Splitter

Component Design Document

1 Description

This is a generic component that can be used to split a single connector of any type into an arrayed connector of that type. This can be useful when a component has a single send connector, but you actually need the data to go to many different places simultaneously. In this case, the splitter component can be attached to the send connector and then distribute the data to many downstream components.

2 Requirements

No requirements have been specified for this component.

3 Design

3.1 At a Glance

Below is a list of useful parameters and statistics that give a quick look into the makeup of the component.

- **Execution** - *passive*
- **Number of Connectors** - 2
- **Number of Invokee Connectors** - 1
- **Number of Invoker Connectors** - 1
- **Number of Generic Connectors** - *None*
- **Number of Generic Types** - 1
- **Number of Unconstrained Arrayed Connectors** - 1
- **Number of Commands** - *None*
- **Number of Parameters** - *None*
- **Number of Events** - *None*
- **Number of Faults** - *None*
- **Number of Data Products** - *None*
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

Figure 1: Splitter component diagram.

3.3 Connectors

Below are tables listing the component's connectors.

3.3.1 Invokee Connectors

The following is a list of the component's *invokee* connectors:

Table 1: Splitter Invokee Connectors

Name	Kind	Type	Return_Type	Count
T_Recv_Sync	recv_sync	T (generic)	-	1

Connector Descriptions:

- **T_Recv_Sync** - The generic invokee connector.

3.3.2 Invoker Connectors

The following is a list of the component's *invoker* connectors:

Table 2: Splitter Invoker Connectors

Name	Kind	Type	Return_Type	Count
T_Send	send	T (generic)	-	<>

Connector Descriptions:

- **T_Send** - The generic arrayed invoker connector.

3.4 Initialization

Below are details on how the component should be initialized in an assembly.

3.4.1 Generic Component Instantiation

The splitter is generic in that it can be instantiated to split a stream of any type at compile time. This component contains generic formal types. These generic formal types must be instantiated with a valid actual type prior to component initialization. This is done by specifying types for the following generic formal parameters:

Table 3: Splitter Generic Formal Types

Name	Formal Type Definition
T	type T is private;

Generic Formal Type Descriptions:

- **`T`** - The generic type of data passed in and out of the splitter.

3.4.2 Component Instantiation

This component contains no instantiation parameters in its discriminant.

3.4.3 Component Base Initialization

This component achieves base class initialization using the `init_Base` subprogram. This subprogram requires the following parameters:

Table 4: Splitter Base Initialization Parameters

Name	Type
`T_Send_Count`	`Connector_Count_Type`

Parameter Descriptions:

- **`T_Send_Count`** - The size of the T_Send invoker connector array.

3.4.4 Component Set ID Bases

This component contains no commands, events, packets, faults or data products that need base indentifiers.

3.4.5 Component Map Data Dependencies

This component contains no data dependencies.

3.4.6 Component Implementation Initialization

This component contains no implementation class initialization, meaning there is no `init` subprogram for this component.

4 Unit Tests

None

5 Appendix

5.1 Packed Types

The following section outlines any complex data types used in the component in alphabetical order. This includes packed records and packed arrays that might be used as connector types, command arguments, event parameters, etc..

No complex types found in component.