IEC 61499 Applications in Embedded Systems

Partha S Roop
Precision Timed Systems Group
http://www.ece.auckland.ac.nz/~pretzel/
University of Auckland, New Zealand

Outline

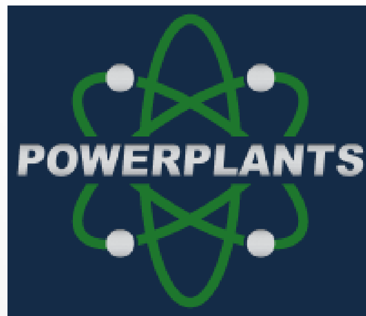
- Specific requirements of Embedded Systems
- TimeMe – a tool-kit for safety-critical embedded systems using IEC 61499
- Case Study 1: Fertigators
- Case Study 2: Animal Weighing Scales
- Extensions to IEC 61499

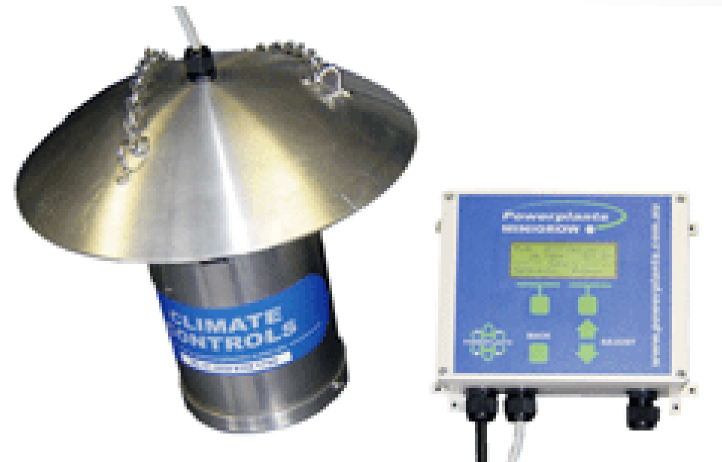
Embedded Systems

- Unique requirements of Embedded systems
  - Smaller code footprint.
  - No OS or runtime.
  - Custom communication protocols.
- Requirement for code generation
  - C Code generation.
  - Re-use of existing code base.
- Verification and Validation
  - Simulation prior to deployment
  - Formal verification
- Hard Real-Time Systems
  - Static timing analysis.
  - Distributed deployment without any middleware.

Powerplants Ltd.

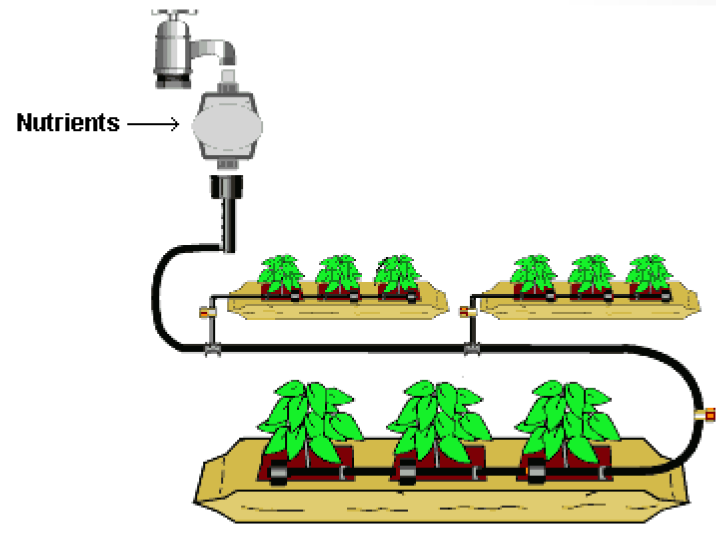
- Provide greenhouse growers throughout Australia with the latest horticultural technology and supplies.
- Embedded Systems:
    - Set point control (non-PID controlled)
    - C code





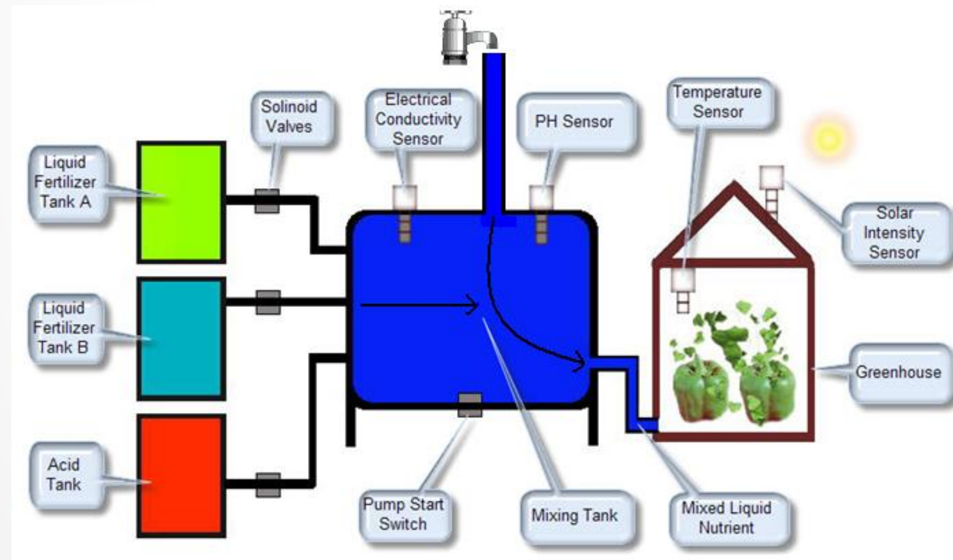
Fertigator

- Modern greenhouses are hydroponic.
- Fertigation is the process of injecting liquid nutrients into water.
- Nutrients and water are mixed inside a tank.
- Instrumentation based control for the optimal release of nutrients.



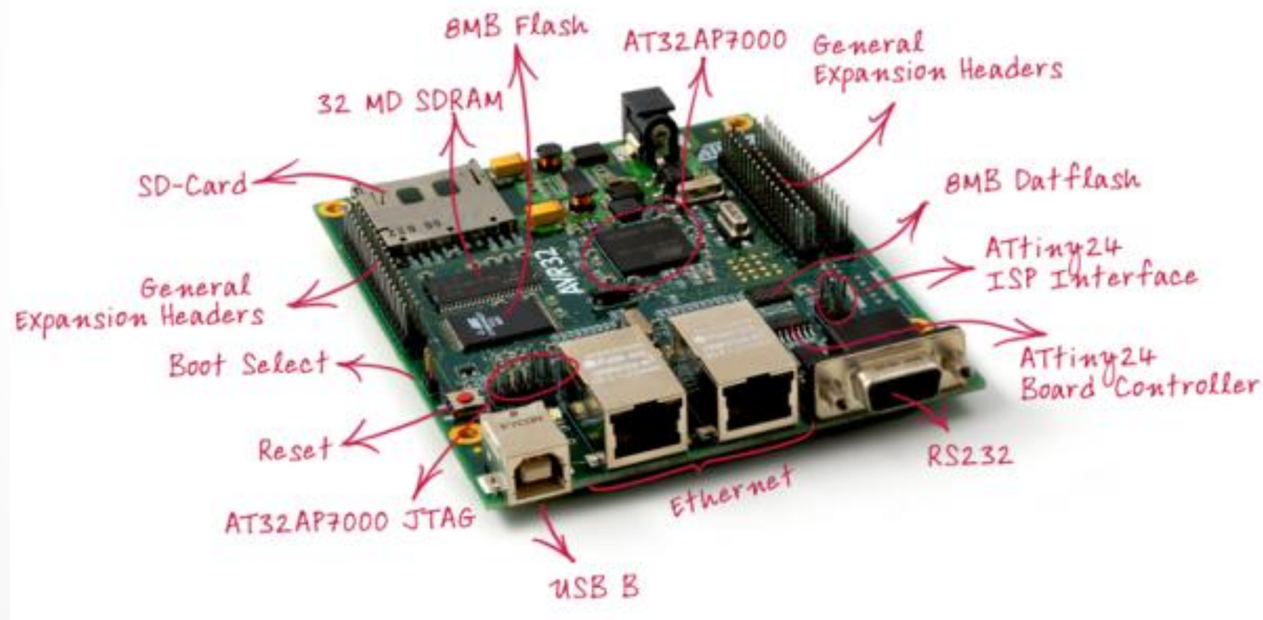
Fertigation Control

- Measure electrical conductivity, temperature, solar intensity and PH level.
- Control pumps and valves to reach desired values.



The Hardware

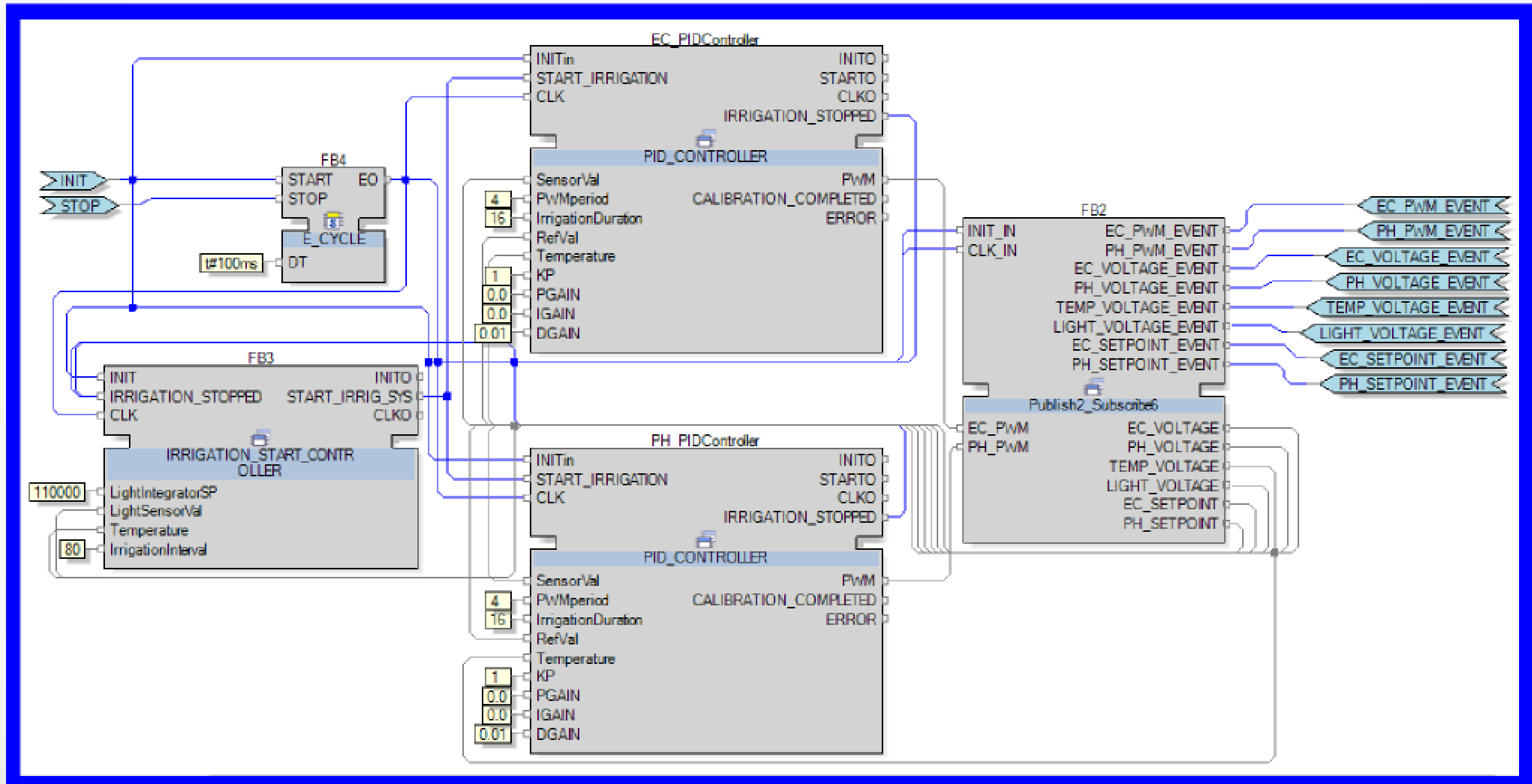
- The AVR NGW100 board, runs embedded Linux.
- On-board WAN setup for Ethernet communication.



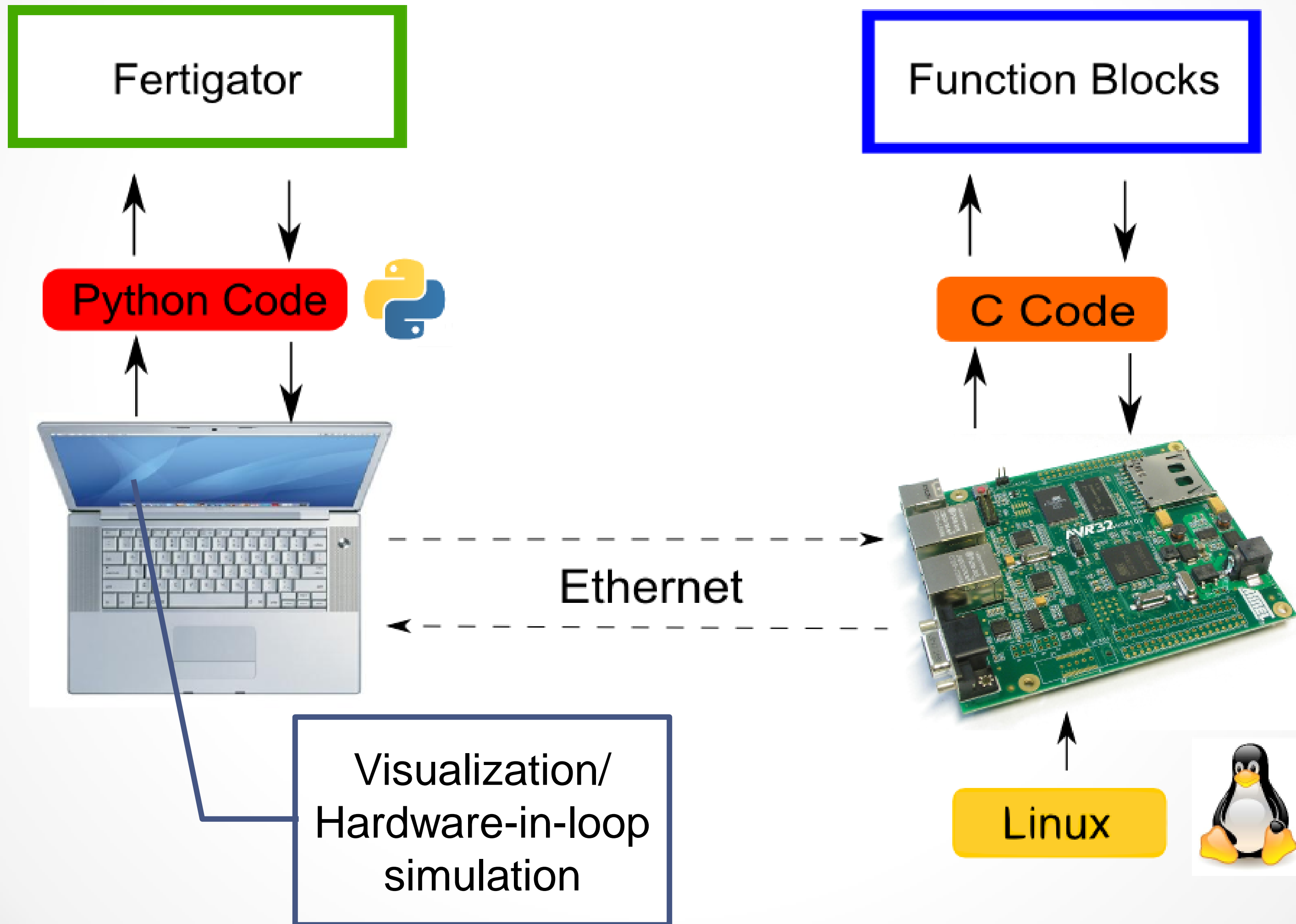
IEC61499 based Solution

Fertigator

Function Blocks



IEC61499 based Solution

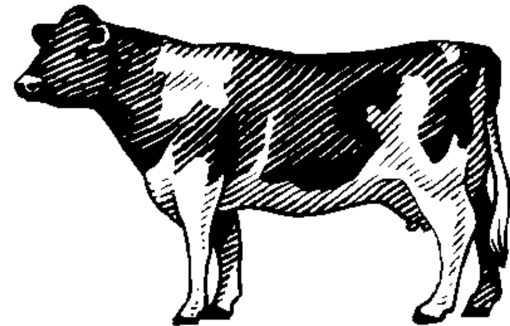


Agri-tech example

- A large agri-tech products company such as livestock weighing systems and milk meters.
- Client interested in the application of MDE methodology.
- We are jointly exploring the feasiblity of using IEC61499 in the embedded system design life-cycle.
- Embedded Systems
  - ARM processor
  - UML and C Code

Weighing Scale

- Secure connection to Bluetooth EID readers.
- Animal weight history and gain predictions.
- Record keeping for weight , treatment and custom fields of more than 50,000 animals.
- Statistics and on-screen display.



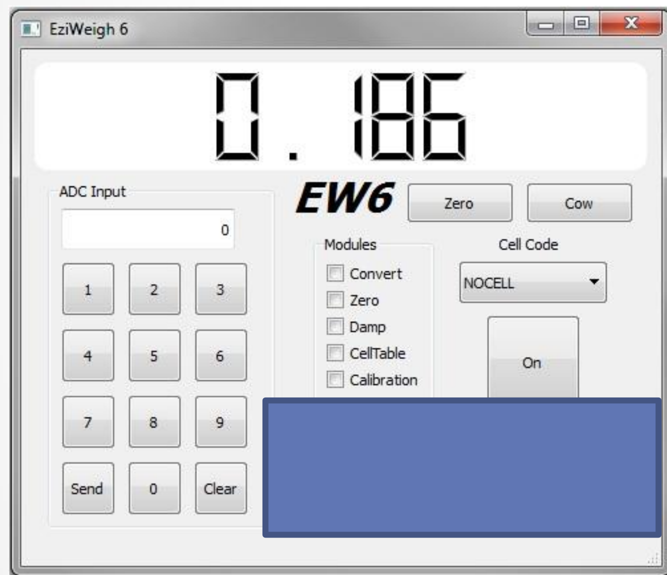
Legacy System Issues

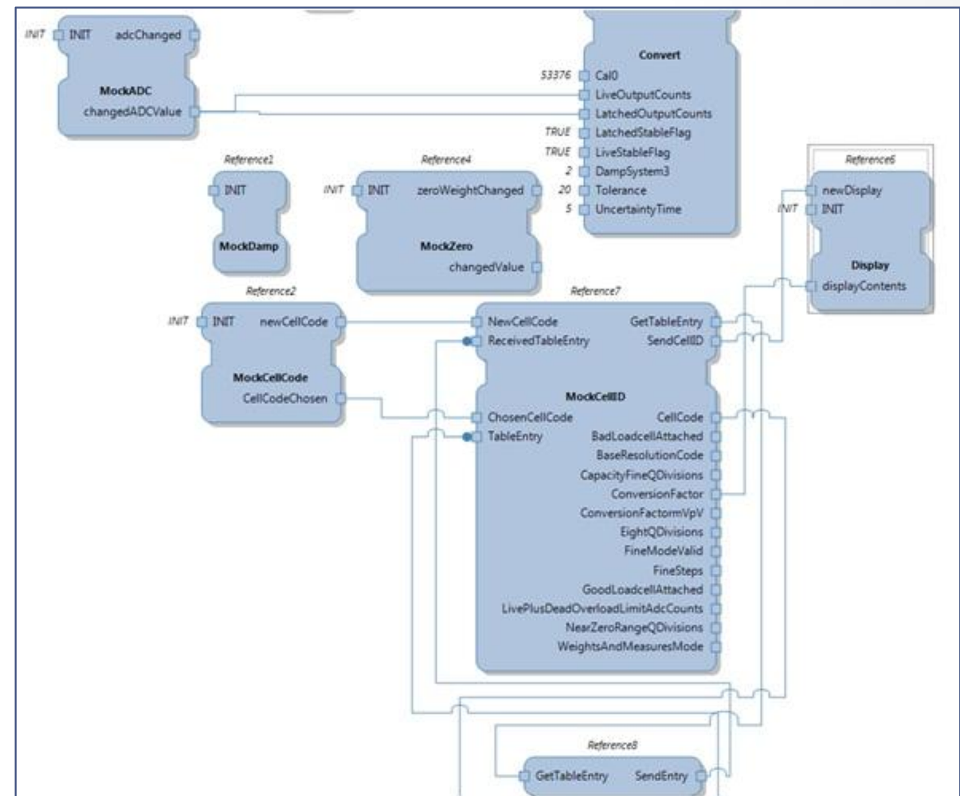
- Hard to debug and maintain code (200k lines of C code developer in over 20 years). With time, speed of design drops off and existing code base is hard to alter.
- Non modular design makes it hard to reuse code.
- Lack of visualisation of data flow between modules introduces high cost of code maintenance.
- Process of documentation is quite labour intensive and prone to quickly becomimg out of date.
- While there is an abundance of validation/simulation tools available there is a noticeable lack of high-level validation/verification techniques such as observer-based verification and model simulation.

IEC61499 based Solution

Visualization

Function Blocks





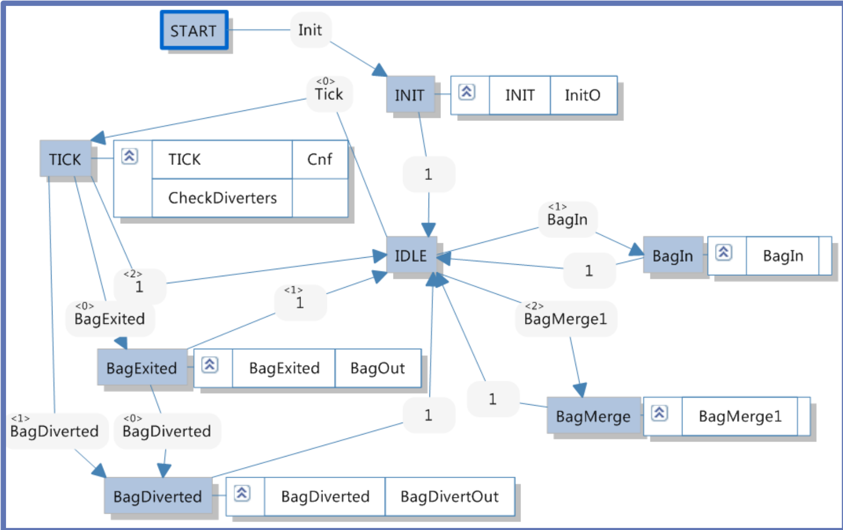
Proposed Approach

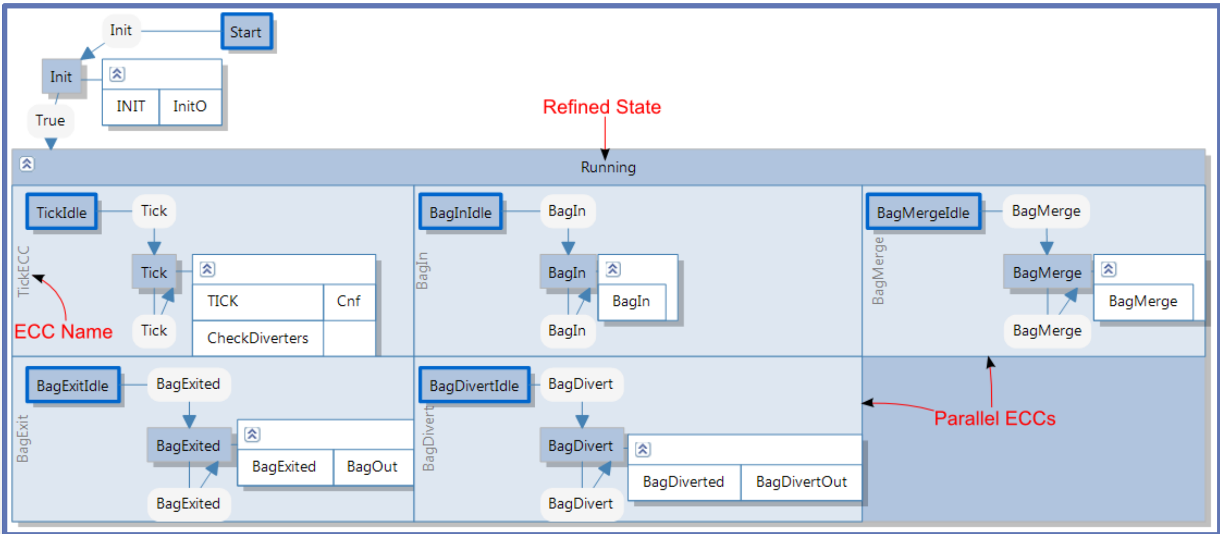
- We are trying to reverse engineer a part of this weighing application in IEC61499.
- This subset consist of a series of signal processing modules, which have been now captured using IEC61499.
- We have used TimeMe to generate code and have performed visualization-based testing on a PC and hardware-in-the-loop simulation.
- The generated code is behaviorally equivalent.

Extensions to IEC 61499

- Need for capturing the Hierarchy and Concurrency at function block level.
  - Hierarchical Concurrent Execution Control Charts (HCECC)
- Need to bridge the gap between industry practices (i.e. UML) and IEC 61499.
  - Mealy/Moore hybrid function blocks
- Need to interop with legacy code and existing system designs.
  - Reusing existing C code files in algorithms using header files.
- Verification and Validation
  - Observer function blocks
  - CTL verification
  - Tick-based simulation

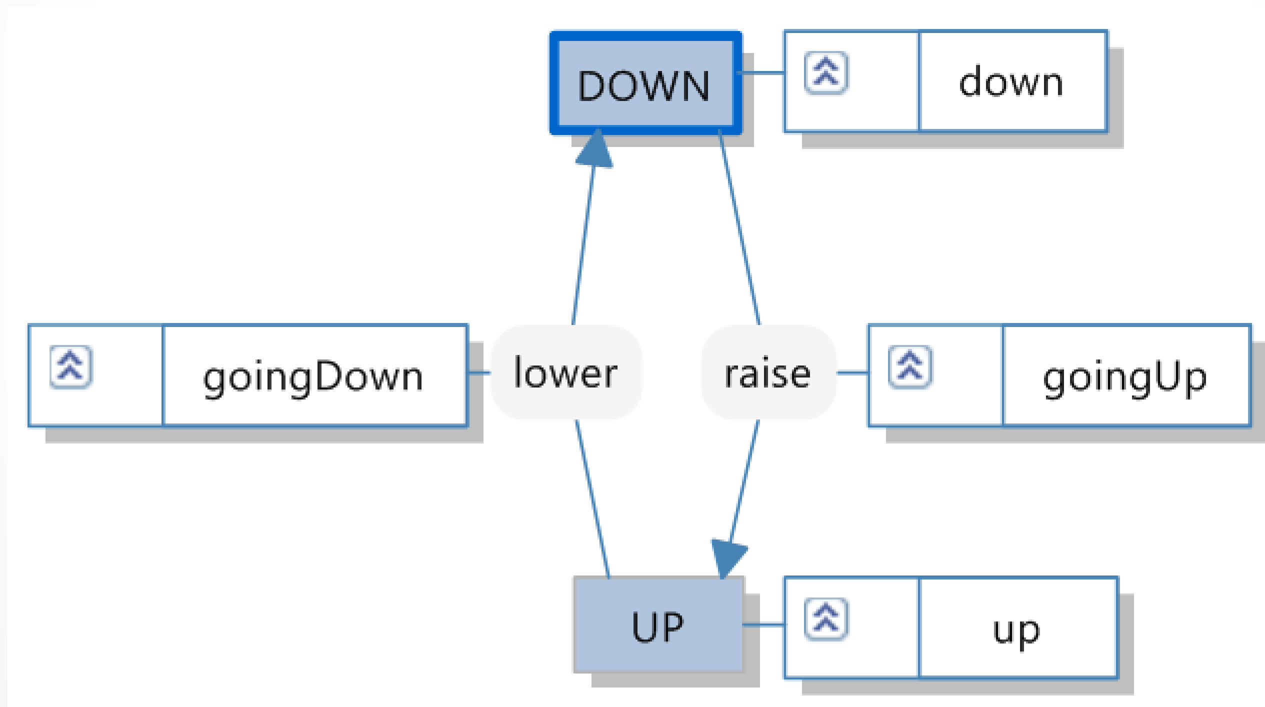
HCECC





Mealy/Moore Hybrid

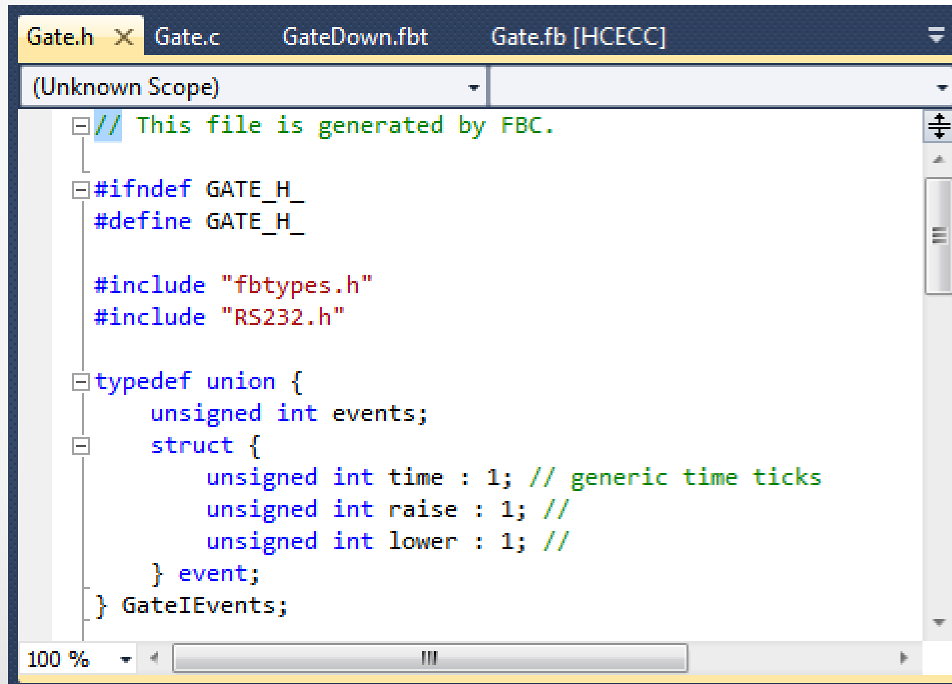
- Results in a smaller state machine.
- Makes IEC 61499 compatible with existing designs.

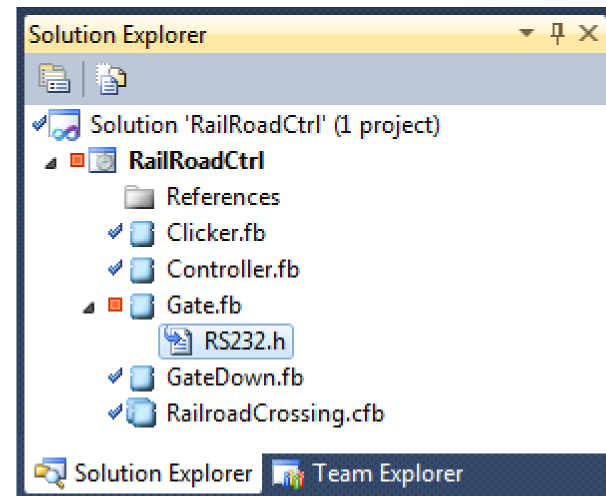


Re-using Legacy Code

- Adding C header file to IEC 61499 XML.

```
<CompilerInfo header="#include &quot;RS232.h &quot;"
      classdef="">
```





Client feedback

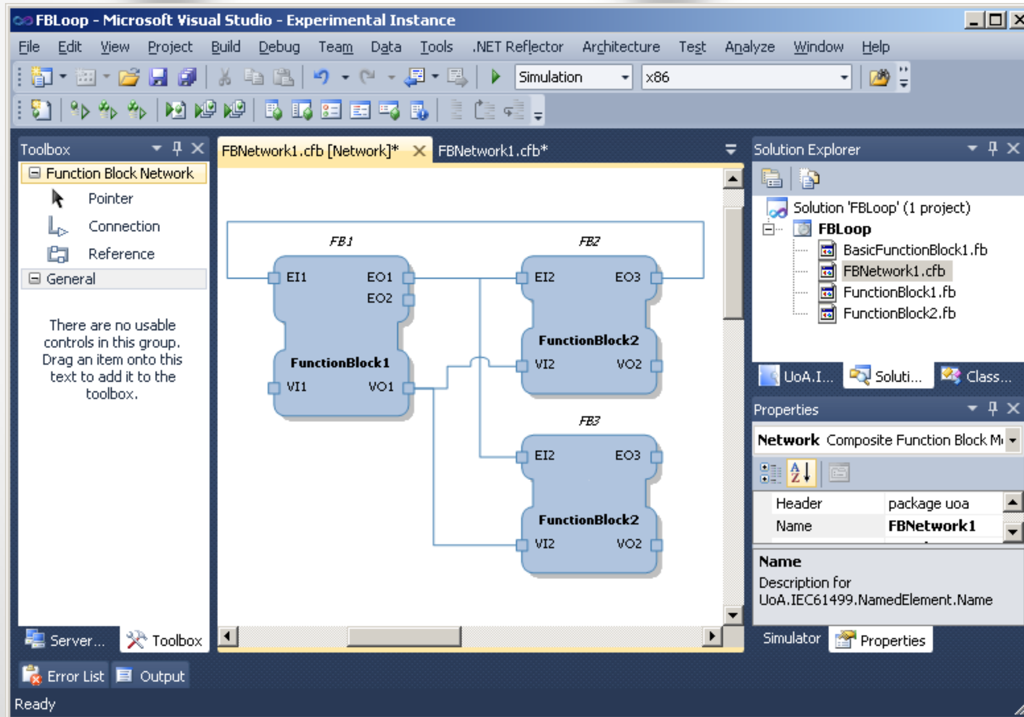
- Main advantages:
  - Explicit connection between modules / functions very useful for code maintenance.
  - HCECCs useful in their design-flow.
  - They envisage that adopting this new paradigm will slow down initial projects.
  - They also envisage that they will gain after a few projects through code reuse.
  - They also envisage reduction in maintenance cost.

Tool Chain Characteristics

A very efficient IEC 61499 code generator

Generates deterministic and deadlock-free code

Only IEC 61499 implementation that doesn't need a run-time environment



Supports observer-based formal verification

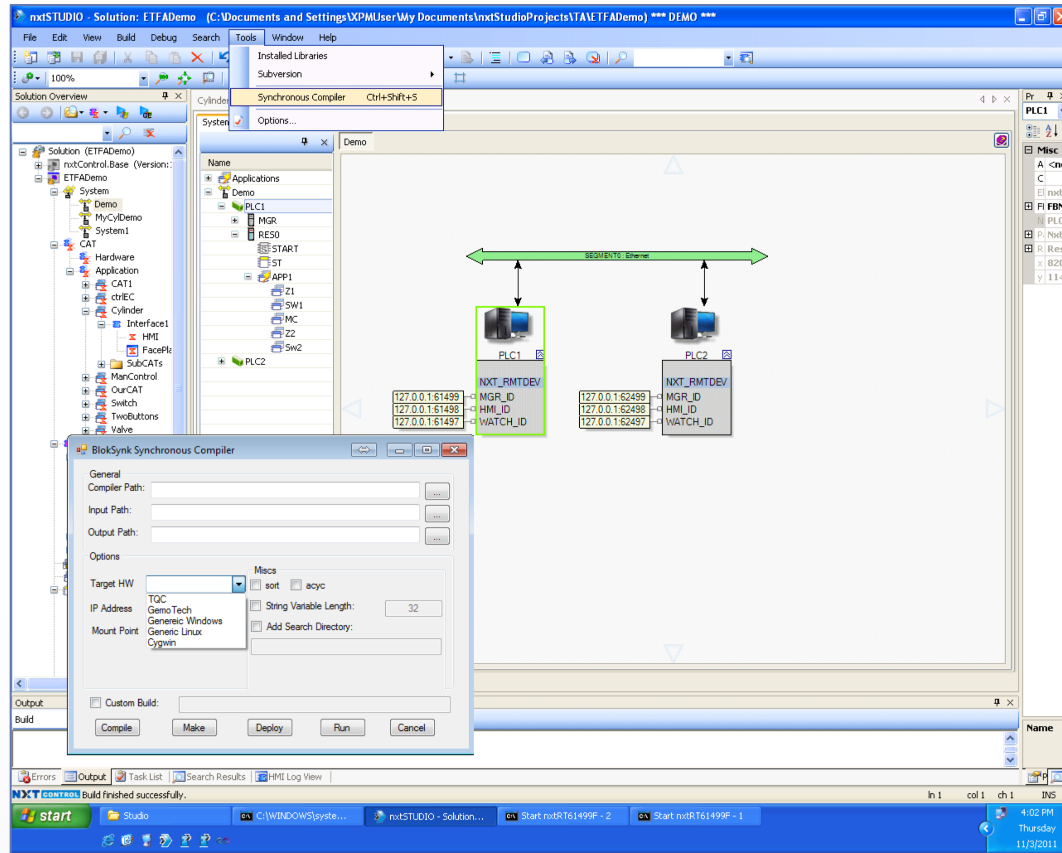
Enables static timing analysis

C Code Generation

```c
/* Function block initialization function */
void DistStnWithNetworkinit(DistStnWithNetwork*
{
    memset(me,0 , sizeof(DistStnWithNetwork));
    me->_input.events = 0;
    SUBSCRIBE_5_1init(&me->Inputs);
    PUBLISH_5_2init(&me->Outputs);
    DistributionStationinit(&me->DistStn);
    SUBSCRIBE_0init(&me->InitSub);
    PUBLISH_0init(&me->InitConfirm);
    strcpy(me->Inputs._ID, "127.0.0.1:6777");
    me->Inputs._QI = true;
    me->Outputs._QI = true;
    strcpy(me->Outputs._ID, "127.0.0.1:6888");
    me->InitSub._QI = true;
    strcpy(me->InitSub._ID, "127.0.0.1:6666");
    me->InitConfirm._QI = true;
    strcpy(me->InitConfirm._ID, "127.0.0.1:9999
}
```

```c
            if (me->_input.event.PosChan
                me->_state = 1;
                me->_entered = false;
                continue;
            }
        }
        break;
    case 1:
        // State: GoingForward
        if (!me->_entered) {
            DistStnPusher_GoForward(me);
            me->_output.event.PusherCtrl
            me->_entered = true;
        }
        else {
            if (me->_input.event.PosChan
                me->_state = 2;
                me->_entered = false;
                continue;
```

Plugin to NxtStidio



Simulation

Formal Verification

CruiseControl - Microsoft Visual Studio

CruiseControlObs.cfb [Network]*

controller

	CruiseManager	
set	set	regulOff
off	off	regulSet
resume	resume	regulStdby
quickAccel	quickAccel	regulResume
quickDecel	quickDecel	speedSet
brakePressed	brakePressed	
cclock	cclock	
	speed	cruiseSpeed

speedo

	SpeedGauge	
cclock	cclock	speed
time	time	speedVal
ryCount	rotaryCount	

CruiseControl - Microsoft Visual Studio

Resume.fb [HCECC]*

regulStdby

set

breakPressed

NotInStandBy

InStandBy

Violation

True

Success

The verification of observer "observer" was successful.
No counter examples were found.

OK

Ready

Static Timing Analysis

CruiseControl - Microsoft Visual Studio

File Edit View Qt Project Build Debug Team Data Tools Mono Architecture Test Analyze Window Help

Timing Analy

cc1af4e1-8b54-4f2b-981e-0155

Throttle.fb [HCECC] Throttle.fb Resume.fb [HCECC] CruiseControlObs.cfb [Network]

Server Explorer
Toolbox

<1> accelPressed
<2> accelReleased
<3> cclock
NORMAL
normal
throttleChg
<0> cruiseOff
<2> cclock
<1> accelPressed
ACCEL
<0> cruiseOn
<0> cruiseOff
<2> cruiseOn
CRUISE
cruise
throttle
<3> speedSet
<4> cclock

Success

Timing analysis of function block was successful.

The Worst Case Reaction Time value ="192"

OK

FunctionBlock Explorer
Solution Explorer
Team Explorer
Class View

Error List Output Find Symbol Results

Build succeeded

Deployment

WagoDistStation - Microsoft Visual Studio

File Edit View Qt Project Build Debug Team Data Tools Mono Architecture Test Analyze Window Help

Deploy

cc1af4e1-8b54-4f2b-981e-0155

WagoSystemWithIO.cfb [Network]

WagoSystemWithIO.cfb

Server Explorer

Toolbox

Inputs

INIT

INIT

CNF

WagoInputBlock

0

ByteIndex

DI0

DI1

DI2

DI3

DI4

DI5

DI6

DI7

CNF

Block

Deployment

Wago Device Deployment

Please provide the wage device connection parameters for the deployment process

IP Address: 192.168.1.190

Username : root

Password : ••••

Deploy

Cancel

UoA.IEC61499.CompositeFunctionBlock Explorer

Solution Explorer

Tear

Error List

Output

Find Symbol Results

Build succeeded