Forwarder

Component Design Document

1 Description

This is a generic component that can be used to forward a single connector of any type. The component that synchronously forwards any type that it receives. It includes commands to enable or disable this forwarding, so can be effectively used as a stream on/off switch.

2 Requirements

The requirements for the Forwarder component are specified below.

1. The component shall be able to receive and send a generic data type.
2. The component shall respond to commands to enable and disable data forwarding.

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** - 7
- **Number of Invokee Connectors** - 2
- **Number of Invoker Connectors** - 5
- **Number of Generic Connectors** - *None*
- **Number of Generic Types** - 1
- **Number of Unconstrained Arrayed Connectors** - *None*
- **Number of Commands** - 2
- **Number of Parameters** - *None*
- **Number of Events** - 3
- **Number of Faults** - *None*
- **Number of Data Products** - 1
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

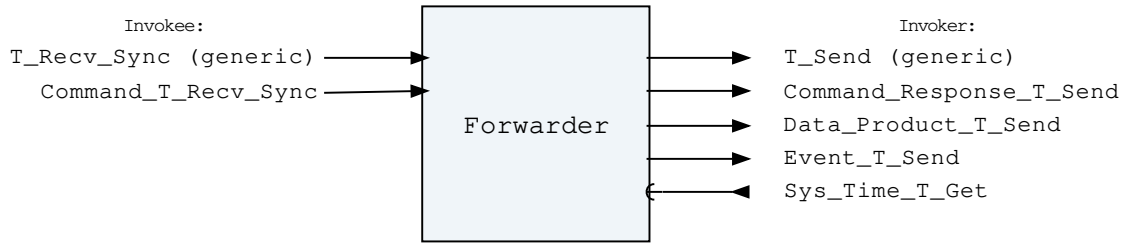

Figure 1: Forwarder component diagram.

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: Forwarder Invokee Connectors

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

Connector Descriptions:

- **T_Recv_Sync** - This connector is the input connector for the data that is coming in.
- **Command_T_Recv_Sync** - This is the command receive connector.

3.3.2 Invoker Connectors

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

Table 2: Forwarder Invoker Connectors

Name	Kind	Type	Return_Type	Count
T_Send	send	T (generic)	-	1
Command_Response_T_Send	send	Command_Response.T	-	1
Data_Product_T_Send	send	Data_Product.T	-	1
Event_T_Send	send	Event.T	-	1
Sys_Time_T_Get	get	-	Sys_Time.T	1

Connector Descriptions:

- **T_Send** - The connector that will forward on unfiltered data.
- **Command_Response_T_Send** - The connector that sends a command response when received.
- **Data_Product_T_Send** - The connector for data products
- **Event_T_Send** - The event connector to send the events specifc to the component.

- **Sys_Time_T_Get** - The system time is retrieved via this connector.

3.4 Interrupts

This component contains no interrupts.

3.5 Initialization

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

3.5.1 Generic Component Instantiation

The forwarder is generic in that it can be instantiated to forwarder 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: Forwarder 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 forwarder.

3.5.2 Component Instantiation

This component contains no instantiation parameters in its discriminant.

3.5.3 Component Base Initialization

This component contains no base class initialization, meaning there is no `init_Base` subprogram for this component.

3.5.4 Component Set ID Bases

This component contains commands, events, packets, faults, or data products that require a base identifier to be set at initialization. The `set_Id_Bases` procedure must be called with the following parameters:

Table 4: Forwarder Set Id Bases Parameters

Name	Type
`Command_Id_Base`	`Command_Types.Command_Id_Base`
`Event_Id_Base`	`Event_Types.Event_Id_Base`
`Data_Product_Id_Base`	`Data_Product_Types.Data_Product_Id_Base`

Parameter Descriptions:
- **Command_Id_Base** - The value at which the component's command identifiers begin.
- **Event_Id_Base** - The value at which the component's event identifiers begin.
- **Data_Product_Id_Base** - The value at which the component's data product identifiers begin.

3.5.5 Component Map Data Dependencies

This component contains no data dependencies.

3.5.6 Component Implementation Initialization

The calling of this implementation class initialization procedure is mandatory. The component achieves implementation class initialization using the `init` subprogram. The `init` subprogram requires the following parameters:

Table 5: Forwarder Implementation Initialization Parameters

Name	Type	Default Value
`Startup_Forwarding_State`	`Basic_Enums.Enable_Disable_Type.E`	*None provided*

Parameter Descriptions:

- **`Startup_Forwarding_State`** - Is the data stream enabled or disabled on startup. Disable means that the component does not forward any data it receives at startup.

3.6 Commands

These are the commands for the Forwarder component.

Table 6: Forwarder Commands

Local ID	Command Name	Argument Type
0	`Enable_Forwarding`	-
1	`Disable_Forwarding`	-

Command Descriptions:

- **`Enable_Forwarding`** - Enable the flow of data.
- **`Disable_Forwarding`** - Disable the flow of data.

3.7 Parameters

The Forwarder component has no parameters.

3.8 Events

Below is a list of the events for the Forwarder component.

Table 7: Forwarder Events

Local ID	Event Name	Parameter Type
0	`Forwarding_Enabled`	-
1	`Forwarding_Disabled`	-
2	`Invalid_Command_Received`	`Invalid_Command_Info.T`

Event Descriptions:

- **`Forwarding_Enabled`** - Data forwarding was enabled by command.

- **Forwarding_Disabled** - Data forwarding was disabled by command.
- **Invalid_Command_Received** - A command was received with invalid parameters.

3.9 Data Products

Data products for the Forwarder component.

Table 8: Forwarder Data Products

Local ID	Data Product Name	Type
0x0000 (0)	Forwarding_State	Packed_Enable_Disable_Type.T

Data Product Descriptions:
- **Forwarding_State** - Is data forwarding enabled or disabled?

3.10 Data Dependencies

The Forwarder component has no data dependencies.

3.11 Packets

The Forwarder component has no packets.

3.12 Faults

The Forwarder component has no faults.

4 Unit Tests

The following section describes the unit test suites written to test the component.

4.1 *Forwarder_ Tests* Test Suite

This is a unit test suite for the Forwarder component.

Test Descriptions:
- **Test_Init** - This unit test tests initialization.
- **Test_Enable_Disable_Forwarding** - This unit test sends commands to enable and disable forwarding and make sure it functions correctly.
- **Test_Invalid_Command** - This unit test exercises that an invalid command throws the appropriate event.

5 Appendix

5.1 Preamble

This component contains no preamble code.

5.2 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..

Command.T:

Generic command packet for holding arbitrary commands

Table 9: Command Packed Record : 808 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Command_ Header.T	-	40	0	39	-
Arg_Buffer	Command_Types. Command_Arg_ Buffer_Type	-	768	40	807	Header.Arg_ Buffer_Length

Field Descriptions:

- **Header** - The command header
- **Arg_Buffer** - A buffer to that contains the command arguments

Command_Header.T:

Generic command header for holding arbitrary commands

Table 10: Command_Header Packed Record : 40 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Source_Id	Command_Types. Command_Source_Id	0 to 65535	16	0	15
Id	Command_Types. Command_Id	0 to 65535	16	16	31
Arg_Buffer_Length	Command_Types. Command_Arg_Buffer_ Length_Type	0 to 96	8	32	39

Field Descriptions:

- **Source_Id** - The source ID. An ID assigned to a command sending component.
- **Id** - The command identifier
- **Arg_Buffer_Length** - The number of bytes used in the command argument buffer

Command_Response.T:

Record for holding command response data.

Table 11: Command_Response Packed Record : 56 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit

Source_Id	Command_ Types.Command_ Source_Id	0 to 65535	16	0	15
Registration_ Id	Command_ Types.Command_ Registration_ Id	0 to 65535	16	16	31
Command_Id	Command_Types. Command_Id	0 to 65535	16	32	47
Status	Command_Enums. Command_ Response_ Status.E	0 => Success 1 => Failure 2 => Id_Error 3 => Validation_Error 4 => Length_Error 5 => Dropped 6 => Register 7 => Register_Source	8	48	55

Field Descriptions:

- **Source_Id** - The source ID. An ID assigned to a command sending component.
- **Registration_Id** - The registration ID. An ID assigned to each registered component at initialization.
- **Command_Id** - The command ID for the command response.
- **Status** - The command execution status.

Data_Product.T:

Generic data product packet for holding arbitrary data types

Table 12: Data_Product Packed Record : 344 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Data_Product_ Header.T	-	88	0	87	-
Buffer	Data_Product_ Types.Data_ Product_ Buffer_Type	-	256	88	343	Header.Buffer_ Length

Field Descriptions:

- **Header** - The data product header
- **Buffer** - A buffer that contains the data product type

Data_Product_Header.T:

Generic data_product packet for holding arbitrary data_product types

Table 13: Data_Product_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63

Id	Data_Product_Types. Data_Product_Id	0 to 65535	16	64	79
Buffer_Length	Data_Product_ Types.Data_Product_ Buffer_Length_Type	0 to 32	8	80	87

Field Descriptions:

- **`Time`** - The timestamp for the data product item.
- **`Id`** - The data product identifier
- **`Buffer_Length`** - The number of bytes used in the data product buffer

Event.T:

Generic event packet for holding arbitrary events

Table 14: Event Packed Record : 344 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Event_Header.T	-	88	0	87	-
Param_Buffer	Event_Types. Parameter_ Buffer_Type	-	256	88	343	Header.Param_ Buffer_Length

Field Descriptions:

- **`Header`** - The event header
- **`Param_Buffer`** - A buffer that contains the event parameters

Event_Header.T:

Generic event packet for holding arbitrary events

Table 15: Event_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Event_Types.Event_ Id	0 to 65535	16	64	79
Param_Buffer_Length	Event_Types. Parameter_Buffer_ Length_Type	0 to 32	8	80	87

Field Descriptions:

- **`Time`** - The timestamp for the event.
- **`Id`** - The event identifier
- **`Param_Buffer_Length`** - The number of bytes used in the param buffer

Invalid_Command_Info.T:

Record for holding information about an invalid command

Table 16: Invalid_Command_Info Packed Record : 112 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Id	Command_Types.Command_Id	0 to 65535	16	0	15
Errant_Field_Number	Interfaces.Unsigned_32	0 to 4294967295	32	16	47
Errant_Field	Basic_Types.Poly_Type	-	64	48	111

Field Descriptions:

- **Id** - The command Id received.
- **Errant_Field_Number** - The field that was invalid. 1 is the first field, 0 means unkown field, 2**32 means that the length field of the command was invalid.
- **Errant_Field** - A polymorphic type containing the bad field data, or length when Errant_Field_Number is 2**32.

Packed_Enable_Disable_Type.T:

Single component record for holding an enable/disable enumeration.

Table 17: Packed_Enable_Disable_Type Packed Record : 8 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
State	Basic_Enums.Enable_Disable_Type.E	0 => Disabled 1 => Enabled	8	0	7

Field Descriptions:

- **State** - The 8-bit enable disable enumeration.

Sys_Time.T:

A record which holds a time stamp using GPS format including seconds and subseconds since epoch (1-5-1980 to 1-6-1980 midnight).

Table 18: Sys_Time Packed Record : 64 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Seconds	Interfaces.Unsigned_32	0 to 4294967295	32	0	31
Subseconds	Interfaces.Unsigned_32	0 to 4294967295	32	32	63

Field Descriptions:

- **Seconds** - The number of seconds elapsed since epoch.
- **Subseconds** - The number of 1/(2^32) sub-seconds.

5.3 Enumerations

The following section outlines any enumerations used in the component.

Basic_Enums.Enable_Disable_Type.E:

This enumeration includes enable and disable state.

Table 19: Enable_Disable_Type Literals:

Name	Value	Description
Disabled	0	The state is disabled.
Enabled	1	The state is enabled.

Command_Enums.Command_Response_Status.E:

This status enumerations provides information on the success/failure of a command through the command response connector.

Table 20: Command_Response_Status Literals:

Name	Value	Description
Success	0	Command was passed to the handler and successfully executed.
Failure	1	Command was passed to the handler not successfully executed.
Id_Error	2	Command id was not valid.
Validation_Error	3	Command parameters were not successfully validated.
Length_Error	4	Command length was not correct.
Dropped	5	Command overflowed a component queue and was dropped.
Register	6	This status is used to register a command with the command routing system.
Register_Source	7	This status is used to register command sender's source id with the command router for command response forwarding.