

IEC 61499 Execution Timing

Definitions

The IEC 61499 standard defines a distributed model for splitting different parts of an industrial automation process and complex machinery control into functional modules called function blocks. These function blocks can be distributed and interconnected across multiple controllers.

System: A collection of devices interconnected and communicating with each other by means of a communication network consisting of segments and links.

Device: An independent physical entity capable of performing one or more specified functions in a particular context and delimited by its interfaces.

Resource: A functional unit having independent control of its operation, and which provides various services to applications including scheduling and execution of algorithms.

Application: A software functional unit that is specific to the solution of a problem in industrial-process measurement and control. An application may be distributed among devices and may communicate with other applications.

Function block: A software functional unit that is the smallest element of a distributed control system. It utilizes an execution control chart (ECC) state machine to control the execution of its algorithms.

Overview

A distributed application exchanges data over the communication interface. An application may consist of one or more function blocks where the input sampling is performed in one function block, control processing is performed in a second function block, and output conversion is performed in a third function block. This distributed application may run function blocks within one resource or across multiple resources. These resources are part of one device or multiple devices.

A function block is a functional unit of software comprising an individual instance or a copy within a resource.

An automation and process control application distributed across multiple resources and devices requires synchronization to guarantee data integrity and good behavior. Figure 1 shows an IEC 61499 cooperative application having two function blocks that are connected together and part of a single application. When FB1 outputs its data and event to FB2, it is important that the event signal correspond to the Data signal because FB2 must receive the event and the data signal corresponded. The WITH qualifier from IEC 61499 indicates that the corresponding event and data signals must be synchronised.

Since **ISaGRAF** applications event and data signals are synchronized, each time an event is sent from a function block, the corresponding data is valid. Therefore, the receiving function block has valid data associated with the event. Each time a function block consumes its inputs, it retrieves all events and data from other function blocks linked to its inputs. Also, a function block produces all of its output signals simultaneously.

In the application shown in figure 1, FB1 can send multiple events and data regardless of what FB2 does with these. The delay between two consecutive events is represented by **Te**. This time delay depends on the cycle time of the function block. The propagation or transmission delay between FB1 and FB2 is represented by **Tt**. The network link adds on this communication time. Faster links provide shorter communication propagation times. The processing delay

of FB2 is represented by **Tp**. This time delay is the cycle time of the virtual machine running the function block. The total time needed to process this event and data is given in the following calculation:

$$\text{Total execution time} = T_t + T_p$$

To avoid wasting events, the time between events should be greater than the total processing time. Otherwise, FB2 will not have time to process all events coming from FB1. Some events could go unused since there is no queue or buffer. When the time **Te** between events is greater than the total processing time, then FB2 receives all events without wasting one and has the time to process them.

$$T_e > T_t + T_p$$

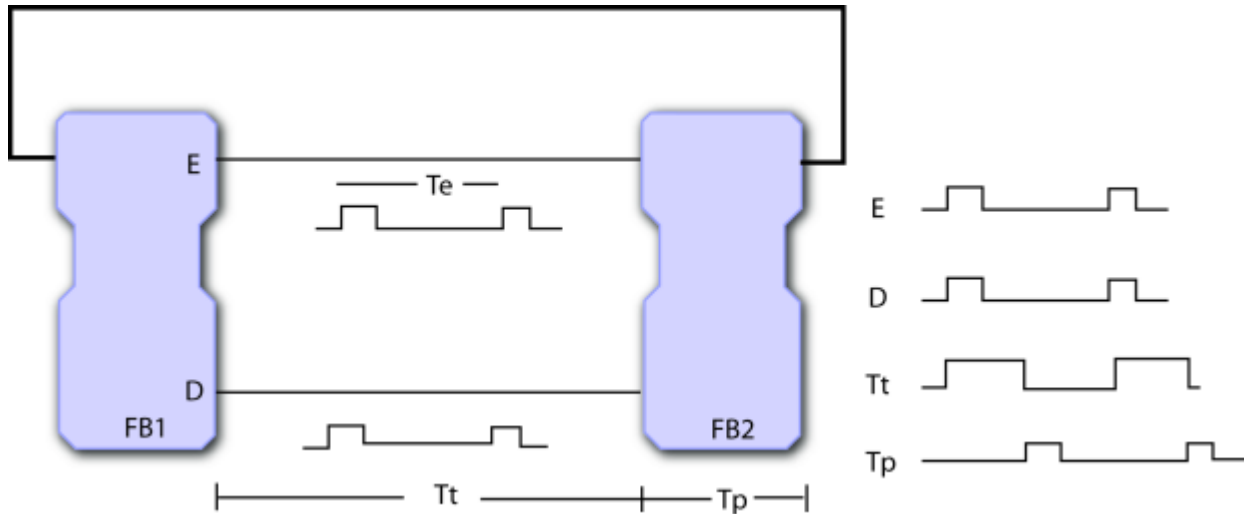


Figure 1: IEC 61499 Cooperative Application

A programming rule such as shown in figure 1 requires time computation as well as trial and error to fine tune an application. The preferred method of programming a distributed system is to ensure all distributed function blocks are synchronized.

Figure 2 shows an **ISaGRAF** cooperative application having a synchronizing signal. This preferable programming methodology sets FB1 and FB2 in perfect synchronization. FB1 only produces its event and data when FB2 is ready to receive them. The **ACK** signal from FB2 signaling its readiness to process another event accomplishes this synchronization. After having produced its event and data, FB1 remains on standby awaiting another **ACK** signal. This methodology wastes no events and data signals.

In a complex application where many function blocks are connected together, synchronization is facilitated when defining the timing signals between the function blocks. This synchronization eliminates additional delays that could affect the behavior of the application. A toolset such as **ISaGRAF** provides all the functionalities required to build distributed applications. Like any other programming language, a good methodology helps build a more stable and comprehensible application.

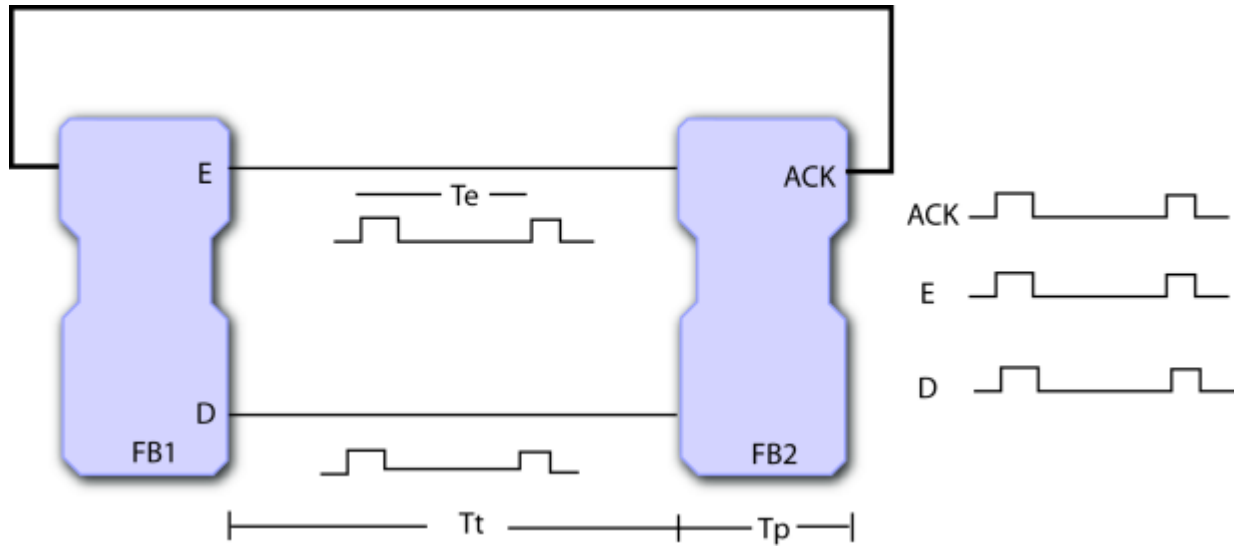


Figure 2: ISaGRAF Cooperative Application

In all cases, an application is said to be real time when the total time required to transmit the signals (T_t), process the logic in FB1 (T_{p1}), and process the logic in FB2 (T_{p2}) is faster than the process under control.

References

International Electrotechnical Commission: Function Blocks Part 1 - Architecture (61499-1 © CEI:200X).

ICS Triplex ISaGRAF Inc.: ISaGRAF User's Guide. November 2005.