IEC 61499 Practical Hints

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

Building a distributed application is not always simple. What is the starting point? What is the cycling time? How to get IO and variable values? How to exchange data between applications and devices? What are good programming practices and data integrity? This application note is all about these issues.

An application may consist of one or more function blocks where 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 a single 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 copy within a resource. An automation and process control application having many function blocks requires synchronization to guarantee data integrity and good behavior.

Starting Point

Figure 1 shows the RESTART function block that sets the starting point of an application. All function blocks contained in an application should be initialized and have a proper starting point. Since individual function blocks can run on any resource or device, their order of execution must be defined. The RESTART function block clearly indicates the starting point of the application. This function block sends an event when the resource runs for the first time, then all other function blocks forming the application start running.

The RESTART function block can be connected to all function blocks contained in the application. In this case, when the function blocks receive the RESTART event, these switch to the running state. Also, the RESTART function block can be connected to a single function block. In this case, the first function block receives the RESTART event and starts running, then its output event and data signals trigger other function block contained in the application.

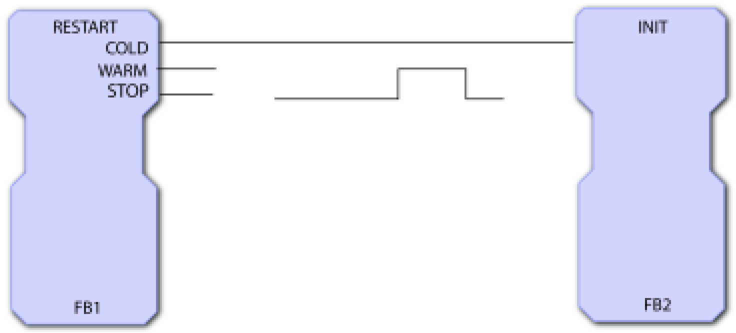

Figure 1: Application Starting Point

Cycling Time

An application can run one time or periodically (cyclically). Function blocks can be on standby awaiting an event signal to start running. Signals coming from input devices or human machine interfaces trigger these function blocks. In applications needing to run periodically, a common practice is to use the PERIODIC function block providing the cycle time to the application. This PERIODIC function block sends an event signal at a specific time interval and sets the cycle time for the control loop. In this type of application, the PERIODIC event should exceed the total execution time of the application. Otherwise, the function block will deviate from the defined cycle time. The total execution time of an application is equal to the propagation delay between function blocks and the internal execution times of the algorithms defined for all individual function blocks.

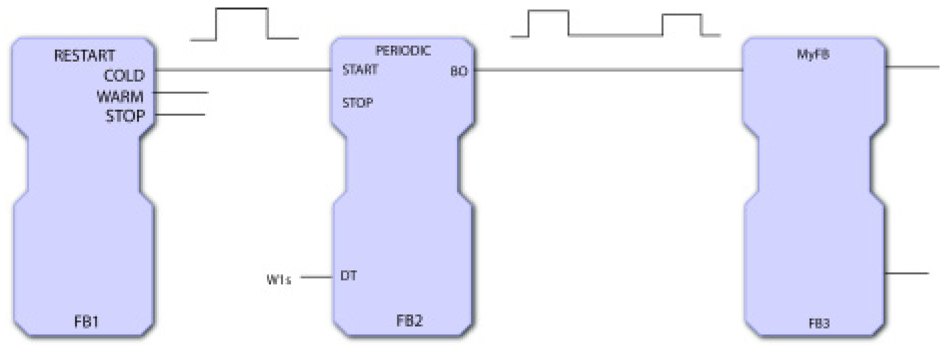

Figure 2: Periodic (Cyclic) Application

Service Interface

An application needs to read and write to the external world. Accessing IO device values, variable values, and communication values is common in any control and automation application.

IEC 61499 application schematic disallow variables since these require declared instances on a resource and device. An interfacing mechanism is needed. Function blocks specializing in such purposes are called Service Interface Function Blocks. These function blocks are the atomic object used for distribution application rather than variables. Therefore, an application must contain service interface function blocks to enable reaching IOs, variables, and communication values.

Figure 3 shows an application having an input service interface (IN) and an output service interface (OUT). The input service interface retrieves values from IO points coming from an IO device driver as well as variable values coming from a resource database or a communication interface such as an OPC server or field bus link. The output service interface sends data values to external IO devices, variable databases, and a communication link.

Individual function blocks contained in an application can reach internal variables on their own, therefore, these function blocks do not require service interfaces. However, Adding service interfaces to an application increases the reuse of function blocks and programming flexibility.

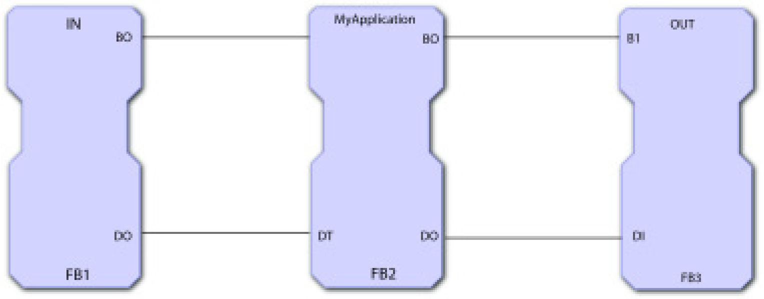

Figure 3: Service Interface Function Blocks

Data Integrity

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.

Figure 4 shows two function blocks interfacing with FB2. How are the event and data signals synchronized since FB1 and FB3 do not talk to each other? This type of programming brings about data integrity problems in the application. FB2 gets the event and data signals from both function blocks correctly, however, the synchronization of the event and data signals is uncertain. Moreover, the data coming from FB3 may not be ready when FB1 outputs its event signal.

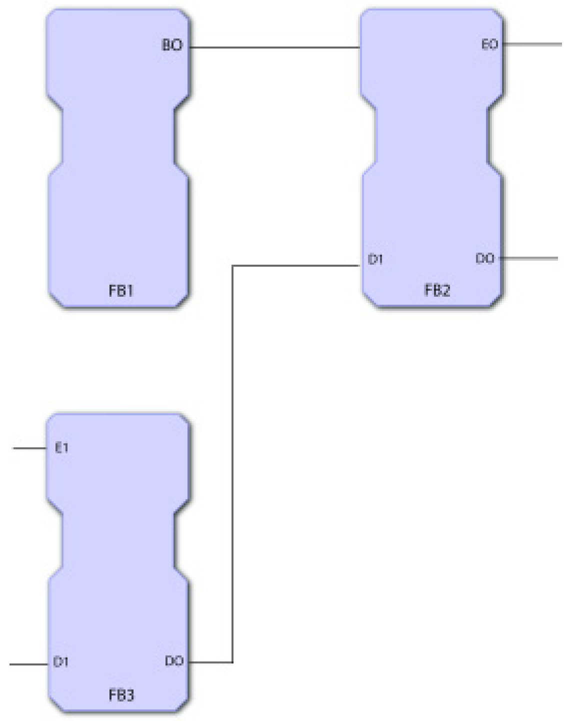

Figure 4: Absence of Data Integrity

Figure 5 shows FB3 receiving its event and data signals correctly. When FB1 sends its event signal, it asks FB2 to prepare the data and drive FB3 correctly. Such good programming practices save a lot of debugging time.

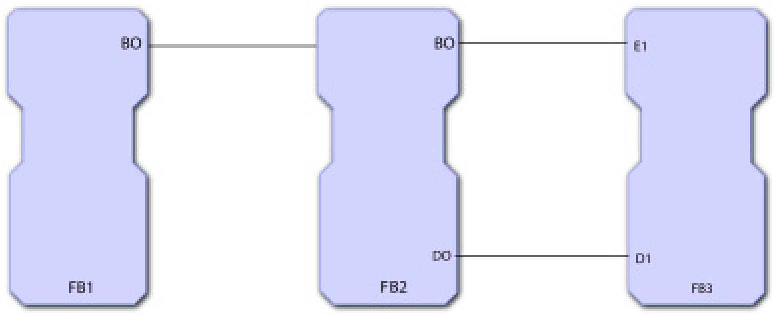

Figure 5: Data Integrity

References

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

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