Real Time Control of Hardware In-the-Loop Simulation Based on CAN Bus and RTX

Sha De, Zhang Jinsheng, Liu Shuo, Hua Chao
Xi’an Research Institute of High Technology, Xi’an, China
Email: nwpu_2012sha@sina.com

Abstract. In Hardware In-the-Loop Simulation (HILS) of missile, real time is the main consideration. Due to the poor real time performance of the Windows system, a real time solution scheme for HILS of missile based on the RTX system is proposed, and the design of the RTX hardware driver and the reflective memory technology are introduced. Aiming at the problem of precise timing and data synchronization in real time system, the timer of RTX is studied and a hardware solution for data synchronization is proposed. For the CAN bus to long distance transmission in high baud rate conditions, the CAN relay station is designed. The HILS platform has been built, and the result of simulation is correct without missing frames and wrong frames, meeting the requirements of stability and reliability.

1. Introduction
Compared with digital simulation, HILS has higher reliability and authenticity. At the same time, it can replace the difficult mathematical modeling components in digital simulation and avoid the error caused by the inaccurate mathematical model. It is a simulation method with high confidence level in simulation technology. Compared with the flight test, the price of HILS is low and can test repeatedly, the development cycle is short and the boundary conditions of weapons can be verified. Because of the physical equipment participation, the HILS is also a real time simulation, which requires that the time scale of the simulation model is exactly the same as the time scale of the real system.[1]

At present, RS-422/485 bus and ARINC429 bus are mostly used as tactical missile communication systems. In these tactical missile communication systems, the point to point double line connection is needed. But the line connection is complex and the risk of the system failure is increased. CAN bus is a field bus. Any node on CAN bus can actively send information to other nodes on the network at any time without dividing its primary and secondary information. It has high reliability and cost-effective. The advantages of this bus are as follows: [2][3][4]

1) High reliability: The working mode of CAN bus can be master-slave or multi-master. Any node in the network can send data to the bus at any time, which can form a system with multi-master structure or redundant structure, thus improving the reliability of the system.

2) Short development cycle: CAN bus has a perfect communication protocol with funds, and the communication protocol has CAN bus controller chip and its own interface chip, thus greatly reducing the difficulty of the development of the system and shortening the development cycle.

3) High security: CAN bus’s sending and receiving data needs to be checked each other. Each frame has a response bit, a hardened CRC check and an insert bit check. Besides, it has short frame message structure, short transmission time and low interference probability.
2. Reflective memory network
A reflective memory network is a special type of shared memory network that enables multiple independent computers to share a general set of data, and it can save an independent backup of the entire shared memory in each subsystem. Reflective memory network card is inserted on each node that needs real time communication. Each card has its own independent local memory. It maps to the host memory from local memory, and users read and write the data on the network card as quickly as the memory of the host memory. In addition, each reflective memory network card maps the local memory on the distributed node card to a virtual global memory by mapping the local memory on the distributed node card, that is, each node writes the local node card data and writes all the memory on all other node cards at the same time. So that the user's reading and writing to the local node memory is equivalent to the global. Memory is read and written, and this global memory is visible and shared by all nodes, so as to achieve data communication between distributed nodes.

3. Solutions of based on RTX
Because of the defects of Windows system in real time, RTX system is widely used in engineering as a real time operation system in HILS. RTX (Real Time Extension) is a real time solution based on Windows operating system developed by American Ardence company. It is the only hard real time extension subsystem of only pure software under Windows platform. RTX does not make any encapsulation or modification to Windows system, but enables the management and scheduling of priority based preemptive real time tasks by adding real time HAL extensions on the HAL layer. It allows the threads of the RTX real time subsystem RTSS to take precedence over all the Windows threads. Real time HAL extension provides RTX's own interrupt management mechanism, and it can directly access I/O hardware ports and memory. These mechanisms ensure that RTX threads occupy priority all the time, and the device driver function can be realized through port control. The RTX architecture is shown in Figure 1.

3.1. Real time applications and non-real time applications
In the software of system, it can be divided into two main tasks: real time task and non-real time task. The real time tasks include: hardware drive, data acquisition, transmission, time series control and so on. The real time task program is run under RTSS. Its program design is different from other Windows programs; non-real time tasks include man-machine interface, parameter input, data display and processing and so on. It runs under the Win32 subsystem of Windows, using MFC to design. Communication between real time tasks and non-real time tasks is achieved by shared memory. The real time task sends the collected information to non-real time tasks through shared memory, and the information is displayed by non-real time tasks which further process the information.

3.2. Design of RTX hardware drivers
Since most hardware boards do not support RTX system, it is necessary to design hardware drivers programs. The RTX drivers of hardware will adopt a strict modular design idea and the running mode of RTSSDLL on the package. In order to expand its application scope, a set of indirect call interfaces for Win32 environment is added based on IPC mechanism. Its composition structure is as shown in Figure 2.
below.

![RTX hardware driver structure](image)

The RTX drivers of the hardware board are driven by the *.rtss file as the main process file, and are driven to reside in the system memory in the form of process instances and export to all real time applications under the RTSS environment for invoking. The biggest feature of this form is that its driver runs in memory with instance data, and can conveniently use the monitoring tools provided by RTX to observe the running state of the driver in real time and facilitate the development of the user's follow-up simulation program.

For most users, the Win32 environment is still the most familiar running platform and development environment. In order to facilitate users to debug and test hardware in the Win32 platform environment, the driver provides an additional set of API interface for the Win32 environment. The interface actually uses the event cycle pump technology through the IPC mechanism to indirectly realize the hardware control by transferring the hardware configuration parameter data structure with shared memory. In view of the complexity of the IPC mechanism and the cumbersome design of the hardware registers, the software encapsulates the process as a dynamic link library under the Win32, making it possible for the user to call the interface transparently without caring about the running of the IPC.

3.3. Design of RTX software framework
RTX has a breakthrough relative to Win32 in real time. The essence is that it provides a stable, accurate interrupt response delay, fast thread switching and high precision timing mechanism. Therefore, the emphasis of RTX real time application are the hardware driver support in RTSS environment, the creation of multi priority threads and the use of high-precision timers.

The RTX software framework will be built on this basis. The structure of the RTX software framework is shown in Figure 3.
Figure 3. RTX software framework

The main thread of real-time application mainly includes kernel variable creation, hardware driver initialization, opening hardware device management, timer creation and working thread creation. In the working thread, the local variables are defined at the entrance, and the large capacity memory buffer used for data acquisition is dynamically allocated. Then the periodic triggering events are waited for, thus the periodic data acquisition is realized. The periodic triggering of the thread is triggered by timer timing, so as to realize the time control of high precision timer for data acquisition of worker threads.

3.4. Timer and data synchronization

A real-time system requires a large number of operating system timer tasks. The operating system must be timed accurately, and the thread is scheduled at accurate time points, and the thread is hung at the exact time interval. At the same time, the functions of RTX clock and timer, hardware counters (0.1 s timing precision) provide the necessary services for the thread to complete cycle tasks. This paper focuses on the clocks under RTX.

The clock under RTX is a counter that measures the specified time interval of the past time. It is based on 100ns. There are three types of RTX clocks, Clock_1, Clock_2 and Clock_3. This system uses the clock Clock_2. Clock_2 is provided by the real-time hardware abstraction layer extension. Its precision is 0.1μs. The timer scheduling cycle under this clock can be 100μs, 200μs, 500μs, 1000μs. The software framework uses the RtCreateTimer function to create a Clock_2 based timer that specifies the processing program to be executed when the timer is aborted, that is, each timing subtask sets the priority of each timing subtask corresponding to the thread at the same time. The time and repetition interval of each timer are set with the RtSetTimerRelative function. When the repetition interval is non-zero, the timer will be repeated at a definite interval after the first abort. If the repetition interval is 0, the timer will be stopped once.

The timer of the RTX is an implicit thread. When the timer is stopped, the RTSS sends out notifications, and the thread will call the handler specified when the timer is created. The timer is related
to the clock of the system. When the timer is created, it will measure its stopping time. Once the timer is aborted, it will automatically restart and the processing thread will run. The operation of the processing thread will not affect the restart of the timer.

For data synchronization, the system uses hardware clock source to synchronize to increase high reliability. The pulse generator is used to generate the pulse signal of 1ms, and the PCI-1751 card is inserted into each node computer. The PCI-1751 PC00 port and the pulse generator signal output port are connected through the wire, and the PC00 port of each card is connected in series. Time1 as the external clock source, according to the card user manual \(^5\), users can set PC00 as the terminal output port. The IRQ level is set automatically. It can greatly reduce the data transmission delay caused by the heavy load of computers and has higher stability and time accuracy.

4. CAN relay technology

The communication distance of CAN bus is inversely proportional to the baud rate. If we want to carry out the CAN bus data transmission with long distance and high baud rate, it is necessary to use the CAN bus relay technology to transfer the data. In this system, a CAN relay node is added between two CAN nodes. The CAN bus board and the reflective memory card are inserted in them. The node A transfers the information on the CAN bus through the CAN bus to the relay station B, and the data from the relay station B is written to the reflection memory network, then the data is shared and the data is passed through CAN bus to the CAN node C. The data shared by reflective memory in relay station not only backup CAN bus information, but also monitor the accuracy of node B receiving relay information. The schematic diagram of the CAN relay technology is shown in Figure 4.

![Figure 4. Schematic diagram of the CAN relay](image)

5. HILS results

The HILS platform is built according to the software design. The test objects include missile-borne computer, damping gyroscope and steering gear. The test equipment includes ADI real time simulation computer and three axis turntable. The simulation period is set to 1ms. The experimental results(Figure 5,Figure 6,Figure 7)show that the result of the experiment is free of frame and frame error, and the simulation results are correct.

![Figure 5. Data of damping gyroscope](image)
6. Conclusion
This paper analyzes the software of the HILS of a missile control system, introduces the solution based on the RTX system, and analyzes the reflective memory technology and the CAN relay technology, and solves the real time and timing problems in HILS. Meanwhile, an external hard clock method is introduced to solve the data synchronization problem in HILS. The simulation results show that there is no missing frame and wrong frame in the simulation process, and the simulation results are correct. This article provides technical reference for HIL Simulation engineers.

References
[1] Fang Huiyu 1995 System simulation of antiaircraft missile weapon. (Beijing: Astronautic Publishing House) p 133
[2] Zhao Shiming, Sun Zhiyue. Error Analysis of Infrared Guidance Hardware In Loop Simulation System[J]. Infrared and Laser Engineering 2017,46(08):49-54.
[3] Yue Chao, Dong Guocai, Yu Ge. Design of Missile Hardware-in-loop Simulation System Based on RTX [J]. Journal of Projectiles, Rockets, Missiles and Guidance 2016,36(01):19-22.
[4] Zhang Jianman. Application of Real-Time System Based on RTX For HL Simulation [J]. Computer Measurement & Control 2016,24(03):136-138.
[5] Advantech Company 2004 PCI-1751U User Manual