South African Computer Journal
Number 24
November 1999

Suid-Afrikaanse Rekenaar-Tydskrif
Nommer 24
November 1999

Special Issue: SAICSIT '99
Contents

Preface
P. Machanick .................................................. 1

Research Articles
Active Learning: Issues and Challenges for Information Systems and Technology
RD Quilling, GJ Erwin and O Petkova .................................................. 5

A Generic Modelling Framework for Interactive Authoring Support Environments
Paula Kotze ............................................................... 15

O Petkova and JD Roode .................................................. 26

An Information-Theoretic Semantics for Belief Change
T Meyer ............................................................... 33

A Complexity Metrics Model for Software Correction
A Törn, T Andersson and K Enholm .................................................. 40

A Conceptual Design for High-Volume Data Processing of Warehouse Database into Multidimensional Database
Paisarn Trakulsuk and Vichit Avatchanakorn ...................................... 49

A Pragmatic Approach to Bitemporal Databases: Conceptualization, Representation and Visualisation
Chiyaba Njovu and WA Gray .................................................. 58

A Building Recognition System
SP Levitt and B Dwolatzky .................................................. 68

Computer Programming and Learning to Write
John Barrow ............................................................... 77

Co-operating to Learn using JAD Technologies
TA Thomas ............................................................... 87

Critical Success Factors for the Implementation of DSS at a Selection of Organisations in Kwazulu/Natal
URF Averweg and GJ Erwin .................................................. 95

Enhancing the Predictability of Two Popular Software Reliability Growth Models
Peter A Keiller and Thomas A Muzzuchi ...................................... 105
Generalised Unification of Finite Temporal Logic Formulas
Scott Hazelhurst .......................................................... 110

Harmonizing Global Internet Tax: A Collaborative Extranet Model
E Lawrence and B Garner .................................................. 119

Improving Object Oriented Analysis by Explicit Change Analysis
Lui Yu, Siew Chee Kong, Yi Xun and Miao Yuan ..................... 128

Reconciling the Needs of New Information Systems Graduates and Their Employers in Small, Developed Countries
Rodney Turner and Glenn Lowry ...................................... 136

Shortest Delay Scheduling Algorithm for Lossless Quality Transmission of Stored VBR Video under Limited Bandwidth
Fei Li, Yan Liu, Jack Yiu-Bun Lee and Ishfaq Ahmad .................. 146

Software Croma Keying in an Immersive Virtual Environment
Frans van der Berg and Vali Lalioti .................................... 155

Some Automata-Theoretic Properties of $\cap$-NFA
Lynette van Zijl and Andries P J van der Walt ....................... 163

Statistical Analysis of an Automated Computer Architecture Learning Environment
JT Waldron, J Horgan and G Keogh ..................................... 168

The CILT Multi-Agent Learning System
Herna L Viktor ..................................................................... 176

The Development of a Generic Framework for the Implementation of Cheap, Component-Based Virtual Video-Conferencing System
Soteri Panagou and Shaun Bangay ....................................... 185

The Role of Experience in User Perceptions of Information Technology: An Empirical Examination
Meliha Handzic and Graham Low ........................................ 194

What are Web Sites Used for: Cost Savings, Revenue Generating or Value Creating?
Man-Ying Lee .................................................................... 201

New Ideas Papers
Approaches to Video Transmission over GSM Networks
Bing Du and Anthony Maeder ............................................. 210

From Information Security Baselines to Information Security Profiles
Rossouw von Solms and Helen van der Haar ......................... 215

Grounded Theory Methodology in IS Research: Glaser versus Strauss
J Smit ............................................................................... 219

Introducing a Continuum of Abstraction-Led Hierarchical Search Techniques
Robert Zimmer and Robert Holte ....................................... 223
Multimedia as a Positive Force to Leverage Web Marketing, with Particular Reference to the Commercial Sector

Stan Shear ................................................................. 229

Understanding HCI Methodologies

Peter Warren ................................................................. 234

Electronically Published Papers


Experience Papers

A Java Client/Server System for Accessing Arbitrary CANopen Fieldbus Devices via the Internet

Dieter Bühler, Gerd Nusser, Gerhard Gruhler and Wolfgang Küchlin .................................................. 239

An Object-Oriented Framework for Rapid Client-side Integration of Information Management Systems

Ralf-Dieter Schimkat, Wolfgang Küchlin and Rainer Krautter ............................................................. 244

Distributed Operating Systems: A Study in Applicability

Jürgen Prange and Judith Bishop ................................................................. 249

Formal Verification with Natural Language Specifications: Guidelines, Experiments and Lessons so far

Alexander Holt ................................................................. 253

Introducing Research Methods to Computer Science Honours Students

Vashti Galpin, Scott Hazelhurst, Conrad Mueller and Ian Sanders ............................................................. 258

Visualising Eventuality Structure

ST Rock ................................................................. 264

Electronically Published Papers

SAICSIT'99
South African Institute of Computer Scientists and Information Technologists
Annual Research Conference 17-19 November 1999
Prepare for the New Millennium
Is there life after y2k?
Mount Amanzi Lodge, Hartebeespoort, South Africa

Sponsors

http://www.cs.wits.ac.za/
PHDS research/PHDS.html
Think different:
Preface

Philip Machanick, Overall Chair: SAICSIT’99

Running SAICSIT’99, the annual research conference of the South African Institute for Computer Scientists and Information Technologists, has been quite an experience.

SAICSIT represents Computer Science and Information Systems academics and professionals, mainly those with an interest in research. When I took over as SAICSIT president at the end of 1998, the conference had not previously been run as an international event. I decided that South African academics had enough international contacts to put together an international programme committee, and a South African conference would be of interest to the rest of the world.

I felt that we could make this transition at relatively low cost, given that we could advertise via mailing lists, and encourage electronic submission of papers (to reduce costs of redistributing papers for review).

The first prediction turned out to be correct, and we were able to put together a strong programme committee.

As a result, we had an unprecedented flood of papers: 100 submitted from 21 countries. As papers started to come in, it became apparent that we needed more reviewers. It was then that the value of the combination of old-fashioned networking (people who know people) and new-fashioned networking (the Internet) became apparent. While the Internet made it possible to convert SAICSIT into an international event at relatively low cost, the unexpected number of papers made it essential to find many additional reviewers on short notice. Without the speed of e-mail to track people down and to distribute papers for review, the review process would have taken weeks longer, and it would have been much more difficult to track down as many new reviewers in so little time.

Even so, the number of referees who were willing to help on short notice was a pleasant surprise.

The accepted papers cover an interesting range of subjects, from management-interest Information Systems, to theoretical Computer Science, with subjects including database, Java, temporal logic and implications of e-commerce for tax.

In addition, we were very fortunate in being able to invite the president of the ACM, Barbara Simons as a keynote speaker. Consequently, the programme for SAICSIT’99 should be very interesting to a wide range of participants.

We were only able to find place in the proceedings for 36 papers out of the 100 submitted, of which only 24 are full research papers. While this number of papers is in line with our expectation of how many papers would be accepted in each category, we did not have a hard cut-off on the number of papers, but accepted all papers which were good enough, based on the reviews. Final selection was made by myself as Programme Chair, and Derrick Kourie, as editor of the South African Computer Journal. Additional papers are published via the conference web site.

We believe that we have put together a quality programme, and hope you will agree.

Acknowledgments

I would like to thank the South African Computer Journal production team, Andries Engelbrecht and Herna Viktor, respectively from the Department of Computer Science and Informatics, University of Pretoria, for their work on producing the proceedings.

The reviewers listed overleaf did an excellent job: many wrote very detailed reports, sometimes after being called in on very short notice. Inevitably, there were some glitches resulting from the unexpected workload, but the buck stops with the programme chair: I promise to do better next time.

I would also like to thank my own department for putting up with the extra work and expense that running a conference entails. I tried not to burden them with too much extra work, but our secretaries, Zahn Gowar and Leanne Reddy, inevitably had to take on some extra work. John Ostrowick provided valuable assistance with design of our web pages and call for papers poster. Carol Kernick, who handles our finances and membership records, did a fine job of keeping up with the demands of the conference.

Finally, I would like to thank our sponsors, whose contribution made this conference been possible:

- PricewaterhouseCoopers – sponsored generous prizes and the conference banquet
- National Research Foundation (NRF) – provided financial support
- University of the Witwatersrand – provided financial support
- Programme for Highly Dependable Systems, University of the Witwatersrand – provided financial support
- Standard Bank – provided financial support
Editorial

- Apple Computer - provided equipment for the conference
- Qualica - provided technical support including helping with the conference web site

Web Site
For more information about SAICSIT, including a pointer to the conference site, see <http://www.saicsit.org.za>.

Referees
- Department of Computer Science, University of Pretoria
  - Derrick Kourie
  - Bruce Watson
  - Vali Lalioti
  - Andries Engelbrecht
  - Ivan Mphahlele
- German National Research Center for Information Technology - GMD
  - Gernot Goebbels
- School of Math Stat and IT, Natal University Pietermaritzburg
  - Peter Warren
- Graduate School of Business, University of Cape Town
  - Kurt April
- CSIR
  - James Jardine
- Department of Computer Science, Rhodes University
  - Shaun Bangay
  - Peter Clayton
  - John Ebden
  - Richard Foss
  - George Wells
  - Peter Wentworth
- Information Technology Division, Rhodes University
  - Caro Watkins
- Department of Information Systems, Rhodes University
  - Brenda Mallinson
  - Dave Sewry
- Department of Informatics, University of Pretoria
  - Carina de Villiers
  - Herna Viktor
  - Niek du Plooy
  - Elsje van Rooyen
  - Machdel Matthee
  - Alan Abrahams
  - Jackie Phahlamohlaka
- School of Information Technology & Engineering, University of Ottawa
  - Stan Szpakowicz
  - Dwight Makaroff
- Department of Information Systems, Victoria University
  - Glenn Lowry
  - Peter Shackleton
  - Tas Adam
  - Alastair Wallace
  - Stephen Burgess
  - Dave Burgess
  - Julie Fisher
  - Jerzy Lepa
  - Geoff Sandy
  - Rod Turner
  - Alastair Wallace
  - Andrew Wenn
- Peninsula School of Computing and Information Technology, Monash University
  - Ainslie Ellis
- School of Information Management and Systems, Monash University
  - Angela Carbone
- School of Computer Science & Software Engineering, Monash University
  - Dianne Hagan
  - Judy Sheard
- School of Management Information Systems, Deakin University
  - John Lamp
  - David Mackay
  - Philip Joyce

SACJ/SART, No 24, 1999
Editorial

- Chris Brew
- Alexander Holt

- Department of Computer Science and Engineering, Helsinki University of Technology
  - Tommi Junttila
  - Nisse Husberg

- Technical University of Budapest, Department of Measurement and Information Systems
  - Andras Pataricza

- NASA-Goddard Space Flight Center
  - Nigel Ziyad

- Department of Computer Science & Information Systems, University of Port Elizabeth
  - Janet Wesson
  - Leon Nicholls

- Computer Technology Department, Indiana University Purdue University Indianapolis
  - Tim Price

- Department of Electrical Engineering and Computer Science, The University of Michigan
  - Trevor Mudge

- Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
  - David Forsyth

- School of Computer Science, University of Birmingham
  - Mark Ryan

- Faculty of Mathematics, Computer Science, Physics & Astronomy, University of Amsterdam
  - Carlos Areces

- Department of Computer Science, Universidade do Vale do Rio dos Sinos - UNISINOS Rio Grande do Sul
  - Marcelo Walter

- School of Information Technology & Mathematical Sciences, University of Ballarat
  - Binh Pham
A Java Client/Server System for Accessing Arbitrary CANopen Fieldbus Devices via the Internet

Dieter Bühler\textsuperscript{a}, Gerd Nusser\textsuperscript{a}, Gerhard Gruhler\textsuperscript{b} and Wolfgang Küchlin\textsuperscript{a}

\textsuperscript{a}Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 72076 Tübingen, Germany; \{buehler, nusser, kuechlin\}@informatik.uni-tuebingen.de

\textsuperscript{b}Institut für Angewandte Forschung in der Automatisierung, Fachhochschule für Technik und Wirtschaft, 72762 Reutlingen, Germany; gerhard.gruhler@fh-reutlingen.de

Abstract

This paper describes the Java CAN API, a portable and extensible application interface to arbitrary CANopen fieldbus devices. This API forms the basis for the implementation of JRCC (Java Remote CAN Control), a tool which can be used to retrieve and modify the whole set of status parameters of arbitrary CANopen devices via the Internet, providing the basic facilities for remote maintenance, remote data acquisition and remote control of CANopen nodes. In an educational context this opens up the opportunity to demonstrate and to experience the behavior of various CANopen modules with respect to specific status modifications, independent of the local availability of those devices.

Since we choose a Java client/server software architecture using TCP/IP sockets, the JRCC client can easily be embedded in an ordinary HTML web page and be executed by any web browser featuring a Java 1.1 compliant virtual machine. The abandonment of a middleware layer results in an extremely small and portable system with high availability and little system requirements, making this approach a candidate for both, a low cost embedded system solution for remote access to CANopen devices and a fieldbus teaching tool for the potentially heterogenous client system environment of a distributed collaborative learning community connected via Internet.

Keywords: network computing, CANopen device, java api, fieldbus, internet, client/server, distance education, remote maintenance

Computing Review Categories: C.2, D.2.11, J.2, K.3

1 Introduction and Related Work

CANopen [2] is a layer 7 network protocol\textsuperscript{1} for CAN (Controller Area Network) [3] communications, primarily used in industrial automation applications where devices such as distributed I/O modules, drives etc. communicate over a CAN fieldbus. Almost all communication with such a CANopen device is done via the so-called Device Profile, a two dimensional array of parameters, reflecting the current status of a device at a given time. The specific entries of the Device Profile are addressed by index and subindex and can be read to retrieve a specific parameter and written to change the current status of a device (e.g. to activate an output channel of an I/O module). CANopen is an open standard, developed within the CAN in Automation (CiA) [1] international users and manufacturers group formed by more than 300 companies worldwide.

The underlying CAN is a real-time, multi-master, serial bus system with outstanding transmission reliability, specified up to layer 2 of the ISO/OSI reference model.

The Java CAN API and the JRCC tool presented in this paper allow client/server access to CANopen device profiles, providing remote access to arbitrary CANopen modules connected to CAN. This enables any user with a connection to the Internet to take a detailed look into the devices connected to the system and to modify the current status parameters. Besides comprehensive possibilities for teaching and demonstrating CANopen fieldbus concepts over the Internet, this system provides the basic facilities for remote maintenance, remote data acquisition and remote control of CANopen devices.

In order to keep the system small and open, we used free and standardized software whenever possible. The abandonment of a middleware software layer, for example Microsoft's DCOM/OPC technology, the Common Object Request Broker Architecture (CORBA) or Java RMI, benefits in very low system requirements. The technology for remotely accessing CANopen devices via the Internet presented in this paper, can easily be used in a small embedded system solution for remote data acquisition and remote maintenance. The requirements for the target server system consist only of an embedded Java virtual machine, a TCP/IP stack and a CAN interface.

Other systems described in [6] or [5, 4] using DCOM/OPC or CORBA respectively, depend on the availability of specific proprietary operating system services like access to the Microsoft Windows Registry, a name service, an Object Request Broker (ORB), or other middleware services. Furthermore, the system presented in this paper establishes a more general view on CANopen devices.

\textsuperscript{1}According to the ISO/OSI reference model

SART / SACJ, No 24, 1999
hiding the device type specific particularities by providing access to the general concept of the CANopen Device Profile.

The Java CAN API developed for the JRCC tool is designed for maximum portability across hardware and operating system boundaries and forms a good basis for both, a low cost embedded system solution for remote access to CANopen devices and a fieldbus teaching tool for the potentially very heterogenous client environment of a distributed collaborative learning community connected via Internet.

The major aim of this paper is to describe the developed software infrastructure to provide worldwide access to arbitrary CANopen devices, with its focus on portability, reusability and little system requirements on the client and the server side.

The rest of the paper is structured as follows: Section 2 describes the JRCC system architecture, discussing hardware issues, the C++ CAN message management, the Java CAN API and the client/server communication principles. The succeeding section describes the JRCC tool from the user’s perspective on the example of a fieldbus teaching tool for distance education. The paper concludes with a summary and outlook.

2 System Architecture

2.1 Overview

![Figure 1: Overview of the system architecture](image)

The JRCC hardware setup (see figure 1) consists of a standard PC (Intel Pentium II, WinNT) connected to CAN via a CAN Interface Card (Vector Informatik GmbH CanCardX). Presently, two CANopen I/O modules from Beckhoff and Selectron are connected to the bus, but they could be replaced by any other CANopen modules, without having to reconfigure the system.

In order to serve the Java client class files and to provide worldwide access to the installed CANopen devices, a HTTP-Server (Apache) and our JRCC server are running on the server machine. If the client class files are installed at the client computer as a Java application, the HTTP server becomes obsolete. The JRCC clients communicate with the JRCC Server directly over Java-TCP/IP sockets which guarantees high performance and almost unrestricted browser support.

![Figure 2: Accessing a specific status parameter via the Internet](image)

During the JRCC development the JRCC client and server, the Java CAN API, formed by the CanProtocol and JCan package, and the JCan.DLL were implemented (see figure 2). The Vcand32.DLL ships with the CANCardX and provides C functions for accessing the CAN controller. The following subsections of this paper describe the developed software modules in a bottom up manner, starting with the JCan.DLL.

2.2 The JCan.DLL

The Vcand32.DLL supports primitive access to the CAN bus, providing only simple transmit and receive functions and no message management. The JCan dynamic link library handles the following two main tasks: first, it encapsulates the CAN interface card resp. the interface to the Vcand32.DLL, providing an abstract, thread safe view of the net traffic on the CAN bus, and second, it is part of the implementation of the Java CAN API, which references the methods of the JCan.DLL via the Java Native Interface (JNI).

The JCan.DLL autonomously manages all CAN bus traffic caused by connected CANopen modules. This is done by the so-called message manager thread (see figure 3) which collects all generated CAN messages on the bus in an infinite loop. Since in most cases CANopen message identifiers incorporate information about the recipient and the meaning of a CANopen message, the messages are managed with respect to their actual identifier. The JCan.DLL can be configured dynamically to hold specific message queues for each identifier. When a new message appears on the bus, the message manager retrieves the ID of the message and checks if some Java instance has subscribed for messages with this specific identifier. If so, the message is stored in a special STL\(^2\) queue data structure reserved for messages with this identifier (see figure 3). If no one is interested in this message, it is ignored.

\(^2\)Standard Template Library
polling techniques to read the incoming messages. Since 
unsubscribed messages are ignored, it is possible to mask 
the CAN traffic that is not interesting, which may be the 
main part, at a fine granularity. The decision to implement 
this message management in C++ results in a high 
performance filtering of incoming CAN messages, keeping 
the Java side of the JRCC system from the time critical 
aspects of the CAN message management.

### 2.3 The Java CAN API

The Java CAN API is formed by the JCan and the CanPro­

tocol package. The JCan package provides Java classes 
and interfaces to access a CANCardX, CANPari or CAN­

AC2 CAN interface card of Vector Informatik GmbH via 
the JNI. The CanPort class for example provides the na­
tive method declarations for the JCan.DLL functions and 
father methods for the CAN event handling (see table 2) 
which was implemented according to the observer pattern. 
To access the CAN bus, a Java instance needs a reference to 
a CanPort object providing the interface to the JCan.DLL. 
If asynchronous notification shall be used, the Java in­
stance needs to implement the CanPortEventListener in­
terface by overriding the canEvent method and sign itself 
as an event listener at the CanPort object using the addE­
ventListener method (see table 2). If the message manager 
of the JCan.DLL encounters a subscribed message with the 
novation flag set, all signed CAN event listeners are noti­
fied by invoking their version of the canEvent method. The 
CanPortEvent argument passed to the canEvent method 
contains information about the event type and the event 
source so the notified java instances can react in an appro­
riate way, e.g. call the receive or peek function of the 
JCan.DLL.

Both, classes for generic CAN layer 2 messages and 
CANopen layer 7 messages are aggregated to the CanPro­
tocol package. The CanMsg class encapsulates a generic CAN message 
with an identifier, 8 bytes of data and the data length code (dlc). The send method (see table 3) writes a CanMsg in­
tece to the CAN bus using the specified CanPort refer­
ence.

CANopen defines a Service Data Object (SDO), which 
can be used to read or write an individual device profile entry 
by index and subindex. The class SDO provides these 

---

3The receive, peek and transmit functions of the JCan.DLL are im­
plemented as critical sections with mutual exclusion to prevent illegal 
interactions of parallel function invocations.
features by extending the CanMsg class. Network management functionality e.g. setting a device from the CANopen pre-operational state to the operational state, is made available by the NMT class.

### 2.4 The Client/Server Communication

The JRCC client and server communicate via TCP/IP sockets, the lowest common denominator with regard to support from current web browsers. This is particularly advantageous in an distributed educational context, with varying operating systems, hardware architectures and web browsers used by students at different Universities.

On the server side only a small Java Virtual Machine, a TCP/IP protocol stack and a CAN interface are mandatory. Even a small embedded system may meet these requirements, allowing the construction of a low priced mobile hardware interface to arbitrary CANopen fieldbus setups. Since a growing part of the automobile industry installs a CAN fieldbus in their products, this mobile CAN interface may be used for the remote diagnostics and even remote maintenance of cars.

The client requests to the CANopen modules have to be synchronized, to avoid confusing or illegal interference of different clients. To achieve this, the server only grants exclusive access by blocking incoming requests of other clients. The server takes care that every requesting client gets access to the devices for at most a specific time slice. When the time slice is over, a time out occurs and the client has to reconnect the server. The server holds all client connection requests in a first in first out queue to ensure a fair server arbitration.

### 3 Using the JRCC Tool in Practice

The JRCC client can be started via the web page located at [http://robo16.fh-reutlingen.de/english/demol-JRCC_client.html](http://robo16.fh-reutlingen.de/english/demol-JRCC_client.html). The menu CAN-Nodes lists all CANopen modules currently connected to the JRCC server. To communicate with one of these devices, the corresponding entry in this menu has to be selected. The Electronic Data Sheet (EDS), which describes the device profile of the selected CANopen module, is then automatically downloaded from the JRCC server, parsed and displayed in a tree view component on the left side of the JRCC client window. The user can expand and collapse the entries in the device profile tree to gather information (e.g. ParameterType, DataType,AccessType, etc) about individual parameters. If a specific device profile entry is selected by double clicking, the corresponding values are transferred into the Index and Subindex fields at the right half of the window and the user can now read or write this parameter via the Internet by pressing the Read or Write button.

The NMT-Services menu provides the CANopen network management services to control the network status of individual nodes. In order to further illustrate the possibilities of this system, we connected the output lines of one of the I/O modules with its input lines and a "Feldbus" display (see figure 4). The JRCC tool can now be used as an interface to this display. When some of the output lines are switched to high, the corresponding letters of the display are lit and be-

---

**Table 2:** The CAN Event Handling API (extract).

<table>
<thead>
<tr>
<th>class/interface</th>
<th>Function Name</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>class CanPort</td>
<td>addEventListener</td>
<td>(CanPortEventListener lsnr)</td>
</tr>
<tr>
<td>class Interface</td>
<td>removeEventListener</td>
<td>(CanPortEventListener lsnr)</td>
</tr>
<tr>
<td>class CanPortEvent</td>
<td>canEvent</td>
<td>(CanPortEvent canEvent)</td>
</tr>
<tr>
<td>class CanPortEvent</td>
<td>getEventSrc</td>
<td>()</td>
</tr>
<tr>
<td>class CanPortEvent</td>
<td>getEventType</td>
<td>()</td>
</tr>
</tbody>
</table>

**Table 3:** The CanProtocol Package (extract).

<table>
<thead>
<tr>
<th>class/interface</th>
<th>Function Name</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>class CanMsg</td>
<td>setId</td>
<td>(int id)</td>
</tr>
<tr>
<td>class SDO</td>
<td>setData</td>
<td>(byte[] data)</td>
</tr>
<tr>
<td>class NMT</td>
<td>send</td>
<td>(CanPort port, int dst)</td>
</tr>
<tr>
<td>class NMT</td>
<td>write</td>
<td>(int index, int subindex, long value, ...)</td>
</tr>
<tr>
<td></td>
<td>read</td>
<td>(int index, int subindex)</td>
</tr>
<tr>
<td></td>
<td>startRemoteNode</td>
<td>(CanPort port, int moduleId)</td>
</tr>
<tr>
<td></td>
<td>stopRemoteNode</td>
<td>(CanPort port, int moduleId)</td>
</tr>
</tbody>
</table>

---

**Figure 4:** The I/O Module Circuit of Node 0x10

---

4We developed a special tree view component without using any Swing or JFC classes to stay compatible with current web browsers.
forced to download any browser plug-ins, additional CORBA classes or install auxiliary software on the client system. The server requirements can even be met by a low cost embedded system, featuring a small embedded Java VM, a TCP/IP stack and a CAN interface. For example, a truck driver experiencing a breakdown in the middle of nowhere, could use such a system, a mobile phone and the Internet to enable remote access to the CAN devices built into the truck, allowing a distant engineer to check his vehicle for malfunctions.

References


Acknowledgements

This paper is based upon work partially funded by the Ministerium für Wissenschaft, Forschung und Kunst.
Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal’s readers will be considered for publication under sections marked as Communications of Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines:

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
  - the title (as brief as possible)
  - the author’s initials and surname
  - the author’s affiliation and address
  - an abstract of less than 200 words
  - an appropriate keyword list
  - a list of relevant Computing Review Categories
  - Tables and figures should be numbered and titled.
  - References should be listed at the end of the text in alphabetic order of the (first) author’s surname, and should be cited in the text according to the Harvard method.

Manuscripts accepted for publication should comply with guidelines as set out on the SACJ web page, http://www.cs.up.ac.za/sacj, which gives a number of examples.

SACJ is produced using the \LaTeX document preparation system, in particular \LaTeXe. Previous versions were produced using a style file for a much older version of \LaTeX, which is no longer supported.

Please see the web site for further information on how to produce manuscripts which have been accepted for publication.

Authors of accepted publications will be required to sign a copyright transfer form.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30.00 per final page for contributions which require no further attention. The maximum is R120.00, prices inclusive of VAT.

These charges may be waived upon request of the author and the discretion of the editor.

Proofs

Proofs of accepted papers may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words. Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book Reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000.00 per full page per issue and R500.00 per half page per issue will be considered. These charges exclude specialised production costs, which will be borne by the advertiser. Enquiries should be directed to the editor.