Real-Time & Embedded Systems
Past, Present, and Future

Dr. Doug Locke
Locke Consulting, LLC
www.doug-locke.com
• What is a real-time System?
• What is an embedded System?
• Where have we come from?
• What have we achieved?
• Where are we going?
What is a Real-Time System?

1. Correctness is a function of time?
2. Must respond to external device in less than X microseconds?
3. Real-fast?
4. Missed deadline means catastrophic result?
5. System should respond “instantaneously”
6. All of the above?
7. None of the above?
What is an Embedded System?

• Small device, like a cell phone?
• Small processor installed in some other device, like a car?
• Software that controls a consumer device?
• Must have real-time response?

My favorite:
• Any system where the user doesn’t want to know that it includes a processor
Examples of Real-Time / Embedded Systems

- Car engine
- Cell phone
- Set-top box
- Car navigation
- Industrial control
- Telecom switch
- Global Positioning System
- Air Traffic Management
- Satellite flight manager
- Satellite Ground Control
- TV receiver
- Flight control
- Electric shaver
- Toaster
Outline

• What is a real-time System?
• What is an embedded System?
• Where have we come from?
• What have we achieved?
• Where are we going?
Model 029 Keypunch
LAMPS Radar Test
LAMPS Mark III
Ticonderoga Class (USS San Jacinto, CG-56)
NASA Space Shuttle
NASA Mariner 10
## Embedded Computer Capacities

<table>
<thead>
<tr>
<th>Year</th>
<th>Memory Size</th>
<th>CPU Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>8-32KB</td>
<td>128 KIPS</td>
</tr>
<tr>
<td>1975</td>
<td>16-64KB</td>
<td>1.2 MIPS</td>
</tr>
<tr>
<td>1980</td>
<td>64-128KB</td>
<td>5 MIPS</td>
</tr>
<tr>
<td>1985</td>
<td>128-1MB</td>
<td>20 MIPS</td>
</tr>
<tr>
<td>1990</td>
<td>1-4MB</td>
<td>50 MIPS</td>
</tr>
<tr>
<td>1995</td>
<td>2-32MB</td>
<td>150 MIPS</td>
</tr>
<tr>
<td>2000</td>
<td>4-128MB</td>
<td>800 MIPS</td>
</tr>
</tbody>
</table>

- Increasing variability throughout this time
Size of Large Embedded Software

• How large is “large”:
  – 1970  10K SLOC
  – 1975  150K SLOC
  – 1980  1M SLOC
  – 1985  2M SLOC
  – 1990  4M SLOC
  – 1995  4M SLOC (increasing component use)
  – 2000  4M SLOC (increasing component use)

• Increasing variability throughout this time
• Shortest Time Constraints Reliably Achievable:
  – 1970 50 milliseconds
  – 1975 1 millisecond
  – 1980 500 microseconds
  – 1985 100 microseconds
  – 1990 50 microseconds
  – 1995 10 microseconds
  – 2000 5 microseconds
Embedded Systems Proliferation

• Applications:
  – 1970 Military / Aerospace
  – 1975 Factory Automation / Telecom
  – 1980 Consumer Electronics
  – 1985 Wireless Telecom / Automotive
  – 1990 Games / Toys / Entertainment / Internet
  – 1995 Appliances
  – 2000 RFID
• What is a real-time System?
• What is an embedded System?
• Where have we come from?
• What have we achieved?
• Where are we going?
What Was (Is Still) the Biggest Challenge?

- Exponentially increasing capacity
- Exponentially increasing software size and complexity
- Linearly increasing pool of developers
- Fixed or decreasing budgets
- The big problem – how to build exponentially more systems, and exponentially more complex systems with linearly increasing labor.
Strange game – the only way to win is not to play!  
- Joshua in the movie *Wargames*

Only way to produce complex software:

- Avoid writing, testing, documenting code
  - Use Commercial Off-The Shelf (COTS)
  - E.g., RTOS, CORBA, Database, Web-based, Automated tools, reuse existing code

- Unintended consequence
  - Performance problems
Top Three Problems:

• Managing software engineering organizations
• Ensuring development of a performance-relevant architecture
• Finding suitable tools (language, COTS, analysis, simulation)
A Taxonomy of Real-Time Architectures

- The vast majority of existing real-time applications use one of four (overlapping) architectural types:
  1. Timeline (a.k.a. cyclic executive or frame-based)
  2. Event-driven (with both periodic and aperiodic activities)
  3. Pipeline
  4. Client-Server
Timeline or Cyclic Executive

I/O

I/O

I/O

I/O

Cyclic Executive

Procedure 1

Procedure 2

Procedure 3

Procedure 4

Timer (e.g., 40 Hz, 25ms. period)

Procedure 1 alone

Procedure 1 and 2

Procedure 1 and 3

Procedure 1 and 4

Procedure 1

40 Hz

Procedure 2

20 Hz

Procedure 3

5 Hz

Procedure 4

1 Hz
Tasks generally priority scheduled
Tasks usually ad-hoc scheduled
Tasks usually ad-hoc scheduled
None of the architectures described are free of problems

- Timeline is extremely expensive to integrate and maintain
- Event-driven model is predictable for relatively static designs
- Pipelines commonly result in non-preemptive delays (i.e., priority or policy inversion), few tools for predictable response
- Client Server infrastructures perform similarly to pipelines except concurrency can be much more limited.

The Bottom Line: Architecture decisions have a major effect on

- Performance
- Safety
- Fault Tolerance
- Life Cycle Cost
What has our community produced?

Quite a lot - a few examples:
- Rate (Deadline) Monotonic Scheduling
- Utility (or Value) Function Scheduling
- Many other scheduling paradigms (e.g., EDF)
- Imprecise Computations
- Fault Tolerant Computing (e.g., Simplex)
- Real-Time Databases

We have had considerable influence
- POSIX
- Real-Time CORBA
- Real-Time Linux
- Ada 95, Real-Time Java

But much of our contribution isn’t widely known
Outline

• What is a real-time System?
• What is an embedded System?
• Where have we come from?
• What have we achieved?
• Where are we going?
Where Are We Going?

- Resources are still limited
  - Therefore they will still need careful management
  - Scheduling still matters
- “Non-functional” requirements are now the primary focus of most designs
  - Real-time response
  - Fault tolerance
  - Availability
  - Quality of Service
  - Power Management
  - Security
  - Cost (people cost + resource cost)
- This is where we continue to make a difference
Doug Locke
Locke Consulting, LLC
www.doug-locke.com