Open System Real Time Architecture and Software Design for Robot Control

B.Bona  M.Indri  N.Smaldone
Politecnico di Torino
LabRob
Dip. di Automatica e Informatica

Summary

Introduction – Motivations and goals
Plant description – The IMI robot and the problems of the original environment
The “OpenDSP” configuration – Some words about the architecture
Our customization – Why we like it …
Some results – A simple example
Conclusions – …
Motivations and Goals

Educational purposes

Practical experiments for the “Industrial Robotics” course

Research purposes

RAMSETE project (MURST)

To obtain an environment

Reliable for “hard real-time”
Matlab compliant
Fully configurable

The IMI Robot and Its Original Environment

• Host PC: C-written SW management
• IMI rack: hardware interface, driver units and power supply
Problems

Software: normal behavior

![Diagram showing control and acquisition intervals](image)

Software: intrusive for intensive data acquisition sessions

Unpredictable duration

Software: data analysis in Matlab and plant management in DOS need for a unique environment

Hardware: interface malfunctions in undocumented hardware
The New Configuration

- Host PC: Matlab based SW management
- OpenDSP rack: DSP board, IO board, hardware interface (FM)

The OpenDSP Hardware

- MB
- GPIOB
- DAC/ADC
- PLDs
- Analog and digital comps
- Field Modules (stackthrough)
- D&A Field bus
- Real World
The OpenDSP Software

<table>
<thead>
<tr>
<th>System</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Matlab toolbox, OpenDSPscope</td>
</tr>
<tr>
<td></td>
<td>Graphical User Interfaces (Matlab)</td>
</tr>
<tr>
<td>DSP</td>
<td>OpenDSP RTOS, Virtualization, Standard macros</td>
</tr>
<tr>
<td></td>
<td>C-code template, custom virtualization</td>
</tr>
<tr>
<td>PLD</td>
<td>PLD communications</td>
</tr>
<tr>
<td></td>
<td>Any logic needed (FSMs,...)</td>
</tr>
</tbody>
</table>

Why we like it …

1. Matlab environment is a powerful and educational tool for the final user
Why we like it …

2. DSP code developed in C language which is well known in the automation field
   • Starting from a rigid framework C-template
   • Filling some specific section with custom code

The user can obtain a reliable control algorithm without worrying about real-time

Why we like it …

3. Hard real-time is granted by dedicated HW and tailored RTOS core

My C-code + RTOS = DSP executable
Why we like it …

4. PLD and the stackthrough structure allow modularity and save lots of HW (e.g., passing from 2 to n axes)

Some results

1. Filling the template with the code of a PD algorithm
2. Compiling/linking it with the RTOS files
3. Downloading the executable into the DSP and starting it
4. Using the OpenDSPscope tool to acquire and save some data to analyze them with Matlab
Conclusions

• The new HW/SW architecture allows to see in some details educational aspects of robotics: trajectory planning, C-code implementation issue of the control algorithms, data acquisition and visualization

• It is now possible to deal directly with the functional logic of the drivers and decide how to control them and the supervision logic

• The modularity of the system makes it possible to introduce additional sensors/actuators (an end-effector or force/torque sensor) in the plant, with few focused changes in the OpenDSP structure