Logic control

Implementation using IEC 61499
Distributed Control: Introduction

- Requirements on the modern manufacturing technique
  - “agile manufacturing“
- Foretime: The Vision (Iacocca Institute, 1991)
  - Production to Order
  - Lot/Batch size $\geq 1$ Unit
  - Information intensive
  - “reprogrammable, reconfigurable, continuously changeable“

- Today:
  - Production to order is possible
  - Period between order and supply: 3-6 months (cars), 1 week (PCs)

- Near future
  - Japanese Vision: "3-day car"
  - Toyota/Canada (planned): "5-day car" (limited configurableness)
Distributed Control: Requirements on the automation system

- Dynamic reconfiguration

- Physical reconfiguration
  - Distributed automation systems required

- Logical reconfiguration
  - Dynamic Reorganization of the plant
  - Minimum human interferences (possible none)

- Not only parameterization
  - Lead to big, complex software systems
  - Makes distribution and flexibility more difficult
Outline

- What's the Problem with Distributed Control anyway?
  - Why not use the known approaches to do it!

- There is a new Standard?
  - What it describes and especially what it not describes!

- Is it useful?
  - Yes, but there are some problems!

- Is it complete?
  - No, but it can be made complete!
What's the Problem with Distributed Control anyway?

- Centralized control means scan-based execution
- All input data is scanned at the same time
- All algorithms are executed in a sequence
- All output data is written at the same time
- Main Features:
  - The scan-cycle assures that all the algorithms work on the same process data
  - The sequential execution is used to assure that algorithms work on current data
- In a distributed system (networked intelligent devices) there are several asynchronous scan-cycles
- Input data may be read at different instants in time
- Algorithms may be executed concurrently
- Output data may be written at different instants in time
- Resulting Problems
  - Several algorithms may work on different samples of the process data
  - The execution order of algorithms is no longer clear. Algorithms may work on old results of other algorithms.
Why not use the known Approaches to do it?

- Problems can be fixed by message exchange between algorithms
- Algorithm A informs Algorithm B that he has now finished calculating the data B needs
- Now the execution order is no longer relevant for the result of the calculation (but still for efficiency)

- Problems with inconsistent I/O is fixed by an additional I/O-image
- Input reading and output writing is modeled as an algorithm (additional internal I/O image)
- Based on message exchange (as above) validity is assured

- Why not implement this in IEC 61131
- Large overhead because it is not a feature of the standard
- Error-prone if built manually
Is there a standard?

- IEC 61499 is an approach to solve exactly this problem

- Definition of a new type of **Function Block**
- Definition of **Events** to control execution and to indicate validity of data
- Definition of **Service Interface Function Blocks** to actively communicate with the process (and the network)

- Definition of **Composite Function Blocks** to allow hierarchical structuring
- Definition of a surrounding model of system/device/resource
- Definition of management functions
- Definition of standard libraries …

- NO definition of new programming language (IEC 61131-3 is re-used)
- NO definition of a development process
- NO definition of function to system mapping

- NO Object-Oriented Paradigm for Automation
Object-Orientation in IEC 61499

- A Function Block according to IEC 61499 in an Object
  - Instantiation from a Function Block Type (Class)
  - Encapsulation of Data
  - Encapsulation of Algorithms (Procedures)

- IEC 61499 as a whole is NOT Object Oriented
  - No Inheritance
  - No OO development process

- IEC 61499 combines concepts from OO with the FB concept already accepted in engineering (IEC 61131, Simulink, …)

- Problem: The IEC 61499 FBs follow an execution model not readily known and accepted by people familiar with other FB concepts
  - EVENT-BASED EXECUTION
What is an Event?

- Signals discrete or continuous are defined at all time
- This is also true for time-discrete representations
- An Event occurs spontaneous
- (In Theory) has no duration
- Events are “consumed” by processes

→ You must not miss an event

- Signals (changes thereof) may be transformed to events
- Events may be transformed to signals
- Most naturally: Binary signal B1 is converted to a set of two events: E1=Change to Zero, E2=Change to One
- Not so naturally: Analogue signals

→ Events fit well in manufacturing but less in process applications
IEC 61499

- Network of Function Blocks
- Two ways of information flow
- Data flow for user/process data
- Control flow controlling the validity of the user data by events

- Model with several hierarchical layers
- System
- Device
- Resource (extension to IEC61131)

- Application (distributed over several resources)
- Function Blocks (extension to IEC61131)
IEC 61499 Funktion Block Model

- Funkcjons Blocks are the smallest functional units of an application (like FB POUUs in 61131)

- Execution Control: Generation and Processing of Events at Event Inputs and Event Outputs (Control Flow)

- Algorithms with Data Inputs and Data Outputs as well as internal Data (Data Flow)

- Programming of the Algorithms like the POU body in IEC 61131-3 (using the same languages)

- Programming of the Execution Control using a state diagram (ECC execution control chart)
Basic Function Block
Event/Data Associations

- Events may be associated with data
- Data is read by the block when the associated event occurs
Execution Control Chart (ECC)

- Separation of Control Flow (Events) and Data Flow
- Local Execution Control in each FB
- Algorithms are implemented using IEC 61131-3 languages or others like C or JAVA
Execution of an IEC 61499 FB

**Scheduling-Function (OS)**

**Event inputs**
- EX
- IN_1

**Event outputs**
- EXO
- OUT_1

**Type identifier**
- Algorithms (IEC 61131-3)
- and/or JAVA

**Internal variables**

**Execution Control Chart**

**Input variables**

**Output variables**

**Execution of an IEC 61499 FB**

**EC initial state**
- START

**EC transition**
- 1

**EC action**
- EX&IN_1

**EC state**
- MAIN
- INIT

**Event**
- event

**Algorithm**

© Tanvir Hussain
Scan-based (time-triggered, cyclic) Execution in IEC 61499

Scan-based execution is a special case of event-based execution.
Distributed Application and System Model

Event flow

Data flow

Application

= Function Block Network

Communication network

Application A

Application B

Device 1

Device 2

Device 3

Device 4

App. C

System

= Communication Network

+ Devices

+ Process / Machines

Controlled Process
IEC 61499 Device-Modell

- Device contains
- Application program
- A interface connected to communication network
- A interface connected to process
- the device hardware, on which the resources run

![Device boundary diagram with resources and applications]
IEC 61499 Resource Model

- A resource represents an independent executable and parameterizable unit (Task in general sense). There may be several resources run on one device, which realize the same or different applications.
- Resource is in charge of scheduling and execution of FB-Algorithms
- Resource represent the communication- and I/O-function on the service-interface-function block.
Is it useful?

- YES, it solves the main problem of distributed systems
- Consistency of data
- Explicit execution model

However there are open Problems

- Unclear definition of
  - Execution model
  - event-handling
  - data-handling
- Not an implementation standard (no tests defined)

Several compliant tools will present different behavior
Execution of a FB-Network

- **Sequential:**
  For FBs running in the same resource under a single-task (possible PLC-like) execution model.
  Note: even this simple case has an additional problem: unlike for example in the FB Diagram of IEC 61131, 61499 does not define an order of the constituent FBs in a diagram. Hence here is another weak point that could lead to different interpretations.

- **Synchronous:**
  For FBs running in one resource under a multi-tasking-system that realizes task switching times very short compared to the execution times in the application. In this case, it could be safely assumed that the algorithms run in parallel.

- **Asynchronous:**
  For FBs running on different resources where it is not possible to make an assumption like in the synchronous case.
Event and Data Handling

- How to implement Events
  - Messages
  - Shared Variables (same resource)
  - Technical Problem could be solved

- How to handle Events
  - Event occurs while FB is still processing the same type of event (unsafe state)
  - Different FBs are waiting for Events from each other FBs (blocking)
  - Different routes in the network (hazards)
  - DES Theory can solve (analyze) these properties iff the model is clear (1st point)

- When is data actually read
  - Occurrence of the Event at the input-port
  - Consummation of the Event by the ECC
  - Could be solved by encapsulating data and event in one message
Is it complete?

- NO

- Development Process is not defined

- FBs are not suitable for all stages in a development process

- UML seems to be the solution
  - several approaches are already published and will be further investigated

- Mapping problem is not solved
  - at least two groups are working at this
  - Component models are needed for HW and SW
  - Metrics have to be defined
  - The rest is optimization. However we will need rules to reduce search space