VxWorks 653 and PikeOS Based Pitch Control Hard Real-Time Application

Rzeszow University of Technology Research Team Contribution to European Community SCARLETT Project

Sławomir Samolej
Department of Computer and Control Engineering
Rzeszow University of Technology
Rzeszow, Poland
e-mail: ssamolej@prz.edu.pl

Tomasz Rogalski, Andrzej Tomczyk
Department of Avionics and Control Systems
Rzeszow University of Technology
Rzeszow, Poland
e-mail: orakl@prz.edu.pl, atomczyk@prz.edu.pl

Abstract — The paper reports Rzeszow University of Technology Research Team contributions to the European Community SCARLETT project. The main objective of the research team was to prepare an avionic pitch control hard real-time application following ARINC 653 and ARINC 664. VxWorks 653 and PikeOS real-time operating systems are the application platforms. The paper mentions the general objectives of the SCARLETT and the ARINC specification 653. The main part of the paper includes a report of the application's development and its evaluation.

ARINC 653, VxWorks 653, PikeOS, hard real-time systems, control systems, avionics, Integrated Modular Avionics

I. INTRODUCTION

A modern airliner's computer systems include a wide set of hard real-time applications [7]. Typical avionic components such as cruise speed, flight level or pitch angle control systems are good examples. It is worth remarking that the avionic hard real-time systems usually have to both execute control tasks and exchange data between each other as well as the pilot's cockpit.

Recently, due to progress in engineering and economic stimuli, airborne real-time systems have been evolving from the so-called federated structure to something new - Integrated Module Avionics (IMA) [2], [3], [4], [5], [12]. The IMA concept has been introduced through these European funded research projects: PAMELA, NEVADA, and VICTORIA. The result of the projects was the first generation of IMA (IMA1G), currently onboard the A380, A400M and B787 aircraft. Following the IMA concept, modern onboard avionic subsystems (software applications) should be grouped in a limited set of standard microprocessor units. The microprocessor units and other electronic devices should communicate via standard network interface - Avionics Full Duplex Switched Ethernet (AFDX) [2], [3]. The so far physically and logically separated (federated) avionic units must be converted into groups of real-time applications controlled by real-time operating systems. The current implementation of IMA covers a limited range of aircraft functions but shows that it may bring some significant benefits: aircraft weight reduction and lowered maintenance costs.

Within the 7th Framework Program the SCARLETT [17] (Scalable and Reconfigurable Electronics Platforms and Tools) international research project was created to define and preliminarily test the next generation of IMA. Its main aim is to define a scalable, reconfigurable fault-tolerant driven and secure new avionics platform, called Distributed Modular Electronics (DME).

The Rzeszow University of Technology (RUT) Research Team (RUTRT), Poland, is taking part in one of the SCARLETT research paths: time-critical application development and testing. The developed application should be a part of an aircraft’s IMA system and make it possible to evaluate whether DME units can be effectively used as hard real-time applications platforms. The preliminary research results of RUTRT were published in [13]. This paper reports the current contributions of RUTRT in the SCARLETT project.

The next sections of the paper are organized as follows, firstly, the SCARLETT project is shortly introduced. Secondly, the ARINC specification 653 and ARINC 653 based real-time operation systems are mentioned. Thirdly, a Pitch Control Application (an illustrative example of the hard real-time avionic control system) developed by the RUTRT is presented. The final part of the paper includes some ARINC 653 based application development remarks and RUTRT future research and implementation plans.

II. SCARLETT PROJECT

The SCARLETT project [17] is a European enterprise joining 39 companies from 16 countries, including large industrial companies, public research centers, industrial research centers, universities, as well as small and medium enterprises. Its main aim is to provide and test a new set of standard hardware components that will be parts of Distributed Modular Electronics (DME). The new DME should provide:

- Scalability, portability and adaptability,
- Fault tolerance and reconfiguration capabilities,
- A minimum number of standardized electronic module types,
- Support for the full range of avionics function.

In the first stage of the SCARLETT project, the following DME units were defined:
• CPM (Core Processing Module) offers generic computation capability, an AFDX end system, and a set of generic bus + housekeeping I/Os capability (Field Bus such as CAN or Flexray, Discrete Inputs for pin-programming / servitudes and RS232 / 485 for in-shop maintenance). The objective is to suppress the A429 [1] interfaces of the CPM and the adherence between the functional I/O and the applications running on the CPM. Initially, SCARLETT identified three types of CPM:
  - High Performance CPM,
  - Time Critical CPM,
  - Avionics Server Function CPM.
• REU (Remote Electronics Unit) is an electronics box dedicated to a specific task and is mounted geographically close to where this task occurs. REUs are not generic units and therefore not part of the IMA perimeter. However, the interfaces of REUs to the IMA world shall be standardized.
• RDC (Remote Data Concentrator) is a type of equipment which supports the exchange of information between sensors/actuators (digital, discrete and analogue data) and aircraft digital communication networks (ADCN). The RDCs are located in pressurized areas close to sensors and effectors, which may be potentially remote from the associated processing resources rather than in the avionics bay.
• RPC (Remote Power Controller) is a power switching “unit based on SSPCs (Solid State Power Controllers). They are able to control the switching of loads with current limitation, self-test, leakage protection, etc. so they are not strictly equivalent to relays.”

Although the individual DME hardware devices may communicate with some external devices using a wide set of interfaces, such as CAN, RS or Field Bus, the main medium of the inter-DME units’ communication should be AFDX [2], [3] - a redundant and reliable Ethernet network developed and standardized by the European constructors of the avionics which equip the Airbus A380. This is a system used to support the internal communications within the plane, and not with external communications. The internal communications are primarily the data exchanged between the different components of the avionics. This communication is possible thanks to new AFDX Switches. The switches are star couples for the AFDX network. Like Ethernet switches, AFDX switches provide packetforwarding, and they also have additional features to guarantee the timing and bandwidth allocation of the entire network, but they only use multicast addresses (Virtual Links).

The AFDX internal message format should follow ARINC specification 664 [3]. The software modules executed on the hardware units should be developed according to ARINC specification 653 [4]. Both message format and software module development paradigms are explained in following sections.

Within the SCARLETT project, the aforementioned set of hardware modules is configured into several laboratory demonstrators. Their task is to assess whether the developed DME units may be effectively used in the next generation of IMA. Some research teams (e.g. RUTRT) are obliged to independently prepare software applications to be run on the demonstrator units.

III. ARINC SPECIFICATION 653P1-2

One of the main IMA concepts assumes the reduction of the number of the individual microprocessor units installed on-board. This should reduce the weight of the aircraft and lower the whole onboard computer system maintenance costs. However, this also entails a new paradigm in avionics development. The group of federated applications which have been executed up to now on separate microprocessor units (communicating by means of ARINC standard 429 based devices [1], for example) must become a set of real-time processes executed on one microprocessor. This will be managed by a specialized real-time operating system and communicate by means of a specialized Ethernet computer network. Provided that the operating system offers a standard API and fulfills safety requirements, this proposed solution significantly broadens the portability of avionic applications and makes it possible to develop and certify hardware and software independently.

A. Partitions

The IMA assumes that a set of time-critical and safety-critical real-time applications (avionics units) may be

Figure 1. Logical real-time operating system structure created according ARINC 653P1-2 specification ([4], pp. 4).
executed on one microprocessor module. To cope with this level of criticality, new real-time operating system architecture was suggested. The main document defining the generic system structure and its API is ARINC 653P1-2 [4]. Figure 1 shows the RTOS logical structure which was suggested in it.

The key concept introduced in the specification is the partition. It constitutes a kind of container for an application and guarantees that the execution of the application is both spatially and temporally isolated. The partitions have been divided into 2 categories: application partition and system partition. The application partitions are dedicated to executing avionics applications. They can exchange data with the environment by means of specific interface – APEX (APplication/EXecutive). The system partitions are optional and their main role is to provide services that had not been predicted in APEX, such as device drivers or fault management.

B. Hardware-Software Module Architecture

The ARINC 653P1-2 also includes some recommendations regarding the microprocessor module architecture for the specialized real-time operating system. The general schema of the architecture is presented in figure 2.

Each module may include one or more microprocessors. The hardware structure may require some core operating system modification but not the APEX interface. All the processes that belong to one application partition (real-time tasks) must be executed on one microprocessor and it is forbidden to allocate them to different microprocessors within the module or between modules. The application program should be portable between processors within the module and between modules without any modifications to the interface of the operating system core. The processes that belong to one partition may be executed concurrently. A separate partition-level scheduling algorithm should take be responsible for this. Inter-application (partition) communication is based on a ports and channels concept. The applications do not have the information about which partition the receiver of data is being executed on. They send and receive data via ports. The ports are virtually connected by channels which are defined in a separate level of system development.

The temporal isolation of each partition has been defined as follows, a major time frame is defined for each module. It is activated periodically. Each partition receives one or more time partition windows to be executed within this major time frame. Generally, time partition windows constitute a static cyclic executive [6]. Real-time tasks executed within the partition can be scheduled locally according to a priority-based policy. The order of the partition windows is defined in a separate configuration record of the system.

An important element of the module should also be a Health Monitor (HM). It is an operating system component that ought to monitor hardware, the operating system and application faults and failures. Its main task is to isolate faults and prevent failure propagation. As an example, the HM is permitted to restart a partition when it detects an application fault.

In general, the applications that reside within partitions may be developed by separate application providers. Thereafter a separate role in the IMA system development process has been proposed. This person or organization, an
integrator, has to collect the data regarding resources, timing constraints, communication ports and exceptions defined in each partition. Then the collected data is transferred into configuration records. The configuration record for each module is an XML document interpreted during compilation and consolidation of the software.

C. APEX Interface

The main part of ARINC 653P1-2 is the APEX interface definition. The APEX makes it possible to create platform-independent software that fulfills ARINC 653 requirements. The three main components of the interface are: real-time application creation and maintenance; partition management; intra- and inter-partition communication.

The application may be constructed as a set of (soft or hard) real-time processes, which are scheduled according to their priorities. It is possible to develop both an event- and a time-driven process activation policy.

The APEX interface provides a separate set of functions that enables the user to determine the actual partition mode and change it. The application may start the partition after the creation of all the application components or monitor the actual partition mode. It is also possible to restart the partition.

The synchronization of processes that belong to one partition may be achieved by the appropriate application of counting semaphores and events. The inter-process communication within the partition (intra partition communication) is made possible by means of APEX buffers (shared message queues) and APEX blackboards (shared variables). Inter-partition (application) communication is based on queuing port and sampling port communication units. The queuing port provides an inter-partition message queue, whereas the sampling port makes it possible to share variables between the ports. During system integration, the ports are connected by means of channels defined in the system configuration tables. The ports may be applied for communication with: other partitions, device drivers within the module or to exchange data between modules (by means of AFDX network interfaces).

D. Real-Time Operating Systems Compatible with ARINC 653P1-2

To RUTRT knowledge, there exist four mature real-time operating systems that meet ARINC 653P1-2 requirements: Wind River VxWorks 653 [18]; Sysgo PikeOS [19]; LynuxWorks LynxOS-178 RTOS; LynuxWorks LynxOS-SE RTOS [20]. RUTRT has been developing the hard real-time demonstrator application for VxWorks 653 and PikeOS operating systems.

IV. PITCH CONTROL HARD REAL-TIME APPLICATION

According to the IMA concept, RUTRT was an application provider. Its task was to prepare a sample hard real-time application following the ARINC specification 653 that should assess whether DME units may be adequate hardware platforms for such applications. The following subsections report the subsequent stages of the application’s development.

A. General Application Specification

The high-level RUTRT application specification was formulated as follows, an application ought to demonstrate whether it is possible to use SCARLETT philosophy to control an aircraft in the pitch control channel. The application should demonstrate if it is possible to use two synchronized actuators deflecting an aircraft elevator’s surfaces. An external controller computes the position of the elevator on the basics of flight parameters, pilot input, and implemented control procedures. The application communicates with actuator controllers and units which collect data from indicators via ADFX bus. The application should be prepared for two kinds of hardware configurations. For the first (called the Time Critical Demonstrator), the application should assess whether it is possible to accurately execute a distributed hard real-time application loaded on the elements of the demonstrator. For the second one (called the Reconfiguration and Maintenance Demonstrator), the application should have a background application. It should be possible to detect whether the control systems work correctly, even if the reconfiguration of other applications is occurring. Figure 3 shows an example of the distribution of a RUTRT application between the SCARLETT hardware modules.

![Figure 3. Example RUTRT’s application distribution scenario.](attachment:image)

A part of the control application is executed on a CPM module. Two other parts are allocated to two REUs. The REUs are directly connected to actuators. The control application modules communicate via ADFX.

B. Control System Project

To fulfill the application requirements RUTRT proposed the following control system architecture. The control application would be a Pitch Control Application (PCA) controlling two actuators (brushless motors) connected to a load (an elevator). Each actuator is controlled by a separate cascade of controllers, as in fig. 4. The single actuator
control system includes an internal current control loop, a velocity control loop, and a position control loop. The Flight Control Algorithm is a superior module that generates the position demand signal for both actuator control subsystems. It collects signals from the aircraft simulator, pilot simulator and actuator. The real-time aircraft simulator makes it possible to adjust the dynamics of the PCA to realistic values. It reflects typical airliner behavior.

C. Control Application Distribution Scenarios

One of the SCARLETT program goals is to evaluate the quality of distributed control applications, where some parts of the application are housed on different devices. Therefore the RUTRT Pitch Control Application was developed with the intention of distributing some of its parts to separate hardware modules. Figure 4 shows the possible control application distribution variants proposed by the demonstrator’s developers (SCARLETT project partners). In the first variant (CPM1_2+REU1_2+REU2_2), the Pilot, Aircraft, Flight Control Algorithm, and two position controllers are housed on the CPM module, whereas velocity control algorithms are housed on separate REUs. In the second variant (CPM1_1+REU1_1+REU2_1), the CPM module executes the superior part of the control system (Pilot, Flight Control Algorithm, and Aircraft), with the position and velocity controllers housed on REUs.

D. Control Application Configuration Assumptions

Bearing in mind application specification and the likelihood of different allocation scenarios of the control application components, RUTRT proposed the following detailed assumptions regarding the developed application:

- The position controller must be movable. It should be possible to install it both on CPM or REU hardware modules. Therefore it should belong to a separate ARINC 653 partition.
- All data packages sent between partitions should be in accordance with ARINC 664P7 [3]. This way, the partition application does not have to be changed even if some of the partitions will be moved to other hardware units. ARINC 664P7 messages will be ready to be sent via the AFDX network.
- Some verification procedures should be built into the application software. They should provide information about the quality of control system and the soundness of the system structure.

E. Pitch Control Application Simulator

The development of both RUTRT Pitch Control Application and SCARLETT hardware units occurred simultaneously. Therefore RUTRT decided to create a complete software simulator of the entire control system. It includes both controllers and real-time simulators of the hardware units which would be replaced by real devices. Figure 5 includes this structure. Figure 5 also depicts the PCA structure from figure 4 in a schema based on ARINC 653. The simulator consists of four ARINC 653 partitions. P1, P2, and P3 are main software modules that RUTRT had...
to provide. The P4 partition includes real-time software simulators of SCARLETT REU modules.

The first (P1) partition includes some real-time simulators of Pilot and Aircraft. It also includes a Flight Control Algorithm (FCA) block that collects signals from Pilot, Aircraft and actuator modules and produces the desired pitch angle signals for controllers. The last module built into the P1 partition is an Error Estimator. It makes it possible to monitor both communication channels and the quality of these timing constraints originate from control engineering needs. The major time frame includes two Spare regions. These regions may be applied by other software modules loaded on the target hardware module. The HARD attribute attached to each of the real-time tasks instructs the ARINC 653 Health Monitor (which is built into the operating system structure) that if any task misses its deadline, the core operating system must be informed about it. This in consequence, imposes the operating system to take an appropriate action. The Health Monitor procedures may even reload the whole partition that signals the missing timing constraints event.

control system during the system run-time. This module will be presented in detail in following sections. The Error Estimator, Pilot, Aircraft and FCA modules are separate real-time tasks. The second (P2) partition includes the first position controller algorithm (CPx1), running as a separate real-time task. Identically, the third (P3) partition includes the second position control algorithm (CPx2). The fourth (P4) partition includes 2 real-time REU simulators: the velocity controller modules, joined with actuators attached to the first (CVx1+Actu1) and second (CVx2+Actu2) control loops.

For the intra-partition communication, ARINC 653 blackboards were applied, whereas the inter-partition communication is based on ARINC 653 sampling ports and channels. Figure 5 includes both ARINC 653 port names and communication channels defined by RUTRT. The port names are the only reference to the application communication interface. The same port names exist in the application configuration record and this makes it possible to connect these ports by means of the channels.

**F. PCA Timing**

The RUTRT’s Pitch Control Application was developed to fulfill the timing constraints shown in fig. 6. Major application frame and partition windows were defined according to ARINC 653 and encoded in the application’s
XML configuration file. Table I includes all the PCA tasks’ ARINC 653 real-time parameters.

The P1 partition and all its real-time tasks acquired 2 time-partition windows within the major frame due to their computing complexity. The real-time tasks that belong to P2, P3, and P4 partitions ought to finish their computations within the minor cycle. These timing constraints originate from control engineering needs. The major time frame includes two Spare regions. These regions may be applied by other software modules loaded on the target hardware module. The HARD attribute attached to each of the real-time tasks instructs the ARINC 653 Health Monitor (which is built into the operating system structure) that if any task misses its deadline, the core operating system must be informed about it. This in consequence, imposes the operating system to take an appropriate action. The Health Monitor procedures may even reload the whole partition that signals the missing timing constraints event.

![Figure 6. RUTRT Pitch Control System Simulator Partition Timing.](image)

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Real-Time Tasks Parameters</th>
<th>Stack Size</th>
<th>Base Priority</th>
<th>Period [ms]</th>
<th>Time Capacity [ms]</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pilot</td>
<td></td>
<td>4096</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>HARD</td>
</tr>
<tr>
<td>Aircraft</td>
<td></td>
<td>4096</td>
<td>19</td>
<td>20</td>
<td>20</td>
<td>HARD</td>
</tr>
<tr>
<td>FCA</td>
<td></td>
<td>4096</td>
<td>18</td>
<td>20</td>
<td>20</td>
<td>HARD</td>
</tr>
<tr>
<td>ERROR ESTIM.</td>
<td></td>
<td>4096</td>
<td>17</td>
<td>20</td>
<td>20</td>
<td>HARD</td>
</tr>
<tr>
<td>CPx1</td>
<td></td>
<td>4096</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>HARD</td>
</tr>
<tr>
<td>CPx2</td>
<td></td>
<td>4096</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>HARD</td>
</tr>
<tr>
<td>VPx1</td>
<td></td>
<td>4096</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>HARD</td>
</tr>
<tr>
<td>VPx2</td>
<td></td>
<td>4096</td>
<td>9</td>
<td>5</td>
<td>1</td>
<td>HARD</td>
</tr>
<tr>
<td>Actu1</td>
<td></td>
<td>4096</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>HARD</td>
</tr>
<tr>
<td>Actu2</td>
<td></td>
<td>4096</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>HARD</td>
</tr>
</tbody>
</table>

G. PCA Real-Time Analysis

As it was mentioned before, the PCA timing restrictions were forced by the control engineers which specified the system. P2, P3 and P4 partitions acquired 1 ms time frames for their computations and are activated every 5 ms. This timing constraints guarantee sufficient frequency (200Hz) of PID algorithm repetition which in consequence guarantees the sufficient quality of control. P1 partition is 4 times “slower” than others. To the RUTRT knowledge the algorithms computed in this partition may produce the results in such rate without any affect on the control system quality. This made it possible to save some computational time for other application that will be installed on the same hardware module.

All of the algorithms applied in the PCA are controllers or simplified numerical procedures which solve some differential equations. During the system development the worst case computation time analysis for each of the algorithm was conducted. It was proved and experimentally checked that the algorithms can meet the aforementioned timing constraints.

The local real-time tasks priorities (which were defined within the partitions) reflect the order of the computations the task should follow. This approach is essential especially in P1 partition. It is expected that the Pilot and Aircraft real-time tasks should finish their computations and produce their computation results before the FCA task starts.

All the communication mechanisms applied in the PCA are both shared variables and monitors [6]. This solves the mutual exclusion problem. The shared variable access (at the operating system level) is conducted according to the priority inheritance protocol [6], [7]. It is easily to notice that the developed PCA communication structure prevents form the deadlock phenomena, too.

H. Inter-Partition Data Exchange Structure

One unified inter-partition data exchange format for all channels and ports for the RUTRT PCA was suggested. This is shown in figure 7. The message structure follows the requirements formulated in ARINC 664 [3] specification. Up to 9 16-bit integer digits per message can be transported.

1. Built-in Self-Testing Procedures

According to application specification, the PCA should provide a set of test procedures informing the user about the quality of the system before or during runtime. Therefore the following extensions of the basic PCA structure depicted in fig. 4 were applied.

1) A separate error (ERR) port was included in the P1 partition structure.
2) A new Error Estimator real-time task was introduced in the P1 partition.
3) It has been decided that the quality of the PCA’s subsystem service would be run simultaneously with the control procedures.
4) The quality of service procedures has been divided into two subsystems:

a) The channel connection detector permanently monitors the channels of the system and indicates whether all the links are properly connected. This subsystem
guarantees that all system components send and receive data from the proper ports and software modules. The channel connection detector checks whether all channels are configured according to the assumed structure. During the runtime of the application, apart from control application data, a separate set of values is sent via the channels. Some additional procedures included in RUTRT’s application control blocks make it possible to detect whether the application’s ports receive data from the assumed sources. The detector also makes it possible to reveal data transmission faults. It produces a 16-bit word, where each bit value equals 0 the channel works properly. If not, the channel is badly established or the function block connected to the channel produces incorrectly formulated data packages.

b) The control system error detector signals to the system operator that the quality of control is below the assumed acceptable level. It may reveal some problems with communication or may suggest that the control system's parameters should be refined. Each single actuator control system is monitored by a separate control procedure built into the ErrorEstimator block. This procedure collects all the possible signals from the PCA modules and assesses the quality of control by:

- Detection of the actuator’s angular velocity oscillations,
- Signalization of the elevator’s position error that exceeds the assumed threshold value.

Finally, the control system error detector produces 2 major and 4 minor values. The major values describe the quality of control of the appropriate actuator. In general, if they both have the value of 0, both control loops work correctly. If they have values between 1 and 3, they include the error code of the monitored control loop. Table II includes the error codes’ interpretation.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Code Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Control system is working correctly</td>
</tr>
<tr>
<td>1</td>
<td>Deflection velocity oscillations in occurs</td>
</tr>
<tr>
<td>2</td>
<td>Tracking error occurs</td>
</tr>
<tr>
<td>3</td>
<td>Both tracking error and control system oscillations occurs</td>
</tr>
</tbody>
</table>

J. PCA Tests
Three stages of RUTRT Pitch Control Application (RUTRT PCA) tests were conducted. The first group of tests assessed the application from a control engineering point of view. The second group of tests covered execution of built-in self-test procedures of the RUTRT PCA. The third group of test included the application of VxWorks 653 analysis tools for PCA timing and communication evaluation.

a) A control engineering-based application evaluation was conducted as follows, the application’s goal was to perform in an AFDX environment and to control actuators using data incoming from the network. To guarantee more realistic test conditions, the actuator controller’s software was put into a simulated flight control environment. A real-time aircraft simulator was used to obtain realistic values from the application. It also produced a reference signal for the Flight Control Algorithm module (comp. Fig. 4). A model of longitudinal dynamics of a DC-8 airplane flying at a speed of 251 [m/s] at flight level 330 was applied during the real-time simulation tests.

The quality of the control system was monitored by analyzing of actuator response feedback and comparing it to the desired values generated by appropriate control blocks. Figure 8 shows the positively-assessed position-tracking test results.

b) A built-in self-test subsystem evaluation was performed as follows, both the channel connection detector subsystem and control system error detector were tested in detail during long-term tests of the RUTRT PCA. All possible channel malfunctions were simulated and properly detected. Similarly, a low quality (from the control engineering point of view) PCA was executed. The control system error detector successfully signaled the lower quality control signals.
c) A VxWorks 653 analysis tools-based system evaluation was conducted as follows, during the real-time PCA simulation, the a System Viewer toolset was applied to collect standard timing and communication events that occurred during the monitoring session. Figure 8 includes a sample graphic representation of the collected data. It illustrates the partition scheduling during the system execution. Thorough analysis of the System Viewer data made it possible to confirm that the timing and communication requirements were fulfilled. A similar evaluation is planned for the final application tests.

![Figure 8. The Time Domain Graph of Desired and Final Positions of Elevator.](image)

K. PCA Implementation Remarks

The PCA was written in C language and follows the ARINC 653 specification. The first software layer consists of a set of periodic hard real-time tasks. The tasks have some timing constraints attached. Each real-time task has a main function joined with it. The function collects data from the input communication objects (blackboards or sampling ports), calls the appropriate control block algorithm, sends the computed data to its output objects, and finally suspends its execution. The function is periodically activated by the core operating systems mechanisms. The second software layer constitutes control procedures that are called from the task’s main functions. Control procedures were developed and tested independently from the real-time task configuration level.

Only the ARINC 653-based communication objects and task-managing mechanisms were applied in the PCA. This made it possible to develop some distributable pieces of application that can be loaded onto any hardware modules. To RUTRT knowledge, it is impossible to prepare a VxWorks 653-based loadable module capable of being loaded onto a PikeOS platform, and vice versa. However it is possible to create ARINC 653 based application source code which is able to be processed by both VxWorks 653 and PikeOS compilers. The only drawback is that VxWorks 653 and PikeOS producers use different names for the header files which include ARINC 653 function calls.

V. CONCLUSIONS AND FUTURE RESEARCH

The paper reports RUTRT contributions to the European Community SCARLETT project. RUTRT’s main objective was to prepare a hard real-time application following ARINC 653 and ARINC 664 and using VxWorks 653 and PikeOS operating systems as the application platforms. An avionic Pitch Control Application (PCA) was chosen as a good example of a system which fulfills the assumed requirements. The paper mentions the general objectives of the SCARLETT project and the specification ARINC 653, both of which played a crucial role in PCA development. The main part of the paper includes a report of the PCA development and evaluation. It covers system specification and the most important project development perspectives: control system structure, ARINC 653-based application structure, real-time timing parameters, built-in self-testing procedures.

In the current state of the RUTRT PCA development, the application is being preliminarily integrated and tested on target hardware platforms. Its preliminary version (delivered earlier to the system integrator) was successfully integrated to the target hardware platforms. Simultaneously, the application documentation and application configuration files are being fitted and evaluated by separate system integrators. The next stage of application development will be the final application integration with hardware and software modules delivered by other partners of the SCARLETT project.
ACKNOWLEDGMENTS

Research reported in the paper is funded by SCARLETT 7th European Framework Project, Grant Agreement No. FP7-AAT-2007-RTD-1-211439. Some of hardware components used in the research published within this paper were financed by the European Union Operational Program - Development of Eastern Poland, Project No. POPW.01.03.00-18-012/09. Ralph McGaw assisted in proofreading the English version of this text.

REFERENCES