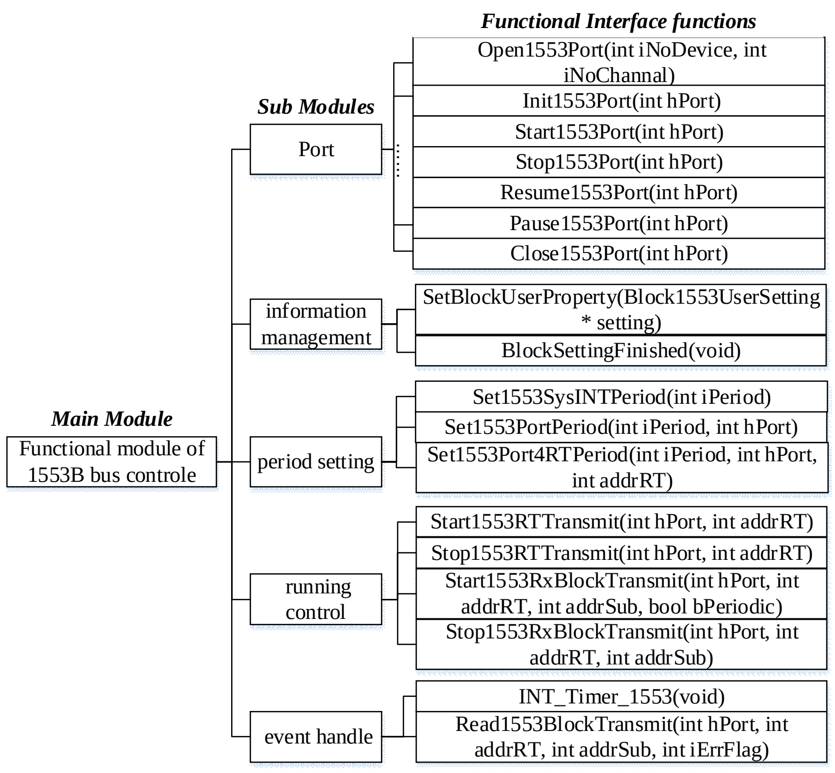

Functional Interface functions
Sub Modules
Main Module
Functional module of 1553B bus controle
Port
information management
period setting
running control
event handle
Open1553Port(int iNoDevice, int iNoChannal)
Init1553Port(int hPort)
Start1553Port(int hPort)
Stop1553Port(int hPort)
Resume1553Port(int hPort)
Pause1553Port(int hPort)
Close1553Port(int hPort)
SetBlockUserProperty(Block1553UserSetting * setting)
BlockSettingFinished(void)
Set1553SysINTPeriod(int iPeriod)
Set1553PortPeriod(int iPeriod, int hPort)
Set1553Port4RTPeriod(int iPeriod, int hPort, int addrRT)
Start1553RTTransmit(int hPort, int addrRT)
Stop1553RTTransmit(int hPort, int addrRT)
Start1553RxBlockTransmit(int hPort, int addrRT, int addrSub, bool bPeriodic)
Stop1553RxBlockTransmit(int hPort, int addrRT, int addrSub)
INT_Timer_1553(void)
Read1553BlockTransmit(int hPort, int addrRT, int addrSub, int iErrFlag)

Fig. 3 Functional Diagram of Interface Function Module

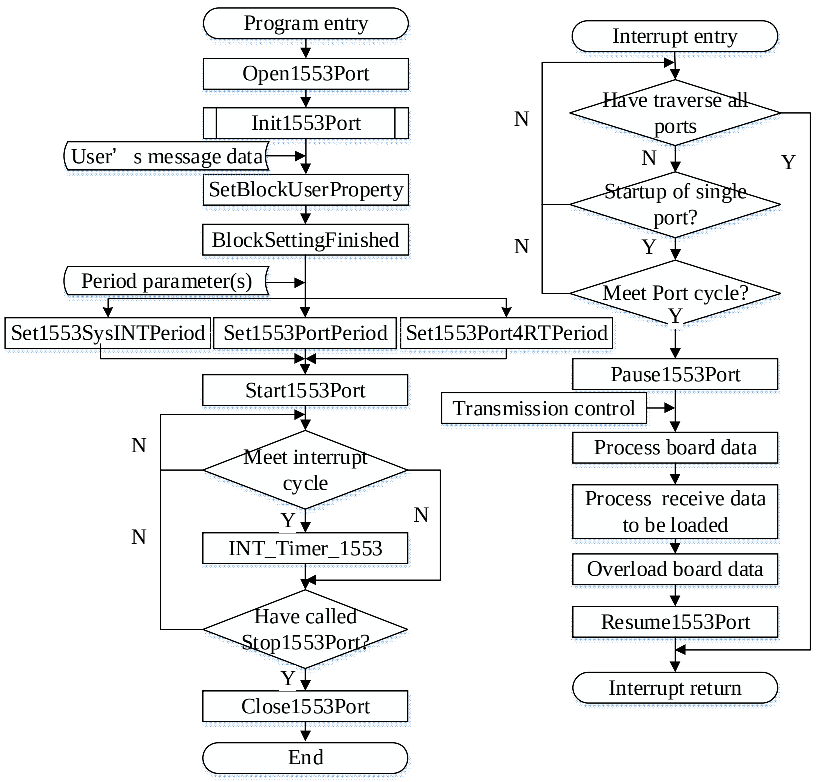


Fig. 4 Flow Chart of Operation Sequence for Bus Controller

Operation sequence for bus controller. Fig. 4 shows the flow chart of operation sequence for bus controller.

To simplify operation procedure, Init1553Port() should be called inside Open1553Port() after a specified port is opened successfully. One or more ports can be opened and initialized, but only one is active on bus at a time.

After inputting all message block data, BlockSettingFinished() is called to allocate these messages to on-board memory. Message command/control entries are sorted and arranged in group associated with RT address, adding an end control entry to the tail of list. In a group, entries are allocated in order of transmit message group/receive message group/vector word message. All messages are assigned continuously and set to branch without condition. After allocating all messages, Set1553SysIntPeriod() are called to set all period values for interrupt/Port/RT to default 25ms.

Periods setting are set after allocating all messages. Periods setting are used to set/reset specified port/RT period or reset the interrupt period. Users can ignore the step.

Start1553Port() is called to allow to transfer message data on bus through a specified port. The application should set a timer using the value of m_iPeriodSysInt, following with starting a port. INT_Timer_1553() is called in the timer function.

Periodic interrupt handle is the most important part of the design of communication module for bus controller. The module employs timed period interrupt to process data exchange between host computer and 1553B bus. Steps of interrupt processing are as follows:

1. Update times of interrupt, calculate the total outage time.
2. Traverse all ports. When a port is started and meets port period requirement, call Pause1553Port() to pause the port running and turn to step 3. If all ports don’t meet requirements, exit the interrupt handle immediately.
3. Process on-board message data and message data of receive message block(s) to be transferred on bus. Read on-board message and store transmit message data into Message Data Buffer associated with specific transmit message block(s). Then verify the data words and update status information to m_infoBlockTx, followed by triggering event of transmit message block status for further processing. Read vector word(s) to decide whether transmit message block(s) to be transferred or not. Bit 0 in vector word indicates RT sub-address 1，bit 15 indicates RT sub-address 16 and so on. If a bit is logic 1, individual transmit message block of corresponding sub-address is allowed to be transferred on bus. Preprocess those receive data word information to be transferred, which meet transmission condition. Head Word is inserted into the first location of Message Data Buffer. If checksum flag is set to TRUE, verify the data words according to GJB1188A-99 and insert the checksum word into the end location (the length of buffer is decided by Data Length).
4. Reallocate and reload bus message to on-board memory. At the time only load those messages, in which associated RT interface structures meet transmission condition. Based on Service Flag in message block, set the type of control instruction (Run/branch/branch without condition) for individual transmit (or receive) message to be loaded. In order to avoid repeating transmission, reset Service Flag to FALSE for one-time receive message after loading. After all messages are allocated completely, add the control instruction of End-of-List to the tail of the message command list.
5. Call Resume1553Port() to run the port again and exit the interrupt handle.

Running control module manipulates bus message transmission from all levels, and causes direct impact on the step 3 the step 4 of interrupt handle. When a RT is disabled to transfer, all messages inside the RT interface structure are prohibited to transfer on bus. Only the transmission of RT and sub-address are activated, the corresponding receive message block is allowed to transfer. Call

Start1553RxBlockTransmit() to decide single receive message to transfer periodically(TRUE) or once(FALSE) by the value of the fourth argument. User can set receive message block period through SetBlockUserProperty(). While a RT is allowed to transfer, the vector word message for the RT is transferred every RT period. The transmission of individual transmit message block depends on content of vector word. By default, all bus messages are initialized to transfer inhibition (all working status flags are set to FALSE).

Stop1553Port() is called to stop transferring message data through a started port. When stopping port, the application shutdowns the timer. If user does not restart the port again, call Close1553Port() to close the port and release board resource, quit the program immediately.

Universal Design Idea of Communication Module for Bus Controller. It must consider the differences of software applications from different manufactures' boards in the design. Class has the characteristics of overloading with virtual functions and inheriting public functions. It is suggested that the universal module (base class) provides virtual function interfaces with the derived class overriding the implements when it comes to on-board operation, and carries out the implements in form of public functions in itself when it comes to the requirements of functional operation without involving on-board operation.

The paper takes prevalent 1553B boards of Excalibur [4] and GE [5] to do some comparative analysis for explaining the design idea of communication.

```
Excalibur'  s board API function related to port initialization
int  Set_Mode_MCH(int handle,   /*The handle returned by Init_Module_MCH*/
               short mode /*Operation mode*/);

GE'  s board API function related to port initialization
BT_INT BusTools_BC_Init(BT_INT wCardnum , /*BusTools card number*/
               BT_UINT wBcOption, /*Procrssing options*/
               BT_U32BIT dwIntEnable, /*Interrupt enable bits*/
               BT_UINT wRetry, /*Retry enable bits*/
               BT_UINT wTimeout1, /*Timeout period for the "  No Response"  error*/
               BT_UINT wTimeOut2, /*time-out period for the "  Late Response"  error*/
               BT_U32INT dFrame, /*minor frame */
               BT_UINT num_buffers /*number of data buffers*/) ;
```

Fig. 5 Board API Function related to Port Initialization (Excalibur's vs. GE's)

For the function, different manufactures' boards may require different parameter lists. The design of virtual function must take into account the difference of software applications, offering the most needed parameter list. To initialize a specific port, for example, the related board API functions of the two manufactures are shown in Fig. 5. It can be seen that the first argument of both API functions is determined by a port handle, the other arguments can be determined internally based on actual requirements. So the function for port initialization (Init1553Port()) only provides a parameter of port handle to user.

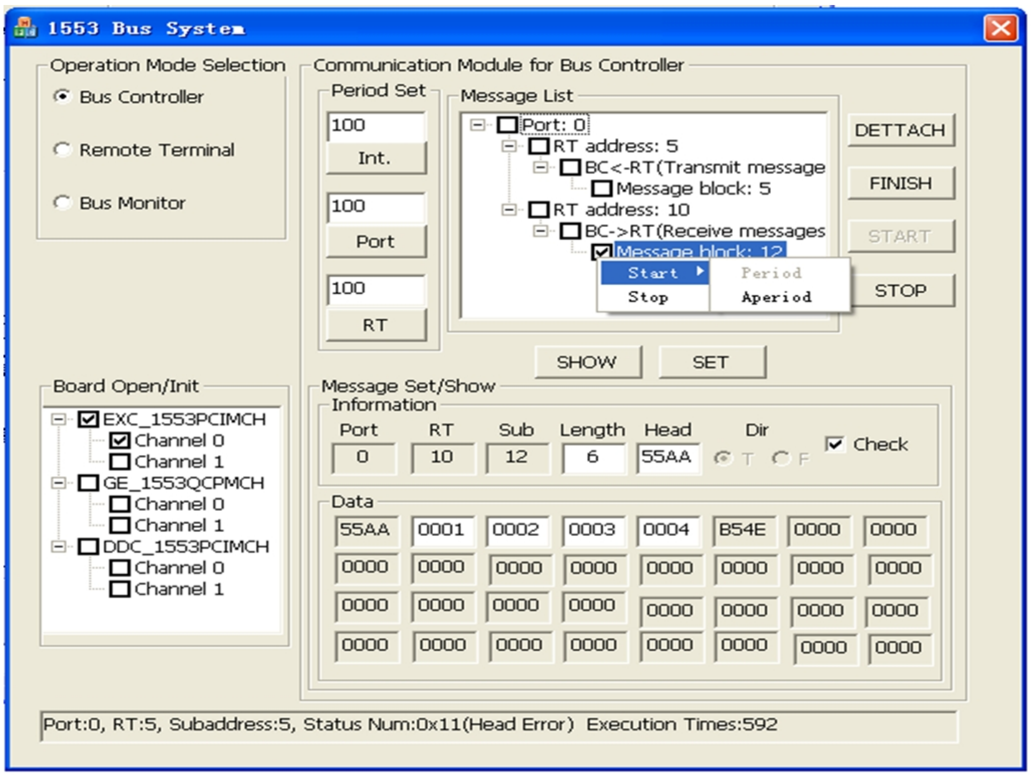
Other operations involving functional operation requirements without on-board operation, such as period setting, running control and single-message setting, are implemented in form of public functions of base class.

Polymorphic mechanism is applied to achieve the universalization for communication module. To ensure proper execution of the program, derived classes do nothing but override virtual functions' implement.

Test and Verification

The paper introduces the design idea and program implementation of communication module for bus controller. The module for remote terminal and bus monitor are implemented, too. Given that the design and implementation for the two latter is much simple, they are not introduced in this paper.

A set of 1553B bus communication test system (as shown in Fig. 6) are designed to verify the feasibility of the design of software module. There are three boards installed into host computer. Those boards are EXC-1553PCI/MCH(Excalibur's), QCP-1553(GE's) and BU-65569i(PCI bus, DDC). Each board has two channels. Only one of three terminal operation modes can be selected at one time. The right screen shows BC mode running.



Operation Mode Selection: [✓ Bus Controller] Remote Terminal Bus Monitor

Period Set: Int. 100; Port 100; RT 100

Message List:
- Port: 0
  - RT address: 5
    - BC<-RT(Transmit message
      - Message block: 5
  - RT address: 10
    - BC->RT(Receive messages
      - ✓ Message block: 12

Board Open/Init:
- ✓ EXC_1553PCIMCH
  - ✓ Channel 0
  - Channel 1
- GE_1553QCPMCH
  - Channel 0
  - Channel 1
- DDC_1553PCIMCH
  - Channel 0
  - Channel 1

Port	RT	Sub	Length	Head	Dir	Check
0	10	12	6	55AA	T F	✓

Data:

55AA	0001	0002	0003	0004	B54E	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000

Port:0, RT:5, Subaddress:5, Status Num:0x11(Head Error) Execution Times:592

Fig. 6 1553B Bus Communication Interface

The communication module for bus controller runs well, it achieves perfect effect in the test.

Conclusion

For software compatibility problem caused by use of different manufactures' board in complex 1553B bus system, the paper designs and develops universal communication for bus controller. Based on typical terminal task application requirements, the module redefines data structures and provides unified functional interfaces. The module uses virtual function technology with the derived class overriding the implements when it comes to on-board operation. By comparing boards of Excalibur's and GE's, the paper briefly explains the design idea. Finally, the paper verify the feasibility of communication module for bus controller.

References

[1]. *MIL-STD-1553 Tutorial*, Condor Engineering (2000), p.10-11.
[2]. *Digital time division command/response multiplex data bus* (1997).
[3]. *Requirements of interface for aircraft/store electrical interconnection system* (1999).
[4]. *Excalibur PCI/MCH Family Software Tools Programmer's Reference* (2001), p.19, p.40-43.

[5]. *Bus Tools/1553-API Software Reference Manual*, GE Fanuc Intelligent Platforms (2009), p.121-123.