An IEC61499 Execution Environment for an aJile-based Field Device

Kleanthis Thramboulidis, Nikolaos Papakonstantinou
Electrical & Computer Engineering
University of Patras, Greece
thrambo@ee.upatras.gr

Abstract—The IEC61499 standard is the first step towards the development of the next generation agile manufacturing systems where distribution, interoperability and re-configuration are between the most important requirements. Development environments that will demonstrate the applicability of this standard in industrial environments are still missing. In this paper an approach that allows the execution of IEC61499 based control applications on an industrial field device is presented. This approach allows the control engineer to design the control application in terms of Function Block network diagrams, and subsequently exploit the model-driven development paradigm to run the control application on a single-chip real-time Java microcontroller especially designed for real-time embedded applications.

I. INTRODUCTION

The IEC 61499 standard [1] enhances the IEC 1131 Function Block (FB) model to exploit in the development process of open, interoperable, distributed control applications (DCAs), many of the well defined and already widely acknowledged benefits introduced by object technology. However, even though the interest from academia for this standard is growing during last years with many researchers working to exploit IEC 61499 in factory automation [2], industrial implementations that will demonstrate the applicability of this standard to factory automation are missing. FBDK, CORFU and Archimedes ESS are the only publicly available Engineering Support Systems (ESS) that support a 61499-based development process for research and demonstration. IsaGraf of ICS Triplex [3] is the first commercially available tool that provides support for the IEC61499. However, the IEC61499 support is strongly coupled with the already supported by the tool 1131 FB model that imposes restrictions on the IEC61499 model.

In the context of this work the possibility of utilizing IEC61499 compliant ESSs with industrial field devices and specifically with the Luciol device is examined. Luciol is an industrial field device that exploits the aJ-100 processor to provide an ideal execution environment for real-time embedded applications [4]. It is bundled with a J2ME-based Java run-time system, optimizing application build tools, debugger, third-party IDE’s and an evaluation board to provide a complete solution for implementing real-time embedded Java applications. However, the control engineer must be a specialist in real-time Java and other current software engineering practices that make the use of the device restrictive to the control engineers’ community where other practices such as the Function Block (FB) concept are currently used. This problem is addressed in the context of this work integrating Archimedes ESS with the Luciol field device. A flexible development process that is supported by the Archimedes ESS was defined and an IEC-compliant execution environment for this aJile-based [5] field device was defined. A prototype implementation is under development while preliminary performance results show better execution times than the ones obtained using real-time Java implementations running on conventional CPUs.

The remainder of this paper is organized as follows. In the next section a brief introduction to the real-time Java and the real-time Java processor used is given. In section III, the proposed environment is presented and the development process is described. In section 4, the prototype implementation is described and preliminary performance results are given. Finally the paper is concluded in the last section.

II. REAL-TIME JAVA

A. The real-time Java Specification

Safety, exception handling, multithreading and garbage collection, made Java popular and successful. However, considering the real-time and embedded domains, characteristics such as large size, the nondeterministic behaviour and the poor performance are first class issues that have to be addressed for the language to successfully be used in these domains. The Real-Time Specification for Java (RTSJ) [6][9] is an attempt to address these problems. RTSJ in order to make Java more real-time defines: a) some modifications to the semantics of the JVM, and b) an additional package, namely java.real-time that defines extra classes and interfaces. RTSJ received a high acceptance from vendors that resulted in a variety of real-time Java implementations and products. RI (http://www.timesys.com), Jamaica (http://www.aicas.com) Aero JVM (http://www.aero-project.org/), jRate (http://jrate.
sourceforge.net/) are example implementations.

B. The real time Java Processor

The real time Java Processor aJ-100 [5] was selected as target environment for the execution of IEC61499 FB-based control applications. The aJile Systems aJ-100, which is a single-chip Java microcontroller that directly execute Java Virtual Machine (JVM) bytecodes, opens new horizons on the application of RTSJ in embedded devices. It implements real-time Java threading primitives, such as, wait, yield, notify, monitor enter/exit, as extended bytecodes to eliminate the need for a traditional RTOS. The selected architecture results in an extremely low executive overhead with thread to thread context switch times less than 1 µsec. The aJ-100 is ideally suited for real-time networked embedded products such as industrial controllers, smart mobile devices, consumer appliances and automotive communications devices [5].

Its run-time environment is based on Sun’s Java 2 Micro Edition (J2ME) Connected Limited Device Configuration (CLDC) Java runtime environment. According to aJile, application developers using commercial Java IDEs, can create standalone real-time Java applications totally in Java with the performance and memory efficiency of systems programmed in C and assembly.

III. THE PROPOSED ENVIRONMENT

A. The Luciol field device.

The Luciol field device is organized around an aJ100 processor running at 70 MHz with 1Mb of RAM and 4 Mb of FLASH. It is composed of a set of hardware modules (CPU, board, Digital and Analog I/O modules) and system software. The CPU is in charge of RS232/485 and Ethernet management and runs only the high-level data processing. The board module is responsible for the time stamping and synchronization and acts as the interface between the I/O modules and the CPU. The Luciol field device was designed for operation in environments which are characterized by intensive electromagnetic noise, such as power and power distribution stations. It is currently mainly used in Protection and Control Systems for Electrical Utilities with National Grid, a network utility specialized in the management of large and complex energy delivery networks in Great Britain, being an example.

Luciol is entirely programmed in C and Java rather than with IEC61131 languages, such as Ladder, SFC, IL, etc. like a programmable controller. It offers to system integrators and equipment manufacturers a powerful Integrated Development Environment (IDE). However, this environment imposes restrictions to the control engineers’ community where other practices such as the Function Block concept are currently used. A time consuming and costly training process is required for control engineers to be able to use this environment effectively. To address this problem a FB61499-based development environment is proposed in this work, to allow the control engineer to develop the control applications using the FB construct and automatically generate the executable model for the network of Luciol devices without any knowledge of real-time Java.

B. Towards a FB-based IDE

The proposed environment should allow the deployment of distributed control applications over the Internet, so as to enable life-cycle management remotely and securely over a network of Luciol devices. Unfortunately the approach adopted by the Archimedes RTSJ-AXE package [7][8] for the deployment and re-deployment of control applications can not be adopted in the case of Luciol device due to the restriction imposed by the CLDC 1.0 that does not support the ClassLoader class, that is extensively used in the RTSJ-AXE package. To overcome this limitation a quite different architecture than the one adopted in RTSJ-AXE is proposed. A Luciol application was defined to provide the Luciol device with the capability of communicating over TCP/IP with an IEC-compliant ESS (as for example CORFU or Archimedes ESSs), as shown in fig. 1. This application contains the Deployment Management Entity (DME) that communicates through TCP/IP with the ESS and provides to the ESS the functionality required to download FB types, create FB instances, and their connections, etc. to setup and initiate the execution of this part of the control application that have to be executed on the specific device. Since the ClassLoader class is not supported, the functionality of the BootLoader class of the aJile environment was utilized to provide the DME application with the ability to upload to the device the IEC-compliant control application.

C. Deployment support for the Luciol device

DME provides the infrastructure required for the Luciol device to be used in a deployment process through the internet. For the Fig. 1. An IEC61499-compliant control application distributed on two Luciol devices
A specific model-to-model Archimedes system platform transformer will be utilized to get the Luciol-compliant java files of the control application, as shown in Fig. 2. This application will include except for the required FB types, a launcher sub-application, as shown in fig. 1.

The image file of this application will be created using JEMBuilder with the option to provide image file to be downloaded through the DME application that is running in JVM1. The so produced application will be downloaded to the Luciol DME application that is running as JEMBuilder with the option to provide image file to be deployed to the Luciol device. The image file of this application will be created using launcher sub-application, as shown in fig. 1.

A specific model creator will be developed to automatically generate the executable model of the FB based design specifications as well as run-time re-configuration. Between the most important classes of this category are: DeploymentManagerEntity (DME), DataConnection Manager (DCM), and the EventConnection Manager (ECM).

During the establishment of an event connection between two FB instances, the application launcher subscribes the consumer FB to the ECM by calling the subscribe() method. The application launcher can subscribe an FB instance to more than one output events that allows an FB to receive notifications from multiple event producer FB instances.

The Luciol-AXE implementation Framework is a set of classes that are re-used by the FB2aJile model interpreters to generate the executable model of the FB based design specifications. FBtype, BasicFBType, ECC, ECState, InputEventMonitor, etc. are among the most important classes of this framework.

During run-time when an FB instance reaches the point in its ECC where it should signal its output events, the fire() method, notifies the ECM for the specific event. ECM which is an active object (thread) notifies the FB instances, which have subscribed for the particular event, writing the event to their
inputEventMonitors. The consumer FB instance is awakened, it reads from the InputEventMonitor the input events and enters it’s ECC in order to check if any of its transition conditions is satisfied.

B. Performance results

To analyze the timing behaviour of our prototype execution environment and compare it with the conventional RTSJ platforms, the Counter example application was developed, deployed and re-deployed. The Counter application is a rather simple application that abstractly represents the part of a real system that has to monitor and count the appearance of external events and respond to them. The system counts external events starting from a certain initial value, using given step and moving upwards and downwards between two given values. A detailed description of the counter example application is given in [7].

As target device for the case of RTSJ implementations a PC was used with Intel Pentium 4 CPU at 2,40 Ghz with 512Mb RAM. The software platforms used to test the features of RTSJ-AXE package are:

1. Fedora Core 2 Kernel 2.6.5 running JamaicaVM 2.6 Release 2 (Build 702) of Aicas Gmbh.
2. TimeSys Linux/RT (GPL version) 4.1 Kernel 2.4.21 with TimeSys Reference Implementation (RI) ver 1.0-547.1. RI implements all the mandatory features of RTSJ being a fully compliant implementation.

Figure 3 presents timing characteristics regarding FB execution time for an FB with algorithms having zero execution time and one event signalled per EC action. The average execution time was calculated over 1000 measured values. FB execution time was measured for 1, 2 and 3 sequential transitions and return to the EC initial state. I should be noted that Luciol runs at 70Mhz while the PC used for RI and Jamaica runs at 2,40 Ghz, and Jamaica was executed without a RTOS.

Figure 3. Timing characteristics of FB execution.

Table I presents the timing characteristics of the deployment process. It must be noted that the 5 instances were all of different FB types so the load time for 5 different classes is responsible for the long total instantiation time.

<table>
<thead>
<tr>
<th>Action</th>
<th>#of times performed</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB instantiation</td>
<td>5</td>
<td>25 ms</td>
</tr>
<tr>
<td>Data connection</td>
<td>10</td>
<td>2 ms</td>
</tr>
<tr>
<td>Event connection</td>
<td>8</td>
<td>&lt;1 ms</td>
</tr>
<tr>
<td>Start FB instance</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Even though many researchers have already reported encouraging results working with different aspects of the IEC61499 standard, industrial implementation that will demonstrate the applicability of this standard in factory automation are missing.

In this work an approach that utilizes the IEC61499 Function Block model in industrial environment through the use of the Luciol device is described. A prototype execution environment based on the aJile CPU that is used by the Luciol device is under development. Archimedes ESS is used to support the development process of the control application as well as its deployment on a network of Luciol devices. Preliminary performance results provide better execution times than the ones measured on real-time Java implementations running on conventional CPUs.

V. CONCLUSIONS

Even though many researchers have already reported encouraging results working with different aspects of the IEC61499 standard, industrial implementation that will demonstrate the applicability of this standard in factory automation are missing.

In this work an approach that utilizes the IEC61499 Function Block model in industrial environment through the use of the Luciol device is described. A prototype execution environment based on the aJile CPU that is used by the Luciol device is under development. Archimedes ESS is used to support the development process of the control application as well as its deployment on a network of Luciol devices. Preliminary performance results provide better execution times than the ones measured on real-time Java implementations running on conventional CPUs.

TABLE I. Deployment timing characteristics of Counter application.

<table>
<thead>
<tr>
<th>Action</th>
<th>#of times performed</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB instantiation</td>
<td>5</td>
<td>25 ms</td>
</tr>
<tr>
<td>Data connection</td>
<td>10</td>
<td>2 ms</td>
</tr>
<tr>
<td>Event connection</td>
<td>8</td>
<td>&lt;1 ms</td>
</tr>
<tr>
<td>Start FB instance</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

ACKNOWLEDGEMENTS

We gratefully thank I.T.Lity Ltd. for partially funding this project and Patrice Mousset for his support regarding the structure of the Luciol device.

REFERENCES