The Embedded Machine
Predictable, Portable Real-Time Code

Thomas A. Henzinger, Christoph M. Kirsch
UC Berkeley

www.eecs.berkeley.edu/~cm
Embedded Software

Environment

Environment Processes

Software Processes

Software
Environment vs. Platform Time

Environment

Platform Time

Software
The Art of Embedded Programming

Environment

Embedded Programming

Software
The Embedded Machine

Environment Ports

Embedded Machine

reads

Driver Ports

actuates

reads

writes

updates

Task Ports

reads

reads actuates

updates

reads writes
The Task Model

Environment

sense

start

Software

actuate

end
...but Atomic

$f(1) = \circ$

Environment

Software
The Driver Model
Non-preemptable, Synchronous Environment

Software

$f(\text{Environment}) = 1$

$f(\text{Software}) = 2$
Syntax

Environment

Software
Environment-triggered Code

Environment

Software
Task-triggered Code

Environment

Software
The Trigger Model

Environment Ports

Embedded Machine

Driver Ports

Task Ports
A Trigger $g$

$g : c' \neq c$
An Embedded Machine Program

Environment

Software

b:
- \textbf{call}(a)
- \textbf{call}(s)
- \textbf{schedule}(t)
- \textbf{future}(g,b)

\begin{align*}
\text{future}(g,b) & : \text{future}\left(\text{future}(g,b),t\right) \\
\text{call}(s) & : \text{call}\left(\text{call}(s),b\right) \\
\text{schedule}(t) & : \text{schedule}\left(\text{schedule}(t),1\right) \\
\text{call}(a) & : \text{call}\left(\text{call}(a),2\right)
\end{align*}
Synchronous vs. Scheduled Computation

Environment

Software

b:
- call(a)
- call(s)
- schedule(t)
- future(g,b)
Synchronous vs. Scheduled Computation

- **Synchronous computation**
  - Kernel context
  - Trigger related interrupts disabled

- **Scheduled computation**
  - User context
The Zürich Helicopter
Helicopter Control Software

Clock

\[ g : c' = c + 5 \]

Sensor

Control 10

Actuator

Navigation 5
sensor gps_type GPS uses c_gps_device ;
actuator servo_type Servo := c_servo_init
uses c_servo_device ;
output
ctr_type CtrOutput := c_ctr_init ;
nav_type NavOutput := c_nav_init ;
driver sensing (GPS) output (gps_type gps)
{ c_gps_pre_processing ( GPS, gps ) }
task Navigation (gps_type gps) output (NavOutput)
{ c_matlab_navigation_code ( gps, NavOutput ) }
...
mode Flight ( ) period 10ms
{
  actfreq 1 do Servo ( actuating ) ;
  taskfreq 1 do Control ( input ) ;
  taskfreq 2 do Navigation ( sensing ) ;
}
...
Environment Timeline

- 0ms
- 5ms
- 10ms

Control

Navigation

Block of synchronous code (nonpreemptable)

Scheduled tasks (preemptable)
b1: call(a_actuating)
call(s_ensing)
call(i_input)
schedule(Control [10])
schedule(Navigation [5])
future(g,b2)
b2: call(s_ensing)
schedule(Navigation[5])
future(g,b1)
Portability

- Programming in terms of environment time yields platform-independent code
Predictability

- Programming in terms of environment time yields deterministic code
Time Safety

Environment

Software
Runtime Exceptions I

Environment

Software

\text{call}(\text{a}_\text{ctuating})
Runtime Exceptions II

Environment

Software

\text{call}(s_{\text{ensing}})
Runtime Exceptions III

1

schedule(t)

Environment

2

Software
An Exception Handler $e$

Environment

Software

.schedule(t,e)

Call graph:

- $b$: call($a$)
- call($s$)
- schedule($t$)
- future($g,b$)

- $e$: call($a'$)

- $a$
- $a'$

Diagram:

- Node 1: schedule($t,e$)
- Node 2: call($a'$)
Trigger $g$: Input-, Environment-Triggered

g call(s) schedule(t) future($g,b$)
Input-deterministic If Time Safe

Environment

Software
Environment-deterministic If Environment-triggered
How to Loose Determinism: Task Synchronization
How to Loose Determinism: Termination

Environment

Software

\texttt{terminate}(t)
Time Liveness: Infinite Traces
Dynamic Linking

e: call(s)
schedule(t)
future(g, b)

E Code

Functionality Code

g

t

E Machine
The Berkeley Helicopter
Platform Timeline: Time-triggered Communication
Code Generation for HeliNav

\[ b2: \quad \text{call}(s\_ensing) \]
\[ \text{schedule} (\text{Navigation}[2]) \]
\[ \text{schedule} (\text{Connection}[\{7,10\}]) \]
\[ \text{future}(g, b1) \]
Instructions

Synchronous Driver:
\texttt{call}(d)

Scheduled Task:
\texttt{schedule}(t)

Triggering:
\texttt{future}(g, b)

\( g : c' \neq c \)