INS Checkout System: Realization as a Real-Time System

Jaya G. Nair, C. Radhakrishna Pillai, S. Hemachandran, Dr. P. P. Mohan Lal
ISRO Inertial Systems Unit, Indian Space Research Organisation,
Vattiyoorkavu, Thiruvananthapuram
g_jaya@vssc.gov.in

Abstract:

Requirements of checkout systems have many areas where timeliness is of utmost importance. Automation, safety and security requirements also create the need for features where deadlines have to be met. The design of checkout system has to be undertaken as a real-time system to effectively meet these needs. The features in the design of RESINS Checkout stsrem as a real-time system are detailed here.

Key words: Checkout Systems, Automation, Safety, Fault-tolerance.

I. INTRODUCTION

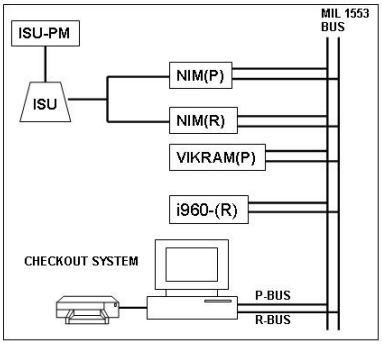
Automated Checkout Systems are used for the test and evaluation of navigation systems used in launch vehicles. AMC-based RESINS Checkout System caters to the testing requirements of the Redundant Strap down Inertial Navigation System (RESINS) which is the navigation system used for PSLV and GSLV launch vehicles. The navigation system uses the Advanced Mission Computer (AMC) for doing the navigation computations. The checkout system provides all necessary interfaces to connect RESINS and AMC and implements the different test procedures with minimum need for user intervention. The tests include calibration and performance evaluation tests in different environmental conditions to qualify the systems for flight. The various functions of the checkout system differ in terms of the requirements of periodicity, criticality and deadlines. To satisfy the multiple requirements, the checkout system needs to be implemented as a real –time system. This paper discusses the design of features and the implementation methodology that leads to the realization of such a real-time system.

The different types of requirements to the checkout software are briefly enumerated in section II. This is followed by an analysis of the requirements with the point of view of a real-time system in section III. Design decisions are enumerated in section IV followed by design of hardware and software in sections V and VI respectively. Section VII covers the evaluation of the checkout system, followed by implementation details and conclusions in sections VIII and IX respectively.

II. REQUIREMENTS TO CHECKOUT SYSTEM

Fig.[1] shows the test articles that constitute AMC-RESINS and the checkout system. Mark-4 ISU is connected to Navigation Interface Modules (NIM). The two NIM and two Navigation and Guidance Processor (NGCP) packages, which form the onboard processor units, are connected to MIL-1553 buses with redundant connections. The checkout system is also connected to the two buses. Interfaces necessary for powering the packages, for monitoring, data acquisition and providing simulated inputs to the packages are also needed, though not shown in fig [1].

The requirements to the checkout system are derived from the test and evaluation plan of AMC-



Fig[1]: Test Articles and Checkout System

RESINS. These include powering packages, data acquisition, data storage and implementing procedures for calibration and performance evaluation tests. The tests include simulation tests where the checkout system has to simulate the inputs to NIM and tests where checkout system has to simulate the behavior of NGCP on the two buses for conducting tests without NGCP packages.

The checkout system is characterized by special challenges related to the environment of use, the complexity of multiple inter-related requirements and stringent demands of quality. These translate to additional implied requirements, which include the need for automation, reconfiguration and implementation of features for safely, security, fault tolerance and robustness.

On analysis, it turns out that the requirements can be categorized as functional, temporal and reliability requirements.

1. Functional Requirements

These include the following:

- Powering packages with control on sequence and checks to ensure safety.
- Automating test procedures to support fool-proof operations with minimum intervention from users.
- Tools for data processing and to generate results leading to decision making about correctness of tests.
- Display of status, logging of data and printouts of reports of tests.
- Archival of data.

2. Temporal Requirements

Requirements that have deadlines with respect to time are classified here and include the following:

- Mil-1553 bus protocol implementation must maintain the timing requirements of the protocol in terms of millisec. Simulation tests where NGCP messages must be simulate on the two buses must satisfy the inter-message timings as specified in the NGCP message format.
- Data acquisition and storage in flight mode require the checkout system to collect data from multiple RTs and broadcast messages from NGCP every minor cycle (20 msec). Along with storage to files, real-time frame validation checks must be conducted on the data and real-time display of critical parameters done.
- Simulation tests needs the checkout system to simulate 11 digital and analog inputs to NIMs every 20 msec, with updates being done in synchronization with minor-cycle of NIM.
- Continuous surveillance of health of the checkout system must be done by acquiring ADC data from upto 50 channels with automatic safety actions, once every second, and automatic checks on data acquired from NIM, every 2 sec.
- Periodic status display at the rate of once every sec and providing interactive response to user input also require timely performance in the checkout system.

3. Reliability Requirements

These include the following:

- Need to ensure safety of the onboard packages, security of stored files with respect to different missions and need for user validation.
- Fault tolerance where a fail-safe condition must be satisfied. User must be alerted in case of any error and diagnostic tools must be available to aid failure analysis.

III. REQUIREMENTS ANALYSIS

Analysis of the requirements point out that the checkout system has real, hard and soft real-time deadlines. Mil-1553 bus protocol has hard real-time deadlines, while periodic status display shows a soft real-time deadline.

To satisfy these deadlines, the checkout system must implement the following:

- Robust, fast hardware for specific function
- Interrupt based task management
- Multiple processes with preemptive priority based scheduling
- Inter-process communication
- Synchronization of processes
- Mutual exclusion of resources like files, data buffers and printer
- Graphical User Interface

IV. DESIGN DECISIONS

The following design decisions are taken before the detailed design is carried out.

- Use of industry standard cPCI- based hardware including robust PC, I/O cards and device drivers.
- Use of a real-time operating system and QNX is selected, with application software in C and C++.
- The requirements are aimed to be realized by a mix of hardware and software.

V. DESIGN OF CHECKOUT HARDWARE

A single cPCI system is chosen with PC and 9 I/O cards. I/O cards with built-in intelligence are used. Provision for spare channels is made in the I/Os so that failures can be supported and the checkout system reconfigured. cPCI is chosen since it ia state-of-the art, rugged, offers more slots and a variety of interface boards with high I/O capacity are available.

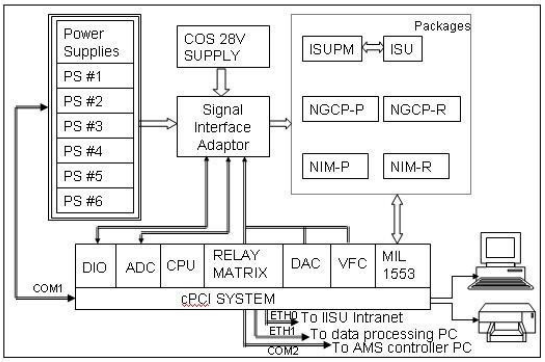
A modular Signal Interface Adaptor (SIA) is designed which implements the interfaces to the onboard packages. The functions include

- Powering packages with hardware interlocks and safety features
- Power supply and package-end voltage monitoring
- Current monitoring with hall sensors
- Bypass control for power and manual emergency OFF switch
- Watch dog and surveillance audio alarm
- Control signal interfaces
- LED display on front panel for ON/OFF status and error indication
- Signal terminations

Fig [2] shows the hardware configuration. It shows the SIA, cPCI system with PC and I/O cards and the many interfaces.

1. Hardware Design- Case study

To illustrate the hardware design to realize real-time requirements, design of the Mil-1553 protocol implementation is described as a case study.



Fig[2]: Hardware Configuration

The capabilities needed on the Mil-1553 bus include the protocol implementation, acquisition of telemetry data with no misses, simulation of NGCP messages on the bus with synchronization of the simulation in the prime and redundant buses.

The Mil-1553 interface selected is an intelligent board with the following capabilities:

- Two channels with BC, RT and MT modes of operation.
- In BC mode, the capability to generate a programmed sequence of messages on the bus, acquisition of RT data and multiple frame simulation capability.
- In MT mode, capability of data transfer at block-interrupt or message interrupts and selective bus monitoring with RT or sub-address as the condition.

The solution for the Mil-1553 capability has the following features:

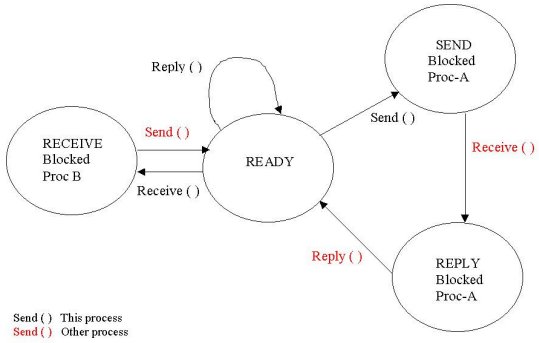
- Tight protocol requirements are shifted to the intelligent board.
- Capability to store upto 1024 messages on board with double buffering scheme ensures no data is lost.
- Block interrupt scheme of the driver software is used to implement simulation requirements.
- Synchronized behavior between prime and redundant channels is achieved.

VI. DESIGN OF CHECKOUT SOFTWARE

The checkout software is designed on top of QNX real-time operating system. QNX is selected due to the availability of features like multiple prioritized interrupts, multiple priority-levels for processes, preemptive priority based scheduling, low interrupt latency, fast context switching times, different mechanisms for inter-process communication, availability of synchronization primitives, rugged file-system etc. The real-time features of QNX enable the development of the checkout software to meet the real-time deadlines.

1. QNX Architecture

QNX is a micro-kernel architecture built with a small kernel which comprises of IPC, scheduler, H/W interrupts and a small part of the network-process. Four other scalable processes viz. process manager, device manger, file system manager and network manager make it a full-fledged OS. The native messaging scheme is the fundamental inter-process communication which is synchronous in nature. Hence sender and receiver must be in phase with execution otherwise one of the processes has to wait. This concept posed a challenge when partitioning the processes during the design of checkout software. The concept of sender get ting blocked if its corresponding receiver is not waiting or the receiver getting blocked when nobody is requesting for data are helpful in portioning the processes. The state diagram of QNX is as given below in Fig[3].



Fig[3]: QNX state diagram

2. Software Architecture

The software is designed to have client-server architecture. The servers are independent, each implementing a particular functionality and expose themselves only through the interface. The main checkout program acts a client to the many servers. The server programs include device drivers for I/O cards, programs for handling processing and storage of high-speed data, implementing the protocol for communicating with the test articles, for printing functions etc. Servers run in the background and the users interact with the clients. Fig[4] shows the list of servers and clients that typically operate during a session of checkout use.

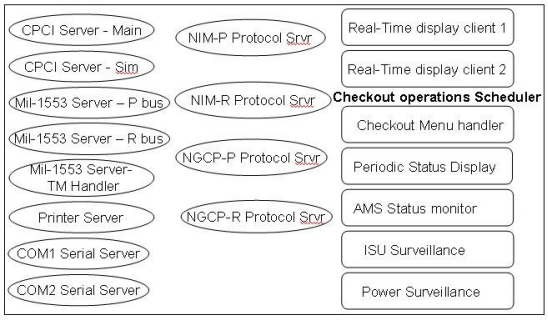


Fig [4]: Servers and Clients

The software is designed to have a graphical user-interface with multiple pull-down menus and uses object-oriented programming methodologies to enable code reuse.

3. Design decision to enable RT performance

Some of the design features that enable design as a real-time system are described here.

- Servers are defined for independent activities. These include the servers for Mil-1553 bus interface, real-time display, printer service etc. The priorities are apportioned to the servers in view of the criticality. Preemptive-priority based scheduling is used with equal priority servers run with round-robin scheduling. Hence it is ensured that processes with high criticality do not miss deadlines.

- Partitioning is done to ensure concurrency. For example, each of the Mil-1553 buses is handled by its own hardware device driver. Use of a common data server is done with shared memory. Real-time display is enabled using asynchronous messages, which means that the storage server need not wait for the readiness of the display servers. Fig[5] illustrates the interfaces.

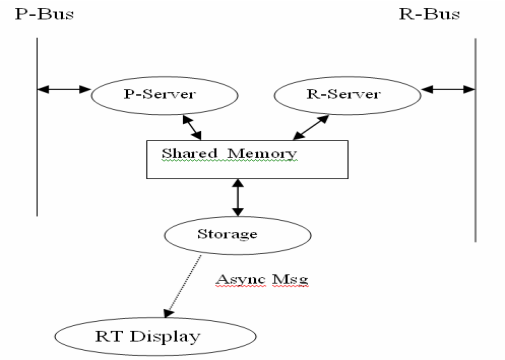


Fig [5]: Interfaces of Mil-1553 servers

- Simulation server receives asynchronous messages from the hardware device driver for Mil-1553 bus to generate the simulation data in synchronization with the onboard systems. No expensive polling is done here.

- Use of concurrent threads of execution is made with servers typically waiting in a receive-blocked state. A message sent from client initiates a computation, where the server replies with a reply immediately, freeing the client to continue its execution concurrently while the server services the request. Results are transmitted with another message transfer. This increases concurrency in the system.

- Priority inversion, which is a classic problem in any real-time system is avoided by running the servers at client's priority. The same server effectively runs at different priorities, flexibly handling the requirements.

- Use of global memory is made for fast data access between processes.

4. Design features in the software

The checkout software implements configurable definition of the test environment and any changes in the environment get reconfigured in the software. Use of multiple invocations of the same software to support similar behaviors in different system is another design feature. An example is the device drivers used for the two Mil-1553 buses. The checkout system uses unified data processing tools across checkout units and makes use of reusable components in the software.

VII. EVALUATION OF CHECKOUT SYSTEM

Evaluation of the checkout system to ensure correctness and completeness of the design is of paramount importance.

Timing measurements are conducted on critical areas of the code, to ensure timeliness. Analysis is done by instrumenting the code with activation of digital outputs and examining with logic analyzers. QNX provides a tool to monitor the system activity which helps in monitoring the load on the system.

1. Verification of timing

The following mechanisms are used to verify the timeliness.

- Use of the timestamp register of Pentium processor is made to verify timing in the checkout system.
- A critical test like simulated input profile test where inputs to NIM are simulated by the checkout system needs extensive study on the timing achieved. It is verified that system is able to simulate the data well before the expected time. Generation of a proxy message from Mil-1553 bus server to the simulation server, to synchronize the simulation is a critical message. The delay in this message was found to be less than 1 msec.
- Periodic surveillance tasks which must run at the rates of 1 sec and 2 sec were ensured of correctness by examining the logged data which showed updates at the expected periodicity.
- Storage of telemetry data corresponding to 6 channels to files is another critical area. The timing for this activity was measured and found to satisfy requirements. The QNX file-system server returns control to the calling client after moving the data to its area. Hence the client does not wait for the completion of the data storage activity.

VIII. IMPLEMENTATION OF CHECKOUT SYSTEM

The checkout software is version controlled. The checkout system is validated by a phase of designer-level validation where standalone checks are conducted, followed by tests with simulated a

data and with QM packages. The checkout system also goes through a formal process of qualification including SDRC and Test and evaluation.

IX. CONCLUSIONS

AMC-RESINS Checkout system has many timeliness requirements and hence has to be designed as a real-time system. The features that help in realizing such a design have been discussed here. The architecture of design which is extensible and configurable makes accommodation of changes and updates easy. The use of QNX and client-server architecture in the software has acted as enablers in the design of a checkout system that is correct as well as easily maintainable.

ACKNOWLEDGEMENTS

Mr. P. S. Veeraraghavan, Director, IISU, is gratefully acknowledged for encouragement and support of the work detailed here. The authors would like to acknowledge Mr. B. C. Vidwani, DD, LVIS for encouraging us in this work. The authors would also like to acknowledge Mr. S. Gopakumar, Head, Computer Division, Vikram Sarabhai Space Centre, Thiruvananthapuram, for his original contributions in the design of RESINS Checkout System.