A Novel Approach for Integrating IEC 61131-3 Engineering and Execution into IEC 61499

Peter Gsellmann[1], Martin Melik-Merkumians[1], *Member, IEEE, IES*, Alois Zoitl[2], *Member, IEEE, IES*, and Georg Schitter[1], *Senior Member, IEEE*

***Abstract*—Automation system engineering becomes more complex, due to the trend towards more flexible, reconfigurable, and modular design approaches, like Industry 4.0. For the modeling and design of the according software, two standards are present: IEC 61131-3 and IEC 61499. In order to satisfy the requirements for modern, large scale, highly-distributed applications while also supporting still existing legacy systems, the demand for a combined development framework arises, where the best of breed tool can be chosen for a given automation task. Considering that, the IEC 61499 model is extended to allow the dual development and execution of IEC 61131-3 programs, and enabling easy and correct interaction between the two paradigms. In order to verify the validity of the chosen approach, an IEC 61499 development tool and a runtime environment is modified to support IEC 61131-3. A sample application is implemented, which comprises a pure IEC 61131-3 part with a 1 ms cycle time, a pure IEC 61499 part, and a part with interaction between both subparts, in order to evaluate possible interference between the runtime parts. Experimental results show that no interference is occurring, and the chosen development approach allows the seamless integration of IEC 61131-3 and IEC 61499 in one combined development framework.**

***Index Terms*—IEC 61499, IEC 61131, interoperability**

I. Introduction

IN CURRENT automation systems, control software is the main driver for functionality and innovation, and therefore is a significant component. Hence, its development makes a large share of the overall costs. Considering nowadays trend towards Industry 4.0, the requirements regarding interoperability, flexibility, and reconfigurability gain importance [1]. Currently, when developing such systems, engineers have to choose between two prominent standards: the IEC 61131-3 – Programmable controllers: Programming languages [2] and the IEC 61499 – Function blocks [3]. The IEC 61131-3 standard's main focus was on easy-to-use programming languages, and single Programmable Logic Controller (PLC) systems, each controlling a defined section of the production process. With the move to modern large scale applications, the control software development had to deal with features like adaptability, reusability, and distributability. IEC 61131-3 evolved (e.g., object-oriented extensions, IEC 61131-5 for communication) to meet these new needs, though it was never designed with these developments in mind [4]. Consequently, a new architecture, the IEC 61499, was developed to satisfy these emerging requirements.

Nevertheless, IEC 61131-3 based systems are still prevalent in industry, due to legacy systems and well-trained staff for this type of programming model. There have been initiatives to support the transition by enabling re-use of the already existing PLC applications. Several studies analyzed and realized tools for a semantic transformation from IEC 61131-3 to IEC 61499 [5], [6], [7]. However, in the recent years, due to the trend towards highly-distributed Cyber-Physical Production System (CPPS), the distribution aspect of control systems became more relevant. This is where IEC 61499 excels, as this distribution aspect was considered in the design process. Although, IEC 61131-3 could be extended to support model driven design and planning of distributed control, this is a nontrivial task [8]. In this aspect, IEC 61499 is superior to the IEC 61131-3 model, as distribution is an inherent part of IEC 61499 system design.

Another important aspect of system engineering is the needed programming effort and resulting code complexity, as this directly affects engineering, commissioning, and maintenance effort. A recently conducted study [9] gives an objective comparison between IEC 61499 and IEC 61131-3 applications based on code measures. Typical application classes in industrial automation, a sequential control and a PI control application, have been developed in both programming models and then evaluated for the suitability of each programming model for the given task. The results show, that the implementation effort for the sequential task is significantly less for IEC 61499 `APPLICATION`s, whereas the IEC 61131-3 Structured Text (ST)/Function Block Diagram (FBD) implementation excels for control algorithms. Each programming model has its strengths and weaknesses, which translates directly into programming effort and code complexity.

These new requirements demand for a combined framework for IEC 61499 and IEC 61131-3 compliant systems, in order to offer the best of breed tool for a given automation task, and thus motivate the contribution of this article. Alongside to the development of a concept for a combined IEC 61499-based Runtime Environment (RTE) and an engineering approach to model IEC 61499 and IEC 61131-3 applications, also the means to achieve easy interaction are taken into account. Furthermore, a sample application is presented, showing the validity of the presented approach. Finally, measurements are conducted to prove that the two execution units within the same device are not disturbing each other, even while

Manuscript received July 7, 2020; revised September 18, 2020; accepted October 19, 2020.

[1]P. Gsellmann, M. Melik-Merkumians, and G. Schitter are with the Automation and Control Institute, TU Wien, Vienna, 1040 Austria.
[2]A. Zoitl is with the LIT Cyber Physical Systems Lab, JKU, Linz, 4040 Austria.
Corresponding author: gsellmann@acin.tuwien.ac.at

Post-print version of the article: P. Gsellmann, M. Melik-Merkumians, A. Zoitl, and G. Schitter, "A Novel Approach for Integrating IEC 61131-3 Engineering and Execution into IEC 61499," *IEEE Transactions on Industrial Informatics*, 2020. DOI: 10.1109/TII.2020.3033330

exchanging data.

In Section II, an analysis of existing approaches and evaluation of their deficiencies are presented. The integration approach for the combined IEC 61131-3 and IEC 61499 framework is discussed in Section III. Section IV deals with the implementation of the particular development and runtime environment. It is followed by the presentation and evaluation of a demonstration implementation in Section V. Finally, Section VI concludes the paper.

II. Related Work

In the past, several approaches of a combined RTE for both IEC 61131 – Programmable controllers [10] and IEC 61499 were presented with the aim of implementing a platform that supports any control logic. These can be roughly categorized into three classes (see Fig. 1) [11]:

(a) Separate IEC 61131 and IEC 61499 RTEs with a communication interface in between.
(b) Extended IEC 61131 RTE with the means to execute event-based IEC 61499 logic.
(c) Extended IEC 61499 RTE with the means to execute cycle-based IEC 61131 logic.

A representative for the first class (Fig. 1a) is designed and implemented in a previous study, where the proposed architecture is based on loosely coupled systems [12]. Thereby, every device can execute both IEC 61131 and IEC 61499 code by using two separated execution environments, which can interact via a communication interface. While offering the advantage of using existing applications on both sides, this implementation requires a lot of engineering effort and resources. Additionally, the re-use of software elements from one standard in the other is infeasible.

As an implementation of the second category (Fig. 1b), where an IEC 61131-3 RTE is extended to enable the execution of IEC 61499 logic, ISaGRAF of Rockwell Automation is considered. Within a `RESOURCE`, IEC 61499 Function Blocks (FBs) are cyclically invoked in a predetermined order [13]. During the activation, the Execution Control Chart (ECC), defined by means of the IEC 61131-3 Sequential Function Chart (SFC), recognizes and processes incoming events. Although this solution has the advantage that IEC 61131-3 and IEC 61499 can be used side by side, several drawbacks appear. Besides the higher effort to implement an IEC 61499 compliant RTE, and to schedule events according to the best execution order [14], this approach exhibits performance penalties and an execution overhead due to the underlying cyclic execution. Furthermore, several IEC 61499 concepts, such as communication between application parts via Service-Interface Function Blocks (SIFBs), are not compliantly implemented in ISaGRAF [15]. Similar issues occur in a different study, where an approach to map event-driven IEC 61499 execution to a scan-based controller system has been presented [16], [17]. Even though this concept enables deterministic execution behavior and thus the possibility for pre-verification, it is not always possible to calculate a scan order to follow the event propagation in complex Function Block Networks (FBNs). Moreover, again the large overhead introduced due to

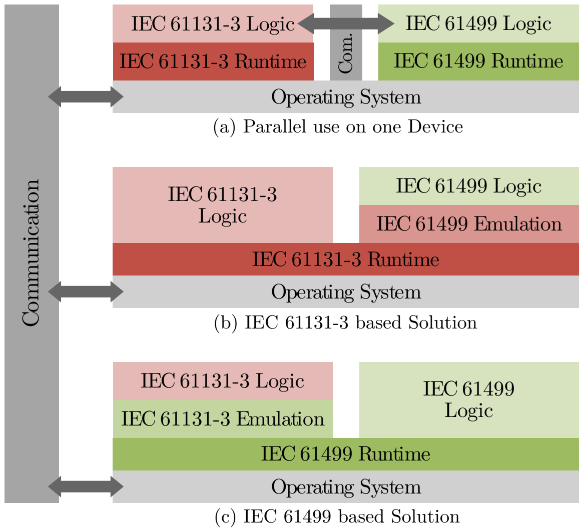


Figure 1: Existing concepts for the combined use of IEC 61131-3 and IEC 61499 (based on [11]).

the cyclic triggering increases the overall response time of the application [18]. In order to overcome these overload effects, a synchronous approach for the execution of IEC 61499 FBNs implementation is proposed [19]. Instead of executing an application on a RTE, IEC 61499 `APPLICATION`s, FBNs, and FBs definition are translated into Estrel code, which results in a deterministic finite state machine, representing the full application. The resulting compiled Estrel program is fully predictable, and can be compiled to, for example, C code for specific platforms. However, this is contradictory to main features of the IEC 61499, since important management functionalities such as online reconfiguration are no longer possible [20].

The third approach (Fig. 1c), where IEC 61131-3 code is executed in an IEC 61499 system, seems to be the most flexible and thus most advantageous of the in [11] proposed approaches. It allows not only the re-use of existing IEC 61131-3 code, but also the possibility of a tight coupling to IEC 61499 control logic and the ability to distribute them among several resources surpasses the other two mentioned approaches. Therefore, it revives the original idea of IEC 61499 as an enabler for modeling distributed systems and the coordination between the individual devices, which were meant to be based on IEC 61131-3 [21]. So far, the only comparable solution is shown by nxtControl with their engineering tool nxtSTUDIO and their hybrid RTE nxtIECRT [22]. Here, IEC 61499 is extended with a special IEC 61131-3 FB, to add the IEC 61131-3 programming model. This FB is wrongly classified as a Basic Function Block (BFB), since it comprises no explicit algorithm execution control. It is also no representative of the Simple Function Block (SFB) type, since it does not represent IEC 61131-3 `FUNCTION`s or FBs. Hence, this special IEC 61131-3 FB is a SIFB, which comprises a complete IEC 61131-3 sub-system. This SIFB has a single

Post-print version of the article: P. Gsellmann, M. Melik-Merkumians, A. Zoitl, and G. Schitter, “A Novel Approach for Integrating IEC 61131-3 Engineering and Execution into IEC 61499,” *IEEE Transactions on Industrial Informatics*, 2020. DOI: 10.1109/TII.2020.3033330

event input named TASKI, a single event output named TASKO, and application specific data in- and outputs. By means of an E_CYCLE FB, the event input TASKI is triggered and subsequently, the implemented Program Organisation Unit (POU) is executed. When reaching the POU's end, the output event TASKO is sent. Although the solution of nxtControl is a first step towards the harmonization of IEC 61131 and IEC 61499, several issues are evident:

1) Usually in IEC 61131, with the start of a new cycle a process mapping of the most current data inputs is carried out. Though the implemented SIFB represents an IEC 61131-3 TASK, it cannot be guaranteed that multiple serially ordered TASKs operate on the same input data. Conversely, parallel running TASKs offer no viable solution for interaction during their execution.
2) For event-based control, only the output event TASKO can be utilized since the only event input TASKI is occupied with receiving the cyclic trigger.
3) Since the output event TASKO is triggered after the execution of the IEC 61131 PROGRAM, large applications can produce blocking behavior due to the run-to-completion semantics of IEC 61499, to which the special IEC 61131-3 FB also adheres to.

A. *Summary*

All the presented solutions allow mixed IEC 61131-3 and IEC 61499 applications. However, all of them show significant drawbacks. For the separated RTEs (Fig. 1a) no common system modeling is available. This increases the difficulty of engineering such systems, as software cannot be shared and interactions between both system types have to be implemented manually for each case. Running IEC 61499 compliant systems on cyclic IEC 61131-3 systems (Fig. 1b) increases execution overhead, and several IEC 61499 elements are hard to implement due to their event triggered nature [15].

A first implementation for the approach which executes IEC 61131-3 PROGRAMs on top of IEC 61499 compliant systems (Fig. 1c) is implemented by nxtControl. It utilizes a special SIFB, which couples the IEC 61131-3 cycle to the event-driven nature of the IEC 61499 application, that can result in unwanted side effects, such as unstable cycle times or data inconsistencies over the whole application. Also, all three presented solutions lack a common engineering approach, which simultaneously allows to model applications in their respective programming and execution models.

III. Integration Approach

As an initial step to create a common system modeling base, it has to be decided if the IEC 61131-3 or the IEC 61499 model shall serve as the foundation. The analysis on the different implementation options in Section II shows, that the IEC 61499 is a better choice, as the IEC 61131-3 model has no inherent support for distribution. In addition, the execution model of IEC 61499 appears to be better suited to host an IEC 61131-3 runtime environment, than vice versa. Although nxtControl offers a first implementation with the IEC 61499 as a basis, several deficiencies are evident. Hence, these reasons motivate the development of a new combined approach based on IEC 61499 (see Fig. 1c). An overview over the proposed system design can be seen in Fig. 3.

A. *Mapping of Model Elements*

Based on an analysis of IEC 61131-3 and IEC 61499 correspondences [23], the execution-driven mapping presented there seems the obvious choice for the integration of IEC 61131-3 model into the IEC 61499 model. As the IEC 61131-3 TASK corresponds to an IEC 61499 RESOURCE, a special RESOURCE type EMB_PLC_RES is created to provide an execution container for IEC 61131-3 POUs. The desired cycle time for this IEC 61131-3 execution container is provided as an additional resource parameter. As an IEC 61499 RESOURCE represents an independent unit of execution, the uninterrupted cyclic execution of the EMB_PLC_RES RESOURCE is guaranteed. Simultaneous execution of IEC 61499 and IEC 61131-3 is then simply achieved by dragging a standard EMB_RES and the new EMB_PLC_RES into an IEC 61499 device (see Fig. 2).

Contrary to the claim, that an IEC 61131-3 program corresponds to an IEC 61499 APPLICATION [23], no such analogon is utilized in the presented approach. The reason for this is, that IEC 61499 APPLICATIONs are inherently distributable to several IEC 61499 RESOURCEs, which are equivalent to IEC 61131-3 TASKs in this approach. However, IEC 61131-3 PROGRAMs are in their current form not distributable over several IEC 61131-3 tasks, so the analogy of IEC 61131-3 PROGRAMs and IEC 61499 APPLICATIONs cannot be maintained. Instead, an IEC 61131-3 programming environment is emulated, providing the basic needs to create an IEC 61131-3 PROGRAM, which is then executed by the assigned EMB_PLC_RES RESOURCE (see RTE part in Fig. 3). In principle, every programming language defined in IEC 61131-3 can be used, but as the FBD language has the largest overlap with the IEC 61499, it is primarily considered in this work.

Another important aspect of the combined programming environment is code sharing between the two worlds, in order to reduce maintenance and development effort. Here, the IEC 61499 SFB [3] is an eligible candidate, originally devised to represent IEC 61131-3 FUNCTIONs and FBs in IEC 61499 APPLICATIONs. It therefore represents the event-triggered analogon to IEC 61131-3 FBs without the object-oriented extensions added in the 3rd edition of the IEC 61131-

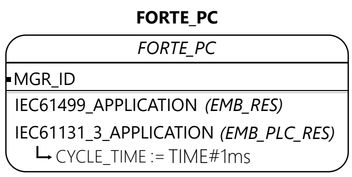


Figure 2: A DEVICE containing both an IEC 61499 EMB_RES RESOURCE for event-based applications, and an IEC 61131-3 RESOURCE EMB_PLC_RES for cyclic PROGRAMs. The cycle time of the EMB_PLC_RES RESOURCE is set by a parameter.

Post-print version of the article: P. Gsellmann, M. Melik-Merkumians, A. Zoitl, and G. Schitter, "A Novel Approach for Integrating IEC 61131-3 Engineering and Execution into IEC 61499," *IEEE Transactions on Industrial Informatics*, 2020. DOI: 10.1109/TII.2020.3033330

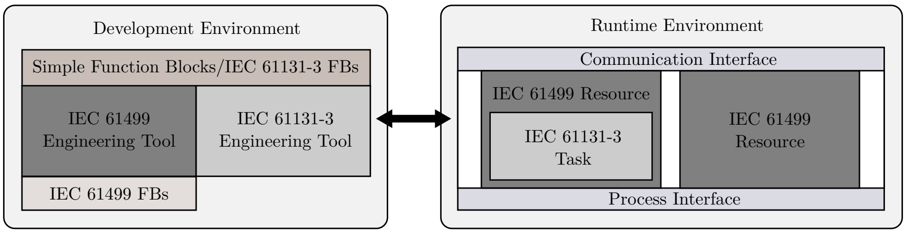


Figure 3: Schematic representation of the integration approach: The proposed IDE supports the engineering of IEC 61131-3 and IEC 61499 applications. Furthermore, the SFB or IEC 61131-3 FB can be used by both as common software element. The adapted IEC 61499 RTE enables the usage of IEC 61499 `EMB_RES` `RESOURCE`s, and IEC 61131-3 `TASK`s embedded in an IEC 61499 `RESOURCE`. All aspects of the IEC 61499 device model [3] (e.g., communication and process interface) can be used by both resource types.

3 standard. The IEC 61499 standard describes the SFB as a special type of SIFB, where each available input event triggers an associated algorithm provided by the SFB. The first issue with this definition is, that a SFB is a SIFB, which means its functionality is defined outside the scope of IEC 61499 and therefore only predefined SFBs can be used. For the proposed system this definition is changed, so that a SFB is a kind of simplified BFB without an ECC and only supporting a single algorithm to be executed. This directly leads to a second change: the original definition allows multiple algorithms, triggered by an associated input event, but as there is no event interface in IEC 61131-3 such a distinction cannot be made. Therefore, SFBs in the proposed system can only have a single input event `REQ` to trigger a single algorithm, and a single output event `CNF` to signal the end of execution of this algorithm. As a consequence of this all data inputs are associated with the single event input via the IEC 61499 `WITH` construct, and all data outputs are associated with the single event output. Now, when the IEC 61499 FB event head is removed, a one-to-one mapping from IEC 61499 SFBs to IEC 61131-3 FBs is achieved (see Fig. 4). With this modified SFB, implementations can be shared across the IEC 61499 and IEC 61131-3 subsystems and thus enables code sharing (compare development environment part of Fig. 3).

B. Interaction between IEC 61131-3 and IEC 61499

With the mapping concept presented so far, the IEC 61499 and IEC 61131-3 subsystems can coexist in one system solution, but no interaction is possible. Interaction is here defined as the ability to exchange data. As IEC 61499 directly

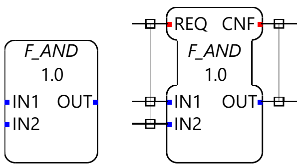


Figure 4: With the modified Simple Function Block (SFB) concept, SFBs can be used as IEC 61131-3 FBs, reducing development effort for dual standard environments. Here, the `F_AND` FB is shown in its dual representations.

adopts the data types as defined in IEC 61131-3, from a pure data view this interaction is trivial, with only the method of exchange missing. The chosen approach provides an explicit modeled interaction channel, utilizing the IEC 61499 communication FBs. Two different communication models are defined: *unidirectional interaction* via the `PUBLISH` and `SUBSCRIBE` FBs and *bidirectional interaction* via `CLIENT` and `SERVER` FBs. This communication model is used in IEC 61499 for inter-`DEVICE`, but also for inter-`RESOURCE` communication. Again, in order to share code between the subsystems, these FBs are transferred to the IEC 61131-3 subsystem. However, these and other FBs (e.g. I/Os) need an `INIT` event to be brought into a ready state. In order to provide this mechanism in the IEC 61131-3 subsystem, the `INIT` event of all FBs which require an initialization are triggered in the `IDLE` to `RUN` transition of the `RESOURCE`. With this mechanism in place data can be exchanged between the two subsystems.

C. Execution Behavior

At last, the execution behavior for the implemented IEC 61131-3 FBD `PROGRAM` is defined. Here, the IEC 61131-3 standard provides a set of rules for the algorithmic static calculation of the execution order:

- Evaluation of a network element starts when all inputs are available.
- Evaluation of a network element shall not be complete until the states of all of its outputs have been evaluated.
- Evaluation of a network is not complete until the outputs of all of its elements have been evaluated.

An exemplary FBD sequence evaluation according to these rules is shown in Fig. 5. After the input scan, which updates all `IN`s, the FBs are evaluated according to the above mentioned set of rules. After all FBs have been evaluated, the output scan is performed, publishing the calculated values to the outputs. The Roman letters indicate the calculated sequence. For the proposed system all I/O inputs and all receiving parts of the communication FBs (e.g. the FBs outputs) are considered to be inputs for the evaluation algorithm. Outputs in the sense of the evaluation algorithm are all I/O outputs and all sending parts of communication FBs (e.g. the FBs inputs). For all other FBs the execution order has to be calculated.

Post-print version of the article: P. Gsellmann, M. Melik-Merkumians, A. Zoitl, and G. Schitter, “A Novel Approach for Integrating IEC 61131-3 Engineering and Execution into IEC 61499,” *IEEE Transactions on Industrial Informatics*, 2020. DOI: 10.1109/TII.2020.3033330

In a formal definition, a FB can be described as a 2-tuple $F = (I, O)$, where I are the FB inputs and O are the FB outputs. In this representation, a `PROGRAM` input can be expressed as $(\emptyset, O_m)$, whereas a `PROGRAM` output can be as $(I_n, \emptyset)$. Consequently, the execution semantics can be described as a 4-tuple $P = (\Sigma_I, \Sigma_O, \Sigma_F, C)$, where

- Σ_I denoting the set of all FB inputs and `PROGRAM` outputs I_n,
- Σ_O denoting the set of all FB outputs and `PROGRAM` inputs O_m,
- Σ_F denoting the set of all FBs F_j,
- C denoting the $|\Sigma_I| \times |\Sigma_O|$ unweighted connection matrix of all connections between the elements of Σ_I and Σ_O.

Thus, by considering a IEC 61131 FBD `PROGRAM` represented as a 4-tuple P, the calculation of the static execution sequence is enabled and realized as a search procedure presented in Algorithm 1. With the assumption, that FBD models with implicit and/or explicit feedback loops are correctly solved by assigning valid initial values, according to the IEC 61131 standard, a correct execution sequence is always computable.

The execution sequence calculation takes a list of the inputs I, a list of the outputs O, a list of the FBs F, and a list of the connections C as parameters. After removing all preinitialized inputs from I, the algorithm searches for FBs in F which have no inputs in the input list I. If this is the case, the FBs are added to the execution sequence S, which is defined as an ordered set of FBs. Subsequently, all of the FBs outputs and consequently all inputs connected via the connections in C are removed from the I and O list, respectively. Thereafter, also the FBs are removed from F. This procedure is repeated until F is empty, which means that all FBs are set in the correct order for execution.

Algorithm 1 Execution sequence calculation

Input: List of inputs **I**, list of outputs **O**, list of FBs **F**, list of connections **C**
Output: execution sequence **S**

Remove all preinitialized inputs from I
while F is not empty **do**
  if Any F_i have zero inputs **then**
    $S \leftarrow F_i$
    Remove all outputs O_i of F_i
    Remove all via C connected inputs I_i
    Remove F_i from F
  end if
end while

IV. Implementation

From this conceptual design, a concrete prototype implementation is presented for both an Integrated Development Environment (IDE) and a combined RTE.

A. Integrated Development Environment

For the implementation of the extended IEC 61499 system presented in Section III-A, the open source IEC 61499 environment Eclipse 4diac™ is used as a base system. Eclipse 4diac comprises the extensible IEC 61499 compliant development environment 4diac IDE, along with the modular RTE 4diac FORTE. This first implementation considers only the IEC 61131-3 FBD language without object-oriented extension. Therefore, only slight modifications and additions to 4diac IDE are required in order to enable side-by-side development. The existing editor for IEC 61499 `APPLICATION`s is used as a basis for the IEC 61131-3 `PROGRAM` editor. In a first step the display of event heads and event connections is deactivated, since these features are superfluous for cycle-based IEC 61131-3 `PROGRAM`s. However, this guarantees that already existing SFBs, such as the event-driven pendants of IEC 61131 `FUNCTION`s and FBs, can be used without modifications in both application types and thus enhances reusability. Furthermore, the SFB editor enables the design of custom IEC 61131-3 FBs, consisting of data inputs and outputs, and one encapsulated algorithm. For the specification of start and end points of FB sequences, input and output elements are added which then can be linked to hardware registers or physical I/Os. The execution sequence of the so-developed IEC 61131-3 FBDs is then evaluated by the rules mentioned in Section III-C.

By adding the new IEC 61131-3 resource type `EMB_PLC_RES` to 4diac IDE, which provides a cycle time parameter (see Section III-A), IEC 61499 `DEVICE`s are able to contain both IEC 61131-3 and IEC 61499 runtime units and thus enable mapping of the according applications. An exemplary device with both types of resources is shown in Fig. 2.

B. Runtime Environment

The concurrent execution of cyclic IEC 61131-3 task and an event-based IEC 61499 runtime is achieved by adapting the IEC 61499 compliant runtime 4diac FORTE.

For the sake of clarity the event-driven execution model of 4diac FORTE is recapped [18]:

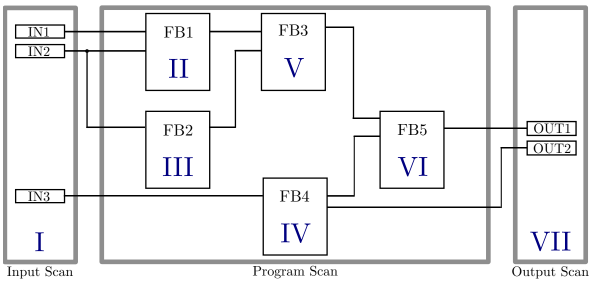
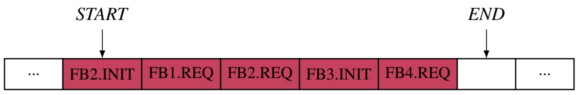
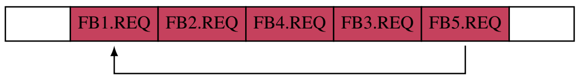


Figure 5: Static FBD execution order calculation: According to the IEC 61131-3 [10], first the inputs are read, then the program execution starts. Therefore, the FBD activation rules, described in Section III-C, come into effect. A FB is activated when all its input values are available. After activation, all FB's outputs are published for subsequent FBs. In this example the network elements' execution order is outlined by blue Roman numerals. After all FBs have been activated, the outputs are updated.

Post-print version of the article: P. Gsellmann, M. Melik-Merkumians, A. Zoitl, and G. Schitter, "A Novel Approach for Integrating IEC 61131-3 Engineering and Execution into IEC 61499," *IEEE Transactions on Industrial Informatics*, 2020. DOI: 10.1109/TII.2020.3033330



(a) Event queue execution: Occurring events are added to the list and are processed consecutively. After execution, the event entry is removed. The *START* pointer indicates which event is executed next, whereas the *END* pointer indicates where the next occurring event is added.



(b) Event queue adaption for the IEC 61131-3 cyclic execution model: FBs (actually their `REQ` events) are added in the calculated sequence (see Fig. 5), but are, in contrast to Fig. 6a, never removed. After the last FB has been activated, the event queue stops until the next clock cycle, establishing a constant cycle time.

Figure 6: Execution principle of the (a) IEC 61499 event queue and (b) the proposed IEC 61131-3 execution model based on the event queue.

- If events occur, they are added in order of their temporal occurrence to the event queue.
- As soon as the event queue contains one or more registered events they are executed consecutively.
- After execution, the event entry is deleted.
- If no events occur, the event queue stays empty.

The used event queue mechanism is illustrated by the linear representation of the circular event buffer (see Fig. 6a).

In order to create the recommended cyclic execution model from IEC 61131-3, the existing event-driven model is extended. As the IEC 61131-3 FBs are realized as SFBs according to Section III-A, they only have a single `REQ` event input. Thereby, this `REQ` event serves solely as a calling mechanism for the SFBs. Consequently, only the `REQ` events of the FBs have to be added to the event queue in the statically calculated execution order of the FBD. Since the IEC 61131-3 `PROGRAM` remains unchanged during the execution, the content of the event queue stays constant and thus the deletion of the executed events is disabled. Synchronized with the start of the execution, also a watchdog timer with a preset cycle time is started in a separate thread. After the last event is processed, the process waits for the expiration of the watchdog timer in order to start the next cycle (see Fig. 6b). In case that the `PROGRAM`'s execution exceeds the configured cycle time, malfunctioning behavior is assumed and the watchdog timer thread shuts down the IEC 61131-3 runtime.

V. Proof of Concept

A. Example Implementation

In order to guarantee correct behavior the combined IEC 61499 and IEC 61131 runtime, it has to fulfill the following requirements:

- The cyclic execution of the IEC 61131-3 `PROGRAM` must not be disturbed by the simultaneously processing event-driven IEC 61499 runtime – and vice versa.
- Communication between both subsystems shall be possible, without interfering the program processing while complying with the desired execution method.

Therefore, an appropriate example implementation for verification needs to include a pure IEC 61131-3 part, a pure IEC 61499 part, and a part with interaction between the IEC 61131-3 and IEC 61499 subparts. The proposed application is designed as follows: The IEC 61131-3 `PROGRAM` (see Fig. 8), running at a cycle time of 1 ms, consists of a `CTU` FB, which is generating a ramp on the `CV` output. Here, the FB's preset value, reset signal, and count-up signal are provided by the IEC 61499 `APPLICATION` via a `SUBSCRIBE` FB. The current counter value is additionally written to an analog output via a `QW` FB. In order to indicate that the counter reached the preset threshold, it's `Q` output is sent to the IEC 61499 `APPLICATION` via a `PUBLISH` FB.

In the IEC 61499 `APPLICATION` (see Fig. 7), a random threshold is generated. After reasonable scaling, this value, a start, and an end signal is transmitted to the IEC 61131-3 counterpart via a `PUBLISH` FB. Additionally, the threshold value is written on an analog output via a `QW` FB. In order to verify if the `CTU` FB in the IEC 61131-3 `PROGRAM` reached the preset threshold, a boolean indicator value is transmitted via a `SUBSCRIBE` FB. After reaching the threshold, the `CTU` counter value is reset and a new threshold is generated. The pure IEC 61499 `APPLICATION` consists of a digital output which is toggled every 2 s. This is achieved via an `E_CYCLE` FBs, generating cyclic events, and an `E_T_FF` FB, which toggles its `BOOL Q` output at every event received.

As an embedded execution hardware, a *Raspberry Pi Zero* with an 1 GHz single-core CPU is chosen, which is connected to a *Coolwell AD/DA expansion board*.

B. Analysis

In order to verify the timing behavior of the implemented combined runtime environment, an oscilloscope measurement is carried out (see Fig. 9). The implemented test program is continuously executed for an extended period, and then at a randomly selected time, a 1 s time window is recorded. In the selected time frame, the random value FB generated six different thresholds. For every threshold change, the value of the counter linearly increased with a constant slope. This indicates that the cyclic execution of the IEC 61131-3 part is not disturbed by the IEC 61499 parts, e.g. the toggling of the digital output. On the contrary, the toggle time of the digital output signal stays constant over the whole recording time, showing that the IEC 61131-3 cyclic execution does not interfere with the event-driven part. The communication between the IEC 61131-3 and the IEC 61499 RTE also does not interfere with their execution performance.

The analysis shows, that the demanded combined framework of IEC 61499 and IEC 61131-3 compliant systems works as users from IEC 61499 and IEC 61131-3 would expect. The possibility to use both standards to implement automation applications and to let them natively interact with each other, enables the application engineer to use the most suitable software tool for a given application.

Post-print version of the article: P. Gsellmann, M. Melik-Merkumians, A. Zoitl, and G. Schitter, "A Novel Approach for Integrating IEC 61131-3 Engineering and Execution into IEC 61499," *IEEE Transactions on Industrial Informatics*, 2020. DOI: 10.1109/TII.2020.3033330

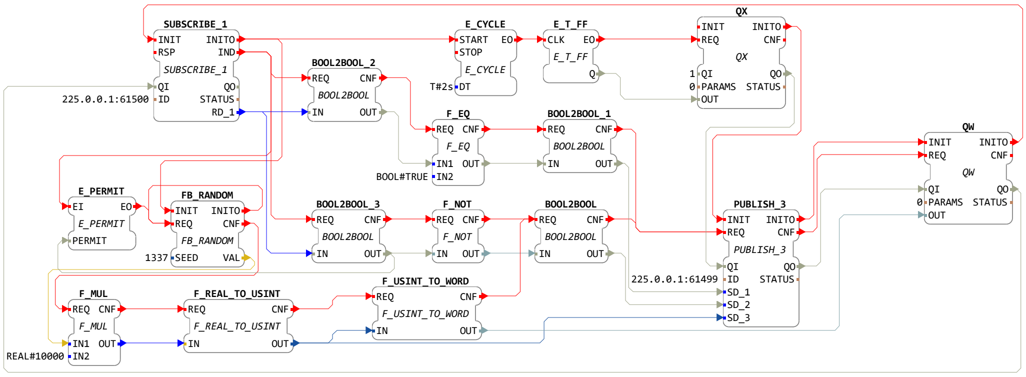


Figure 7: IEC 61499 part of the example implementation: The RANDOM FB generates a pseudo random number between 0.0 and 1.0, which is multiplied by a factor of 10.000 to gain reasonable output voltages. This resulting output value is set to an analog output, and doubles as the threshold for the building-up ramp signal. The threshold, the signals for counting up and resetting are sent via a PUBLISH FB to the IEC 61131-3 part. The SUBSCRIBE FB receives the Boolean value of the CTU FB, which indicates when the threshold is reached. Simultaneously, a digital output is toggled every 2 s.

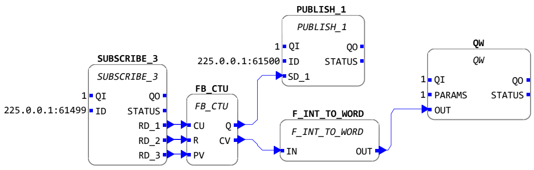


Figure 8: IEC 61131-3 part of the example implementation, running at a cycle time of 1 ms: Via the SUBSCRIBE FB, the signals for counting up, resetting, and presetting the threshold are transmitted from the IEC 61499 part to the CTU FB. The CTU FB is generating a ramp signal, which is put on an analog output. When the CTU FB count reaches the preset threshold, its Boolean indicator is transmitted to the IEC 61499 part via a PUBLISH FB.

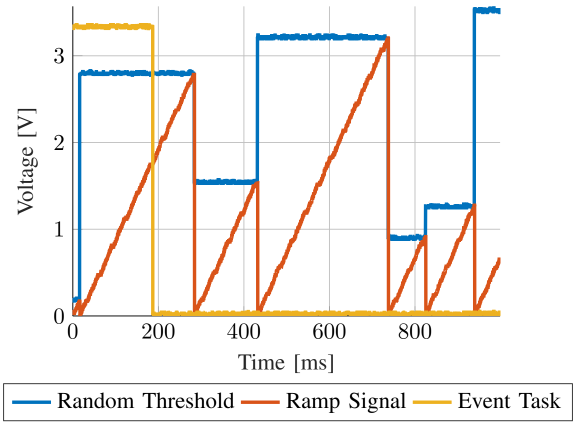


Figure 9: Verification measurement for the implemented RTE: The *Random Threshold* signal changes its value as soon as the *Ramp Signal* has the same value, triggering the generation of a new threshold. Simultaneously, a digital output is toggled via the event-driven application. It can be seen, that both coexisting RTEs do not disturb each other and perform as expected.

VI. Conclusion

This paper presents a development approach for a combined framework with the goal to model and execute IEC 61499 APPLICATIONs and IEC 61131-3 PROGRAMs. Thereby, an IEC 61131-3 TASK is directly mapped in IEC 61499 as a RESOURCE, and thus enabling use of both IEC 61499 and IEC 61131-3 within one DEVICE. The resulting mixed environment device allows now the integration of legacy systems and the use of already acquired IEC 61131-3 expertise, while still maintaining IEC 61499 distribution aspects. Furthermore, the modified SFB enables code sharing at development and execution time. Experiments have been conducted to verify, that both RTE parts are not affecting each other unintentionally. A demonstration application consisting of a pure IEC 61131-3 part, a pure IEC 61499 part, and a mixed application part is developed and analyzed. The experimental results show, that neither the cyclic execution of the IEC 61131-3 PROGRAM, nor the simultaneously executing event-driven IEC 61499 APPLICATION are disturbed. The presented united programming approach combines the best of both worlds for improved development efficiency.

Future research will focus on the incorporation of the remaining relevant languages Ladder Diagram (LD), and SFC of the IEC 61131-3 standard. Beyond that, also additional software elements such as FUNCTIONs are considered. Moreover, the integration of concepts like *global variables*, *access paths*, and *directly represented variables* need also further analysis, since conceptional contradictions are present. Finally, the possibility of distributed IEC 61131-3 PROGRAMs based on the proposed combined development framework will be analyzed, enabling improved development efficiency without the prejudice to key features for industrial automation implementations.

Post-print version of the article: P. Gsellmann, M. Melik-Merkumians, A. Zoitl, and G. Schitter, "A Novel Approach for Integrating IEC 61131-3 Engineering and Execution into IEC 61499," *IEEE Transactions on Industrial Informatics*, 2020. DOI: 10.1109/TII.2020.3033330

References

[1] W. Dai, V. Vyatkin, J. H. Christensen, and V. N. Dubinin, "Bridging Service-Oriented Architecture and IEC 61499 for Flexibility and Interoperability," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 771–781, 6 2015.

[2] International Electrotechnical Commission, *IEC 61131 – Programmable controllers, Part 3: Programming languages*, 2013.

[3] IEC TC65/WG6, *IEC 61499: Function blocks for industrial-process measurement and control systems – Parts 1 to 4*. Geneva: International Electrotechnical Commission (IEC), 2005.

[4] A. Zoitl, T. Strasser, C. Sünder, and T. Baier, "Is IEC 61499 in Harmony with IEC 61131-3," *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, pp. 49–55, 2009.

[5] M. Wenger and A. Zoitl, "Re-use of IEC 61131-3 Structured Text for IEC 61499," in *2012 IEEE International Conference on Industrial Technology*, 3 2012, pp. 78–83.

[6] M. Wenger, A. Zoitl, C. Sunder, and H. Steininger, "Transformation of IEC 61131-3 to IEC 61499 based on a model driven development approach," in *2009 7th IEEE International Conference on Industrial Informatics*, 6 2009, pp. 715–720.

[7] C. Gerber, H.-M. Hanisch, and S. Ebbinghaus, "From IEC 61131 to IEC 61499 for Distributed Systems: A Case Study," *EURASIP Journal on Embedded Systems*, vol. 2008, no. 1, p. 231630, 10 2007. [Online]. Available: https://doi.org/10.1155/2008/231630

[8] T. Hadlich, C. Diedrich, K. Eckert, T. Frank, A. Fay, and B. Vogel-Heuser, "Common communication model for distributed automation systems," in *2011 9th IEEE International Conference on Industrial Informatics*. IEEE, 7 2011.

[9] P. Gsellmann, M. Melik-Merkumians, and G. Schitter, "Comparison of Code Measures of IEC 61131-3 and 61499 standards for Typical Automation Applications," in *Proceedings of the 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation*, 2018, pp. 1047–1050. [Online]. Available: https://publik.tuwien.ac.at/files/publik_271875.pdf

[10] IEC TC65/WG6, *IEC 61131: Standard - Programmable controllers – Parts 1 to 8*. International Electrical Commission, 2003.

[11] C. Sünder, A. Zoitl, J. H. Christensen, H. Steininger, and J. Rritsche, "Considering IEC 61131-3 and IEC 61499 in the context of component frameworks," in *2008 6th IEEE International Conference on Industrial Informatics*. IEEE, 7 2008.

[12] S. Campanelli, P. Foglia, and C. A. Prete, "Integration of existing IEC 61131-3 systems in an IEC 61499 distributed solution," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*. IEEE, 9 2012.

[13] V. Vyatkin and J. Chouinard, "On comparisons of the ISaGRAF implementation of IEC 61499 with FBDK and other implementations," in *2008 6th IEEE International Conference on Industrial Informatics*, 7 2008, pp. 289–294.

[14] G. Cengic and K. Akesson, "On formal analysis of IEC 61499 applications, Part B: Execution semantics," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 2, pp. 145–154, may 2010.

[15] K. Thramboulidis, "IEC 61499: Back to the well proven practice of IEC 61131?" in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, 9 2012, pp. 1–8.

[16] J. M. Lastra, L. Godinho, and A. Lobov, "Closed loop control using an IEC 61499 application generator for scan-based controllers," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*. IEEE, 2005.

[17] J. M. Lastra, L. Godinho, A. Lobov, and R. Tuokko, "An IEC 61499 application generator for scan-based industrial controllers," in *INDIN '05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005.* IEEE, 2005.

[18] A. Zoitl, *Real-time Execution for IEC 61499*. ISA, 2008.

[19] L. H. Yoong, P. S. Roop, V. Vyatkin, and Z. Salcic, "A Synchronous Approach for IEC 61499 Function Block Implementation," *IEEE Transactions on Computers*, vol. 58, no. 12, pp. 1599–1614, 12 2009.

[20] M. Vallee, M. Merdan, W. Lepuschitz, and G. Koppensteiner, "Decentralized reconfiguration of a flexible transportation system," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 505–516, 8 2011.

[21] R. Schoop and A. Strelzoft, "Asynchronous and synchronous approaches for programming distributed control systems based on standards," *Control Engineering Practice*, vol. 4, no. 6, pp. 855–861, 6 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0967066196000780

[22] H. Mayer, "The bridge is built," *Computer & Automation*, 2013. [Online]. Available: http://www.computer-automation.de/steuerungsebene/steuern-regeln/artikel/93521/

[23] C. Sunder, M. Wenger, C. Hanni, I. Gosetti, H. Steininger, and J. Fritsche, "Transformation of existing IEC 61131-3 automation projects into control logic according to IEC 61499," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, 9 2008, pp. 369–376.

Peter Gsellmann received an MSc. in Energy Systems and Automation Technology from the Vienna University of Technology, Vienna, Austria in 2018 and is currently pursuing a PhD degree with the Automation and Control Institute of the Vienna University of Technology, Vienna, Austria. His primary research interests are on industrial automation, and vision-based control of industrial robots.

Martin Melik-Merkumians received his master diploma in Electrical Engineering with focus on Industrial Automation at TU Wien in 2009, and started immediately afterwards a doctoral program at the Automation and Control Institute within the group for Advanced Mechatronic Systems. His research topics are component-based automation, service-oriented methods and self-learning algorithms in automation systems, as well as knowledge-based development methods for industrial automation software.

Alois Zoitl is professor of Cyber-Physical Systems for Engineering and Production at the Johannes Kepler University, Linz, Austria. His research interests are in the area adaptive production systems, distributed control architectures, and dynamic reconfiguration of control applications as well as software development and software quality assurance methods for industrial automation. Since 2009 he is an active member of the IEC SC65B/WG15 for the distributed automation standard IEC 61499. He was named convenor of the group in May 2015.

Georg Schitter is Professor for Advanced Mechatronic Systems at the Automation and Control Institute (ACIN) of TU Wien. He received an MSc in Electrical Engineering from TU Graz, Austria (2000) and an MSc and PhD degree from ETH Zurich, Switzerland (2004). His primary research interests are on high-performance mechatronic systems, particularly for applications in the high-tech industry, scientific instrumentation, and mechatronic imaging systems, inline metrology systems, as well as automation and production systems. He received the journal best paper award of the IEEE/ASME Transactions on Mechatronics (2018), IFAC Mechatronics (2008-2010), Asian Journal of Control (2004-2005), and the 2013 IFAC Mechatronics Young Researcher Award. He served as an Associate Editor for IFAC Mechatronics, Control Engineering Practice, and for the IEEE Transactions on Mechatronics.

Post-print version of the article: P. Gsellmann, M. Melik-Merkumians, A. Zoitl, and G. Schitter, "A Novel Approach for Integrating IEC 61131-3 Engineering and Execution into IEC 61499," *IEEE Transactions on Industrial Informatics*, 2020. DOI: 10.1109/TII.2020.3033330