Distributed IEC 61499 Material Handling Control based on Time Synchronization with IEEE 1588

Cheng Pang, Jeffrey Yan, Valeriy Vyatkin
Department of Electrical and Computer Engineering
The University of Auckland
Auckland, New Zealand
E-mail: [cpan024, jyan110]@aucklanduni.ac.nz,
v.vyatkin@auckland.ac.nz

Steven Jennings
Interroll Holding GmbH
Hoeferhof 16
42929 Wermerlskirchen, Germany
E-mail: S.Jennings@interroll.com

***Abstract*— This paper proposes a method of time-driven control with high-precision synchronous clocks in distributed control systems built following the IEC 61499 standard. It investigates the impact of applying time-driven control on performance of material handling systems. A time-driven control system for a multi-diverter conveyor line has been developed using IEC 61499 Function Blocks architecture with support of the IEEE 1588 Precision Time Protocol. Analytic performance model has been developed and comparisons between the time-driven and two other possible control designs have been conducted and elaborated in terms of costs, logic design, and system throughput.**

***Keywords*- manufacturing automation; industrial control; industrial engineering**

I. Introduction

The shift from PLC (Programmable Logic Controller) based centralized systems to distributed intelligent systems is the main trend in the development of industrial automation systems. This trend was reflected in the development of the international standard IEC 61499 [1]. The standard supports the design paradigm based on function blocks (FB). A function block is design abstraction for a distributed process or a part thereof. Communication between processes is modelled in IEC 61499 using events with associated data. Therefore, this model primarily relies on event-driven activation of processes in distributed system and their asynchronous execution.

Time and process synchronization have always been important factors when designing the control of industrial automation systems, but not sufficiently addressed in IEC 61499 research and development. The way of time used in the control varies depending on the system's functionality. For example, in motion control systems and robotics, time is used to synchronize the actions of individual motors or to coordinate the axes of motion; while in material handling system (MHS), time can be used to stamp the input data and then to schedule the output actuation. The appropriate applications of time in the control design can help improve the system's throughput, simplify the logic design, and even reduce the hardware costs. Time-driven control design is more suitable for automation systems requiring high precision and performance than the traditional scan-based solution [2]. However, the control design is usually subject to the layout of the control system and available hardware devices. For instance, in centralized control systems, the use of time in control algorithms is more intuitive while in distributed control systems a fundamental issue is to establish and maintain the same notion of time among all the control devices.

In this paper we investigate the use of time-driven distributed IEC 61499 control in material handling systems, where the use of distributed automation approaches is most natural due to high spatial distribution and modularity of the machinery. Time-driven distributed control scenarios are compared to traditional ones based on central control and event-driven. Executable control specifications for all scenarios are implemented using IEC 61499 architecture. In particular, an implementation of the IEEE 1588 Precision Time Protocol (PTP) [3] using the RTS IEEE 1588 Network Stack [4] has been proposed to solve the time synchronization issue in distributed MHS.

The paper is organized as follows. Section II describes the overall layout and operations of the multi-diverter material handling system used as a test case in this work. Three possible control configurations are presented for this test case along with a conceptual mathematical model of material throughput. The model allows comparing the configurations in terms of system throughput. The related issues of cost savings and complexity of control logic design are also discussed. Then, Section III introduces the basic concepts of the IEEE 1588 PTP protocol and Section IV introduces basic concepts of IEC 61499 Function Block architecture. Section V presents the main ideas of using synchronized time in IEC 61499. The FB implementation details of the multi-diverter test cases are elaborated in Section VII. Finally, the paper is concluded in Section VIII.

II. Multi-Diverter Material Handling Test Case

The test case studied in this work consists of a single long conveyor line with a number of diverters as schematically illustrated in Figure 1.

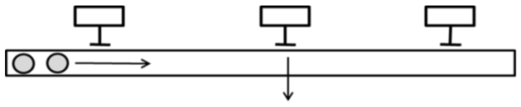

Figure 1. Multi-Diverter Material Handling Test Case

This work has been sponsored in parts by The University of Auckland, New Zealand, and Interroll GmbH, Germany

This multi-diverter conveyor line is typically used in material handling systems where different items must be first scanned to determine their attributes and then diverted to the corresponding locations further down the line. This sorting function can be achieved in a number of ways, which are summarized and discussed in the rest of this section.

In order to achieve quantitative comparisons, a set of variables and constraints has been defined to describe the configuration scenarios, as follows:

1. T_{div} is a constant denoting the time the diverter takes to divert an object and return to its ready state;
2. t_{cpu} specifies the controller's algorithm execution time;
3. t_{io} is the update time of the I/O configuration (remote I/O scan cycle);
4. t_{net} is the worst case delay time for a message to travel within the system;
5. $T_{scanner}$ is a constant denoting the time that the scanner takes to scan a single object;
6. W is the reaction time for a diverter to successfully divert an object; and,
7. R is the system's throughput rate in object per second.

The reaction times of the following configuration scenarios will be analyzed based on the assumption that the systems have reached the steady states.

A. Centralized Control with Remote I/O's

The centralized control with remote I/O's configuration, as shown in Figure 2, is one of the most common setups in MHS due to the architecture of PLC. In this configuration, a single powerful controller and a scanner are installed at the beginning of the conveyor line with remote I/O's attached to the diverters alongside. Items are detected down the line by the sensors and sorted by the diverters based on the signals sent by the central controller.

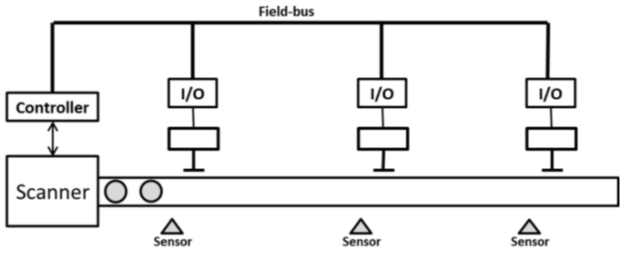

Figure 2. Centralized Control with Remote I/O's Configuration.

For each diversion, the reaction time of this configuration can be determined by the addition of the followings:

- $t_{io\text{-}cc}$: item arrives at sensor and sensor reading is placed on I/O;
- $t_{cpu\text{-}cc}$: controller executes algorithm and makes decision about diversion;
- $t_{io\text{-}cc}$: diverter actuator activated by controller; and,
- T_{div}: diverter physically pushes the item.

The reaction time, W_{cc}, can be formulated as:

$$W_{cc} = t_{io\text{-}cc} + t_{cpu\text{-}cc} + t_{io\text{-}cc} + T_{div}$$
$$= 2t_{io\text{-}cc} + t_{cpu\text{-}cc} + T_{div} \quad (1)$$

The throughput rate of the centralized control system, R_{cc}, can be thought of as limited by the reaction time of the system, thus:

$$R_{cc} \leq \frac{1}{W_{cc}} \quad (2)$$

It can be noticed that the jitter delay between sensor reading and diverter actuation limits the speed of conveyor belt, which is especially significant in PLC control systems. Both sensor reading and diverter actuation could potentially take a single I/O scan to complete.

Moreover, the scan time of remote I/O's increases as more remote I/O modules are added and subsequently increases the response time. For performance critical systems, running I/O with the lowest scan time is ideal. However, faster scan time results in lower bandwidth over the field-bus or network which could limit the scalability to larger system. Thus, to avoid this delay distributed control with dedicated I/O's configuration would be a reasonable option.

B. Distributed Scanning Control

Instead of having a powerful central controller, in the distributed control configuration, along each diverter there will be a scanner and a lightweight controller as indicated in Figure 3, where each diverter controller handles its own I/O.

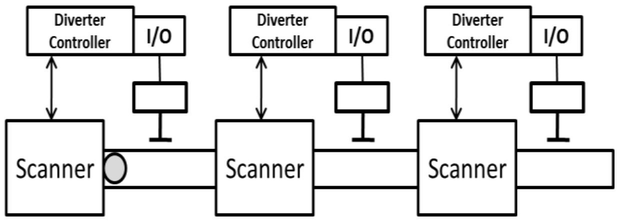

Figure 3. Distributed Control with Dedicated I/O's Configuration

This distributed architecture simplifies the control logic's design as each controller now only considers its own scan result and the diverter's actuation. By introducing dedicated scanners and I/O modules, the sensor delay in the centralized configuration is eliminated.

Although the sensor delay is avoided, there is an extra delay introduced by the additional scanners. On the other hand, the saving on sensors and powerful central controller may not cover the costs of additional scanners and lightweight controllers. Moreover, as the items' diverting decisions are made only at the points of diversion, it may be difficult for the distributed controllers to implement complex sorting criteria requiring a global view of the system. The reaction time of this configuration is determined by the addition of the following factors:

- $t_{io\text{-}dsc}$: item arrives at sensor and sensor reading is placed on I/O;
- $t_{cpu\text{-}dsc}$: controller executes algorithm and makes decision about diversion;
- $t_{io\text{-}dsc}$: diverter actuator activated by controller; and,
- $T_{div\text{-}dsc}$: diverter physically pushes the item.

Similar to the centralized scenario, the reaction time W_{dsc} for the distributed configuration can be formulated as:

$$W_{dsc} = t_{io\text{-}dsc} + t_{cpu\text{-}dsc} + t_{io\text{-}dsc} + T_{div}$$

$$= 2t_{io\text{-}dsc} + t_{cpu\text{-}dsc} + T_{div}. \quad (3)$$

The throughput rate for the distributed scanning control scenario is limited by the reaction time as well as the scanning time of the system, thus:

$$R_{dsc} = \frac{1}{T_{scanner} + W_{dsc}} \quad (4)$$

C. Time Synchronous Distributed Control

The clock-synchronized distributed configuration shown in Figure 4 is an improved variant of the previous distributed configuration. Each diverter controller stores a table of times at which it should activate the diverter. The table is updated by the main controller, which transmits the corresponding time upon scanning the next material item.

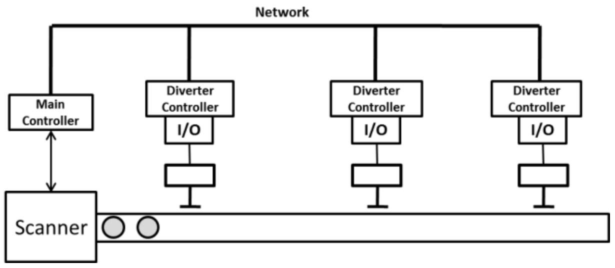

Figure 4. Synchronous Distributed Control Configuration

With the new configuration, the time spent on scanning is constant and complex sorting algorithms relying on a global view of the system are possible. The reaction time for this synchronous scenario can be determined by the following sequence of actions:

- $t_{cpu\text{-}tsdc}$: diverter controller waits until the corresponding divert time;
- $t_{io\text{-}tsdc}$: diverter actuator activated by the diverter controller; and,
- T_{div}: diverter physically pushes the item.

As a result, the reaction time W_{tsdc} can be calculated as:

$$W_{tsdc} = t_{cpu\text{-}tsdc} + t_{io\text{-}tsdc} + T_{div}. \quad (5)$$

The throughput rate of the synchronous distributed system is:

$$R_{tsdc} = \frac{1}{W_{tsdc}} \quad (6)$$

The gain in throughput of the synchronous scenario over the centralized scenario can be obtained as:

$$Gain = \frac{R_{tsdc} - R_{cc}}{R_{cc}} = \frac{\frac{1}{W_{tsdc}} - \frac{1}{W_{cc}}}{\frac{1}{W_{cc}}}$$

$$= \frac{\frac{1}{t_{cpu-tsdc} + t_{io-tsdc} + T_{div}} - \frac{1}{t_{cpu-cc} + 2t_{io-cc} + T_{div}}}{\frac{1}{t_{cpu-cc} + 2t_{io-cc} + T_{div}}}$$

$$= \frac{t_{cpu-cc} + 2t_{io-cc} + T_{div}}{t_{cpu-tsdc} + t_{io-tsdc} + T_{div}} - 1. \quad (7)$$

For simplicity purposes, it is assumed that the network related delays, such as propagation time and congestion, are negligible, and hence $t_{net} \approx 0$, which gives:

$$Gain = \frac{t_{cpu-cc} + 2t_{io-cc} + t_{div}}{t_{cpu-tsdc} + t_{io-tsdc} + t_{div}} - 1. \quad (8)$$

The diverter push time remains constant for all scenarios. The algorithm execution time and I/O scan time of the centralized scenario would increase depending on factors such as complexity of the application and the I/O configuration.

For comparison of the centralized and synchronous distributed control configurations some sample parameters are used. For an average diverter the divert time could be approximately 300ms. We assume for the centralized scenario an algorithm execution time of 20ms and a relatively fast I/O configuration that results in an I/O scan time of 1ms. For the synchronous scenario, each distributed controller runs a small amount of code and only manages a single I/O module thus algorithm complexity is less and algorithm execution times are much shorter. This system is assumed to have an algorithm execution time of 10ms and I/O scan time also of 1ms. A comparison is tabulated below:

TABLE I. PERFORMANCE GAIN

	Centralized	**Synchronous**
Alg. execution	20ms	10ms
I/O scan	1ms	1ms
Push duration	300ms	300ms
Total	20 + 1*2 + 300 =322	10 + 1 + 300 = 311

With these parameters, the overall performance gain is 3.5%. However, when more diverters are added to these systems, the main area affected will be the I/O scan time. If we assume the I/O scan time for the centralized system is a function of the number of diverters and other factors such as algorithm execution time remain constant, then the previous equation becomes:

$$Gain = At(n) + B \quad (9)$$

If we approximate the I/O scan time to be proportional to the number of diverters then we get a linear increase in gain as we increase the number of diverters.

Furthermore, in both the centralized scenario and the time synchronous distributed scenario, we assume the algorithm execution time to be constant. However this is not entirely true once both systems get extremely large. For the centralized scenario, if the algorithm that manages the diversion decisions, is a simple list comparison than the time taken to iterate the list will grow with the complexity of the system. Contrast this to the time synchronous distributed system where each diverter controller only manages the list of the items that it is required to divert. The algorithm complexity for the time synchronous distributed system will be much less than that of the centralized scenario and hereby the algorithm execution time will be much shorter.

However, to achieve this efficiency improvement, one fundamental issue is to precisely synchronize clocks in the distributed controllers and maintain this synchronization with minimum computation power. Otherwise, the items are likely to be mistakenly sorted. With synchronous time, the tasks of the controllers as shown in Figure 4 are simplified to the followings:

- Main controller records the timestamps of input items, makes decisions upon the sorting criteria, and then forwards the time-stamped orders to the downstream diverter controllers correspondingly; and,

- Downstream diverter controllers only need to actuate their own diverters at the time specified in the timestamps.

There are few protocols for time synchronization. Section III below elaborates the protocol used for the MHS test case.

III. The IEEE 1588 Precision Time Protocol

In this work, the IEEE 1588 PTP protocol is used to synchronize the system nodes in the MHS test case following the synchronous distributed configuration. The IEEE 1588 standard defines a protocol that enables precise synchronization of clocks in measurement and control systems. Comparing to other clock synchronization protocols, such as the widely used Network Time Protocol in large distributed computing systems, the PTP protocol addresses the synchronization needs of spatially localized distributed control systems requiring microsecond to sub-microsecond accuracy in the field of industrial automation. Distributed systems consisting of nodes capable of processing PTP messages over a network can adopt the PTP protocol.

In a PTP system, clocks are organized into a master-slave synchronization hierarchy with the top-level grandmaster clock determining the reference time for the entire system. All the other clocks ultimately derive their time from this grandmaster clock by exchanging PTP messages to synchronize their time with their masters in the hierarchy. In this way, the PTP protocol is intended to be administration free. Figure 5 shows a simple master-slave clock hierarchy.

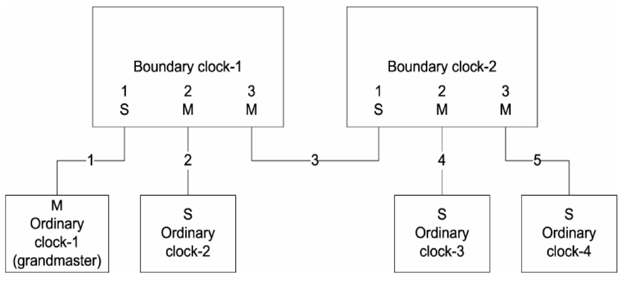

Figure 5. Simple Master-Slave Clock Hierarchy [3].

According to the number of ports, clocks are classified as either ordinary clock (single-port) or boundary clock (multi-port), where each port maintains its own state as:

- Master (M): the port is the source of time on the path;
- Slave (S): the port synchronizes to the master clock on the path; or,
- Passive (P): the port is neither a master clock nor synchronized to a master clock.

This master-slave clock hierarchy is established using the Best Master Clock (BMC) algorithm. By comparing the data describing the clocks' characteristics, such as accuracy and stability, the BMC algorithm concludes which clock is better and hence updates the clocks' states. This BMC algorithm will be executed whenever there is a change in the system, such as discovery of new clock or removal of existing clock. Once the clock hierarchy is established, clocks can be synchronized by exchanging the PTP messages over the communication path.

IV. Function Blocks Architecture of IEC 61499

The IEC 61499 standard establishes an event-driven modular architecture for designing the logic of distributed control systems. The basic building artefact in IEC 61499 is called Basic FB (BFB), which consists of an interface and an Execution Control Chart (ECC) as indicated in Figure 6:

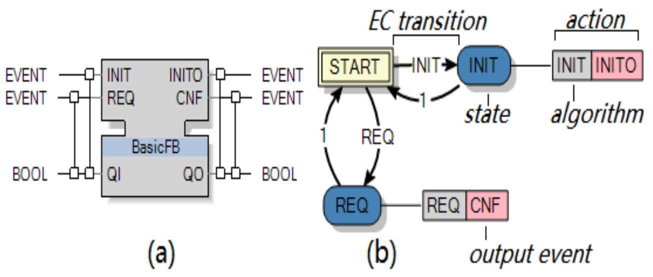

Figure 6. Basic Function Block: (a) Interface and (b) ECC

The FB interface consists of event and data I/O's, where event signals trigger the evaluation of the FB's algorithms and data signals stores evaluation results. The functionality of a BFB is defined in the ECC, which is a state machine whose semantics is similar to Moore finite automata with actions assigned to states. The state's transition condition is defined in the EC transition, which consists of an input event and a predicate over the data inputs and outputs. When a state transition occurs, algorithms in the associated actions will be executed and the corresponding output events will be issued.

A Composite Function Block, on the other hand, is specified by interface and functionality, defined as a network of function block instances interconnected via event and data connections.

FBs can be interconnected via event and data connections to form an FB Application (FBA), which is the highest level structure in the IEC 61499 hierarchy. As IEC 61499 is a deployable execution specification, by adding deployment details, such as control device layout and communication network, to the FBA, a deployable system configuration is created. Examples of the introduced artefacts will be encountered by the reader further in the paper.

V. IEEE 1588 Presision Time Protocol in IEC 61499 Function Block

In our previous work [5], we have discussed possible implementations of the PTP protocol at different levels of IEC 61499. A software-only implementation of the PTP protocol has also been elaborated. In this work, a hardware-supported off-the-shelf FB-based PTP solution is developed using the commercial RTS IEEE 1588 Network Stack and nxtStudio IEC 61499 IDE [6].

The RTS stack allows users to configure the PTP settings and network layout and then automatically explores the network to establish the communication paths. Then the master-slave clock hierarchy will be created and the clocks will be synchronized. To access the synchronized time, a Service Interface Function Block (SIFB), as shown in Figure 7 is developed.

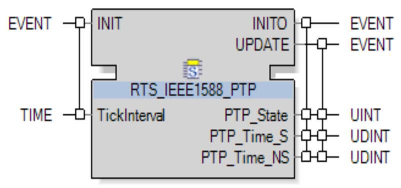

Figure 7. Interface of RTS_IEEE1588_PTP SIFB.

The main function of this SIFB is to retrieve the current PTP time from the stack via the RTS API and then provide this time to the control logic, where:

- *INIT* initializes the underneath RTS stack;
- *TickInterval* specifies how frequent the *UPDATE* event is emitted;
- *UPDATE* refreshes the associated data outputs;
- *PTP_State* indicates whether the current node is a Master, Slave, or not yet initialized; and,
- *PTP_Time_S* and *PTP_Time_NS* represent the second and nanosecond parts of current timestamp.

The RTS stack is continuously running at the background to maintain the local clock's synchronization. However, due to FB's event-driven semantics, the latest PTP time can only be available to other FBs when the *UPDATE* event is emitted. As a result, the *TickInterval* must be precise enough to match the time resolution required by the control logics.

VI. MHS Test Case Control Implementations

This section compares and discusses the logic design for the three MHS test case configurations. Specifically, the control logics are implemented using event-driven FBs with cyclically scanned I/O's. It is also aimed to demonstrate how time can be incorporated in current FB control design.

Figure 8 (a) illustrates the FB design for the centralized control configuration, where:

- *CentralControl* implements the control logic; and,
- *KL1408* and *KL2408* are FBs for accessing Beckhoff CX1010 digital I/O modules.

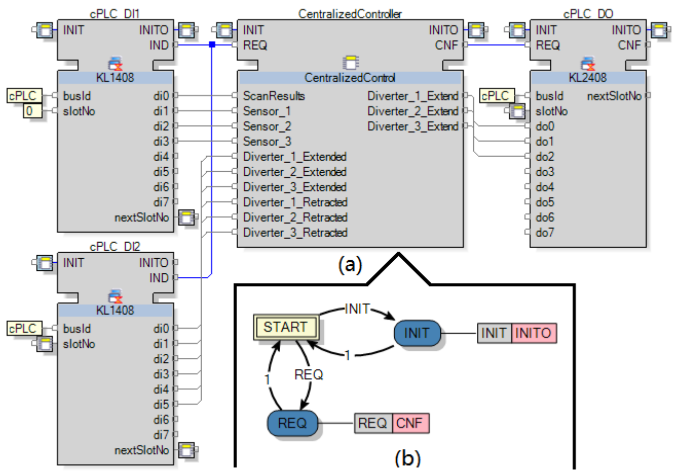

Figure 8. Function Block Design for Centralized Control: (a) FB Interfaces and (b) CentralizedControl FB's ECC

The structure of *CentralControl*'s ECC as shown in Figure 8 (b) is simple. However, the algorithms are complicated as the control logic must keep track of all the scan results and then based on the entire system's sensor values decide which diverters must be activated. The more I/O's in the system, the more complicated the control algorithms will be.

In contrast to the centralized control logic, each of the lightweight controllers in Figure 3 only needs to consider the scan result of the incoming item and makes sorting decision immediately. Figure 9 illustrates the FB design for the distributed configuration, where the *DistributedControl* FB's ECC is identical to Figure 8 (b).

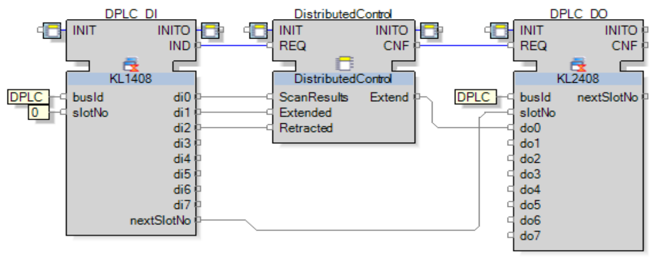

Figure 9. Function Block Design for Distributed Control

Unlike the isolated controllers in the distributed configuration, the diverter controllers in the synchronous configuration must be connected to the main controller to update their local timestamp records. Figure 10 shows the distributed system layout for the synchronous configuration, where controllers are communicating over an Ethernet.

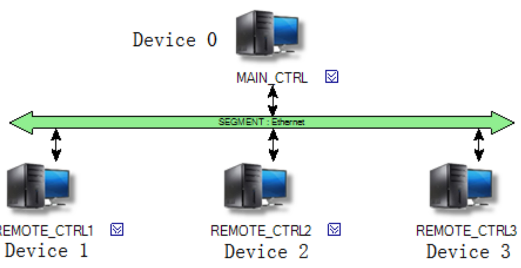

Figure 10. Overall System Layout for the Synchronous Configuration

Device0 contains the control logic for the main controller and holds the most accurate clock in the system as illustrated in Figure 11.

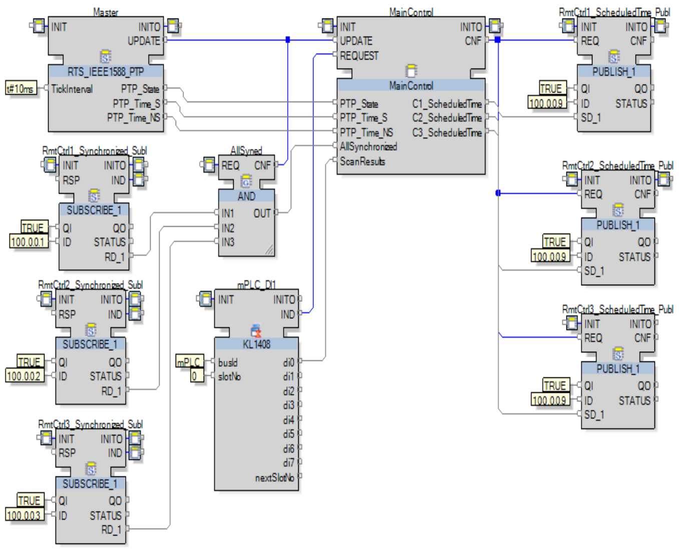
Master
RTS_IEEE1588_PTP
MainControl
RmtCtrl1_Synchronized_Subl
RmtCtrl2_Synchronized_Subl
RmtCtrl3_Synchronized_Subl
AllSynced
AND
mPLC_DI
KL1408
RmtCtrl1_ScheduledTime_Publ
RmtCtrl2_ScheduledTime_Publ
RmtCtrl3_ScheduledTime_Publ
SUBSCRIBE_1
PUBLISH_1
C1_ScheduledTime
C2_ScheduledTime
C3_ScheduledTime
AllSynchronized
ScanResults

Figure 11. Function Block application in the main control device.

An instance of *RTS_IEEE1588_PTP* provides the PTP time for the *MainControl* FB, which receives the scan result from the *KL1408* FB and communicates with the remote controllers through the *PUBLISH*/*SUBSCRIBER* FB pairs.

As indicated in Figure 12, upon the *UPDATE* event input, the *MainControl* FB checks whether the associated PTP clock is properly initialized and the remote controllers are synchronized. Then upon the *REQUEST* event input, the new scan result will be examined based on the sorting criteria and the next diverting time for the corresponding diverter controller will be scheduled.

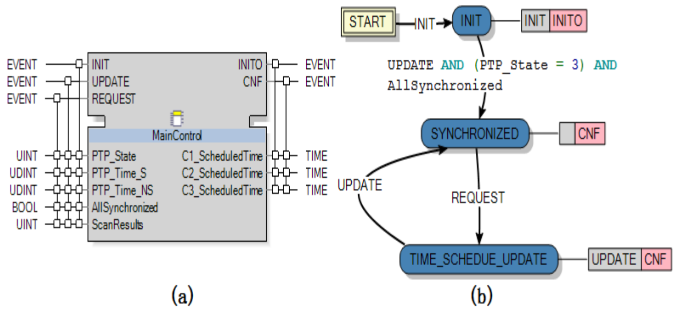

Figure 12. Interface and ECC of the MainControl Function Block

Comparing to the control logic of the centralized configuration, the main controller does not need to keep track of all the scan results. Instead, the scan results are converted into timestamps and stored in the corresponding remote controller as indicated in Figure 13.

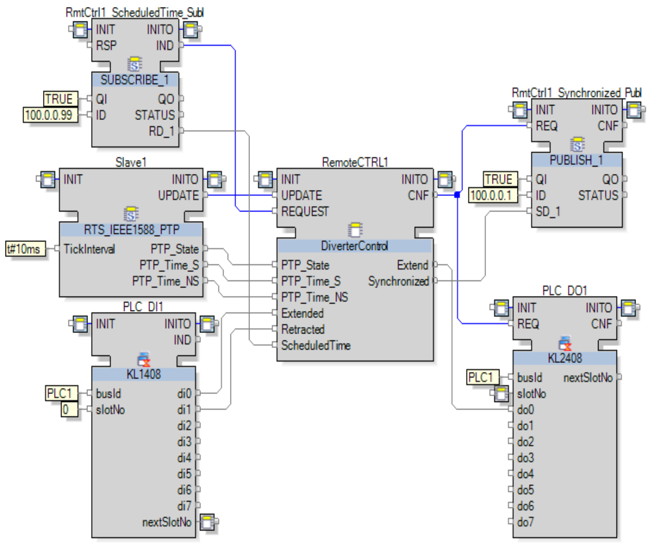

Figure 13. Remote Control Device

As the scan results are examined in the main controller, the remote diverter controller only needs to match the stored timestamps with current time and then actuate the diverter. As shown in Figure 14, the remote controller first waits for the associated PTP clock to be synchronized with the Master clock. Then, upon every *REQUEST* event input, it stores the next scheduled diversion time. At last, on every *UPDATE* event input, the remote controller will only need to compare its current time with the first stored timestamp. If matched, the diverter will be actuated. By converting the scan results into diversion timestamps, the overall logic design of the synchronous configuration is much simpler than the centralized configuration.

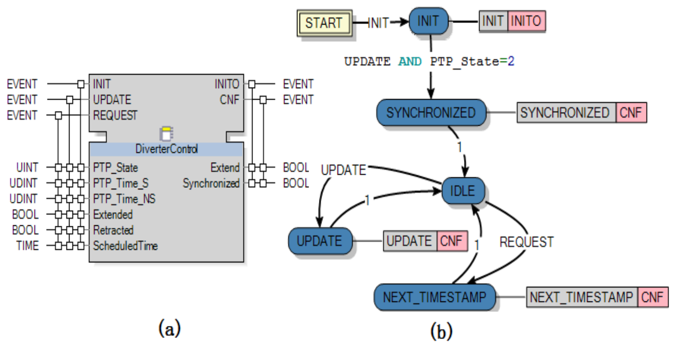

Figure 14. Interface and ECC of RemoteIOCtrl Function Block

VII. Conclusions

Appropriate application of time-driven control design in industrial automation can improve the system performance, simplify the logic design, and reduce the costs. This paper demonstrated the use of the IEEE 1588 PTP protocol to solve the time synchronization issue when designing time-driven control for distributed automation systems with IEC 61499 architecture. It has been demonstrated that comparing with centralized control configuration the synchronous distributed configuration can deliver throughput performance gain of 3.5% for our simple 3-diverter test case. The gain increases linearly with the number of diverters in the system.

The proposed method can be applied in any domain that requires distributed precision time control despite unpredictable asynchronous nature of IEC 61499 run-time platforms and communication networks like Ethernet. In future work we are going to confirm the presented analytic estimations in experiment and extend the case studies to new domains, such as distributed automation of SmartGrid.

Acknowledgment

The authors are grateful to Real-Time Systems GmbH for providing trial version of IEEE 1588 stack, and to NxtControl (Austria) for providing NxtStudio and NxtForte software. The authors appreciate fruitful discussions with William Dai.

References

[1] International Electrotechnical Commission, "IEC 61499- Function blocks for industrial-process measurement and control systems - Part 1: Architecture," ed. Geneva: International Electrotechnical Commission, 2005, p. 111.

[2] K. Harris, "An application of IEEE 1588 to Industrial Automation," in Precision Clock Synchronization for Measurement, Control and Communication, 2008. ISPCS 2008. IEEE International Symposium on, 2008, pp. 71-76.

[3] IEEE Standard 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," ed. New York, 2008, p. 289.

[4] Real-Time Systems GmbH. (2011, April). *RTS IEEE 1588 Network Stack [Online]*. Available: http://www.real-time-systems.com/ieee_1588/index.php

[5] C. Pang, V. Vyatkin and C. Fantuzzi, "Time-Complemented Event-Driven Control Framework for Distributed Motion Control Systems," in *9th IEEE Conference on Industrial Informatics (INDIN 2011)*, Caparica, Lisbon, Portugal, 2011.

[6] nxtControl. (2011, April). *nxtStudio [Online]*. Available: http://www.nxtcontrol.com/en.html